UNIVERSIDAD NACIONAL DE SAN CRISTÓBAL DE HUAMANGA

FACULTAD DE INGENIERÍA DE MINAS, GEOLOGÍA Y CIVIL

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

"ADAPTABILIDAD DEL PATRON MVC DEL FRAMEWORK .NET A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX. CASO DE ESTUDIO: REGISTRO DE ESTUDIOS AMBIENTALES DEL DREMA"

Tesis presentada por	:	Bach. Samuel Kalip Yarcuri Paredes
Para optar el título profesional de	:	Ingeniero de Sistemas
Tipo de investigación	:	Obsevacional, retrospectiva, descriptiva
Área de investigación	:	Ingeniería de Software
Asesor	:	Mg. Ing. Hubner Janampa Patilla

Ayacucho - Perú

2019

DEDICATORIA

A mí querida madre: Orfelinda Paredes, a mis hermanas Rebeca y Samy, a Karina. Por su gran amor y apoyo incondicional, por enseñarme a ser responsable y luchar para seguir adelante, y sobretodo porque gracias a ellas he logrado esta meta, a Cesia, Eduardo y Kaleb por ser una inspiración de aprendizaje continúa.

AGRADECIMIENTO

A Dios, por permitirme vivir y seguir el camino correcto, con objetivos claros, luchando por obtener lo que más quiero y por darme la oportunidad de que mi familia se sienta orgulloso de mis triunfos.

A mis profesores: por tener la vocación de compartir el conocimiento y confiar siempre en los alumnos. A la Escuela de Formación Profesional de Ingeniería de Sistemas y a la Universidad Nacional Nacional de san Cristóbal de Huamanga: por haber sido mi segunda casa y lugar de grandes vivencias

CONTENIDO

Pág.

CAPITULO IV

ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

RESUMEN

En el desarrollo de aplicaciones web tradicionales existe la dificultad de la separación de la lógica de negocio, y la lógica de presentación, además es difícil realizar pruebas automáticas debido a que ASP.NET se basa en un entorno en tiempo de ejecución monolítico, que se puede extender en cierta medida, pero no es un sistema flexible y conectable.

Por otro lado la Dirección Regional de Energía y Minas DREMA es un órgano desconcentrado de la Gerencia Regional de Desarrollo Económico del Gobierno Regional de Ayacucho, tiene relación directa con los usuarios en materia de Minería, Energía, Ambientales y Sociales en coordinación con el Ministerio de Energía y Minas (MINEM), el Instituto Geológico Minero Metalúrgico (INGEMMET), y el Organismo Supervisor de la Inversión de Energía y Minería (OSINERMIN), teniendo presente la importancia del uso de las nuevas tecnologías de información en sus procesos internos, las cuales hasta el momento, no se han utilizado, motivo por el cual es tomado como caso de estudio.

El objetivo de esta investigación fue determinar que el Patrón MVC del Framework .Net es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX. Caso de estudio: registro de estudios ambientales del DREMA.

La presente investigación se respaldó por un marco teórico, siendo una investigación de tipo experimental y aplicada. Finalmente, y como consecuencia de lo expuesto, se concluyó que el Patrón MVC del Framework .Net es un mecanismo de adaptabilidad a la arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

Palabras claves

Patron MVC, Framework .Net, Estudio Ambiental, ICONIX, Aplicación web, Visual Studio.

INTRODUCCIÓN

MVC es un patrón arquitectural; que define en qué bloques (o capas) estructuramos lógicamente nuestra aplicación (Modelo, Vista y Controlador), pero además detalla las responsabilidades exactas de cada capa y la forma que tienen de relacionarse entre sí. En el ámbito .NET, existe el framework ASP.NET MVC, que aporta una forma distinta de crear aplicaciones web. Este marco de trabajo ofrece una forma distinta de trabajar, aprovechando las ventajas del uso del patrón, más ligero, simple y cercano al funcionamiento real de los sistemas web.

La razón que motivo la realización de esta investigación es por la evolución de las nuevas tecnologías ha permitido el mejoramiento de los procesos administrativos en las empresas a nivel mundial, convirtiéndolas en organismos competitivos y de alta calidad, el gran potencial que estos tienen por ser un sector que evoluciona día tras día, y además apoyar a La Dirección Regional de Energía y Minas (DREMA) en las consultas diarias que realizan dentro y fuera de la institución.

Actualmente el DREMA ha venido aceptando la importancia de la información. Pensando en esto se tomo como caso de uso sistematizar los procesos de gestión de Estudios Ambientales, que son instrumentos de consulta que a diario se realizan dentro y fuera de la Institución.

La investigación se centrará en la adaptabilidad del patron MVC del framework .NET a la Arquitectura Tecnica de la Metodologia Agil y Formal ICONIX. Caso de estudio: registro de estudios ambientales del DREMA, y tiene como objetivos específicos: utilizar el Modelo ADO.NET Entity, el Motor de Vista Razor y las Clases Controladores C# del Patrón MVC del Framework .Net como mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

CAPITULO I

DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

1.1. DIAGNÓSTICO Y ENUNCIADO DEL PROBLEMA

En el desarrollo de aplicaciones web tradicionales existe la dificultad de la separación de la lógica de negocio, y la lógica de presentación, además es difícil realizar pruebas automáticas debido a que ASP.NET se basa en un entorno en tiempo de ejecución monolítico, que se puede extender en cierta medida, pero no es un sistema flexible y conectable. Además para lograr mantener el estado de la aplicación es necesario guardar el mismo en un control oculto de la página, ocasionando esto perdida de funcionalidad.

Por un lado, MVC es un patrón arquitectural; que define en qué bloques (o capas) estructuramos lógicamente nuestra aplicación (Modelo, Vista y Controlador), pero además detalla las responsabilidades exactas de cada capa y la forma que tienen de relacionarse entre sí.

En el ámbito .NET, existe el framework ASP.NET MVC, que aporta una forma distinta de crear aplicaciones web. Este marco de trabajo ofrece una forma distinta de trabajar, aprovechando las ventajas del uso del patrón, más ligero, simple y cercano al funcionamiento real de los sistemas web.

La Dirección Regional de Energía y Minas es un órgano desconcentrado de la Gerencia Regional de Desarrollo Económico del Gobierno Regional de Ayacucho, desarrolla acciones conducentes a la promoción, orientación y asesoramiento relativos a la actividad Minero, Energético y ambiental. El DREMA tiene relación directa con los usuarios en materia de Minería, Energía, Ambientales y Sociales en coordinación con el Ministerio de Energía y Minas (MINEM), el Instituto Geológico Minero Metalúrgico (INGEMMET), y el Organismo Supervisor de la Inversión de Energía y Minería (OSINERMIN), Así mismo, coordina con las empresas Mineras, Eléctricas, Asociaciones de Mineros funcionarios del Gobierno.

En la actualidad el DREMA ha venido aceptando la importancia de la información. Pensando en esto, ha decidido usar la tecnología para sistematizar los procesos de gestión de Estudios Ambientales, que son instrumentos de consulta que a diario se realizan dentro y fuera de la Institución. Lo anterior lleva a la necesidad de diseñar un software, mediante la utilización de herramientas tecnológicas para agilizar los procesos relacionados con la documentación producida y recibida, con el objeto de facilitar su uso, control, almacenaje, recuperación, clasificación, ordenación, descripción, protección, conservación y difusión.

En el presente trabajo de investigación se adaptara la arquitectura técnica de la metodología Ágil y Formal ICONIX al patrón MVC, y dividiremos el sistema que proponemos en tres capas.

1.2. FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN

PROBLEMA PRINCIPAL

¿Cómo el patrón MVC del framework .Net se adapta a la arquitectura técnica de la metodología ágil y formal ICONIX, 2018?

PROBLEMAS ESPECÍFICOS

a) ¿De que manera el modelo ADO.NET Entity Framework del Patrón MVC se adapta a la arquitectura técnica de la metodología ágil y formal ICONIX?
b) ¿De que manera el modelo del motor de vista razor del patrón MVC del Framework .Net se adapta a la arquitectura técnica de la metodología ágil y formal ICONIX?
c) ¿De que manera el modelo de clases controladores C# del patrón MVC del framework .Net se adapta a la arquitectura técnica de la metodología ágil y formal ICONIX?

1.3. OBJETIVOS DE LA INVESTIGACIÓN

OBJETIVO GENERAL

Adaptar el patrón MVC del framework .Net a la arquitectura técnica de la metodología ágil y formal ICONIX, 2018.

OBJETIVOS ESPECÍFICOS

a) Utilizar el modelo ADO.NET entity framework del patrón MVC del framework .Net a la arquitectura técnica de la metodología ágil y formal ICONIX.

b) Utilizar el motor de vista razor del patrón MVC del framework .Net a la arquitectura técnica de la tetodología ágil y formal ICONIX.

c) Utilizar las clases controladores C# del patrón MVC del framework .Net a la arquitectura técnica de la metodología ágil y formal ICONIX.

1.4. JUSTIFICACIÓN Y DELIMITACIÓN DE LA INVESTIGACIÓN

IMPORTANCIA DEL TEMA

La importancia del presente trabajo de investigación radica en la adaptabilidad de la arquitectura técnica de la metodología Ágil y Formal Iconix para con el marco de desarrollo Framework ASP.NET, aprovechando además sus librerías de clases y lenguaje de ejecución común.

JUSTIFICACIÓN

Se plantea adaptar la Arquitectura Técnica de la metodología Ágil y Formal Iconix al patrón MVC, y dividir el sistema en tres capas, lo cual ofrece una forma más flexible para el desarrollo de aplicaciones web y la realización de pruebas automáticas en ASP.NET, aprovechando las ventajas del uso del patrón más ligero y simple.

DELIMITACIÓN

La presente investigación se realizará en la arquitectura técnica de la metodología Ágil y Formal ICONIX.

CAPITULO II

REVISIÓN DE LA LITERATURA

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Garcia (2013), en su investigación: Estudio del estado del arte sobre frameworks MVC para el desarrollo de aplicaciones web cliente, caso de estudio ember.js, concluyo; a) El patrón de diseño MVC es el que en esencia da respuesta a la necesidad de que una aplicación web sea escalable, a través de la separación de funciones. b) La implementación del modelo MVC en frameworks para el desarrollo de aplicaciones web cliente, ofrece el mantenimiento de un código estructurado. c) Se elimina la figura del Web máster en el desarrollo web que todo lo hacía, actualmente se dividen las cargas en un desarrollo front-end y back-end.

Vázquez (2010), en la investigación: Impacto de implementaciones web del patrón MVC en los requisitos de calidad percibidos, concluye: en este trabajo de investigación se muestra que una implementación del patrón Modelo-Vista-Controlador, en los cuales los conectores y componentes adopten el estilo de tuberías y filtros supera en términos de tiempo de respuesta y escalabilidad a la alternativa tradicional de llamada-retorno.

Izquierdo (2012), en la investigación: Arquitectura de Software para el Modelado de Dominios en Sistemas Informáticos, concluye: que la arquitectura ha surgido de la aplicación sucesiva del Principio de Especialización. Éste no solo ha separado datos y operaciones sino que además a estas últimas se las ha extraído a su vez tres responsabilidades: comprobar si se puede hacer la operación (monitor), saber todas las entidades que deben ser afectadas (reacciones) y recuperar el modelo si algo de lo anterior falla (reconstructores). Por tanto las operaciones se quedan en la verdadera esencia de la tarea a realizar lo cual facilita su implementación, comprensión y reutilización.

2.2. MARCO TEÓRICO

2.2.1. METODOLOGÍA ICONIX

"ICONIX proceso es un análisis de casos de uso basada en el diseño y la metodología. Su foco principal está en cómo llegar de forma fiable a partir de casos de uso de código en menor número de pasos posible". (Rosenberg y Stephens, 2007).

Para Porras (2011), la metodología ICONIX esta compuesta por; a) Primer paso.- Identificar el mundo real y los objetivos de dominio del negocio (modelo de dominio), b)Segundo paso.- Definir los requisitos de comportamiento (casos de uso), c)Tercer paso.- Realizar análisis de robustez para eliminar la ambigüedad de los casos de uso y determinar los defectos del modelo de dominio (diagrama de robustez), d) Cuarto paso.- Asignar comportamiento a los objetos (diagrama de secuencia), e) Quinto paso.- Finalizar el modelo estático (diagrama de clases), f) Sexto paso. Escribir y generar el código (código fuente), g) Séptimo paso.- Realizar pruebas de aceptación (prueba).

El Proceso de Iconix se divide en flujos de trabajo dinámicos y estáticos, que son altamente repetitivo; usted puede ir a través de una repetición de todo el proceso para una pequeña cantidad de casos de uso (quizás un par de paquetes de valor, que no es una cantidad enorme teniendo en cuenta que cada caso de uso es sólo un par de párrafos), todo el camino a la fuente de código y pruebas unitarias. Por esta razón, el proceso Iconix es muy adecuado para proyectos ágiles, donde se necesita información rápida sobre factores tales como los requisitos, el diseño, y las estimaciones (Rosenberg y Stephens, 2007, p. 35).

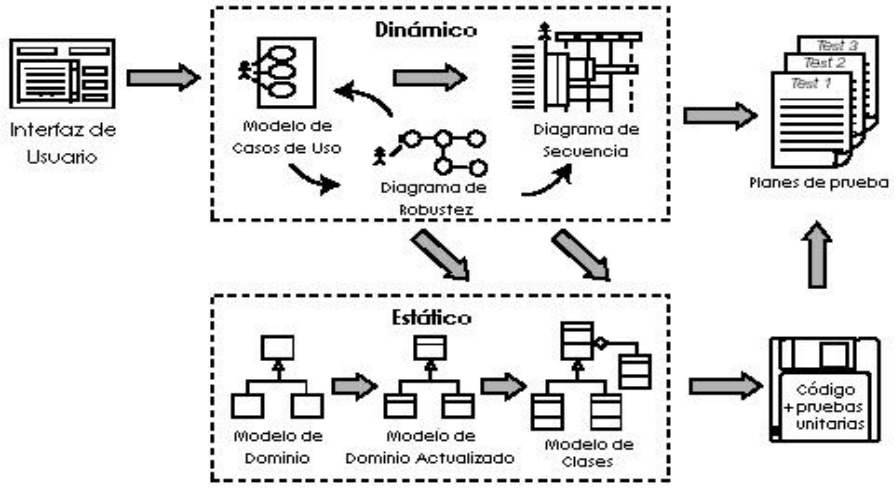

Figura N° 2.1: Flujo de trabajo Iconix (Rosenberg y Sthephens, 2007)

Rosenberg y Stephens (2007), en el análisis de requisitos se tiene; a) Requisitos funcionales.-Definir lo que el sistema debe ser capaz de hacer. Dependiendo de la forma en que su proyecto está organizado, ya sea que usted participe en la creación de los requisitos funcionales o que los requisitos serán "dictados desde lo alto" de un cliente o un equipo de analistas de negocios, b) Modelo de Dominio.- Entender el espacio del problema en términos inequívocos (sin ambigüedad), c) Requisitos de comportamiento.- Define la forma en que el usuario y el sistema interactúan (es decir, escribir el primer proyecto de casos de uso). Le recomendamos que empezar con un prototipo GUI (lo que llamamos "guión" GUI) e identificar todos los casos de uso que vamos a poner en práctica, o, al menos, llegar a una primera lista de casos de uso, que espera razonablemente cambiar a medida que se explora los requisitos con mayor profundidad. Inicia la etapa 1 con la revisión de requisitos, debe asegúrese de que la descripción de los caso de uso coincidan con las expectativas de sus clientes. Tenga en cuenta que se deben revisar los casos de uso en pequeños lotes, justo antes de diseñarlos.

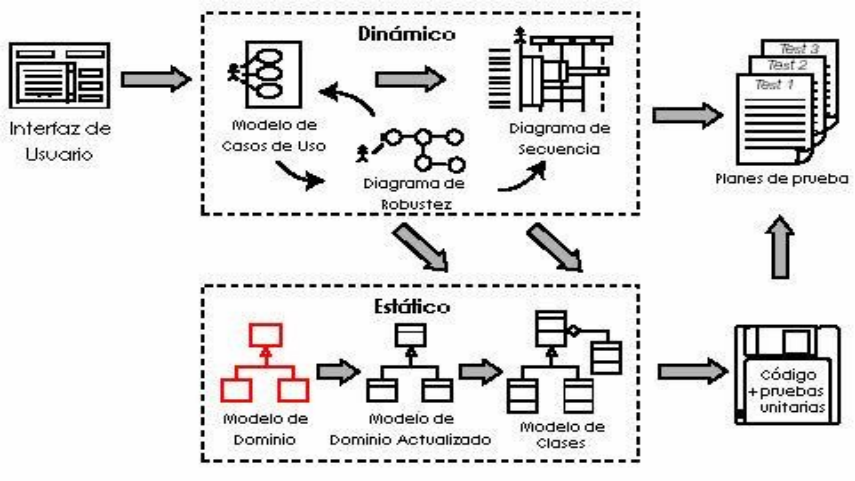

Figura N° 2.2: Modelo de dominio (Rosenberg y Sthephens, 2007)

Para Porras (2011), los modelos de casos de uso son desarrollados en cooperación con el modelo de dominio; los casos de uso sirven de para realizar las pruebas de funcionalidades durante la fase de implementación; un caso de uso es una secuencia de acciones que un actor realiza en el sistema para alcanzar un objetivo.

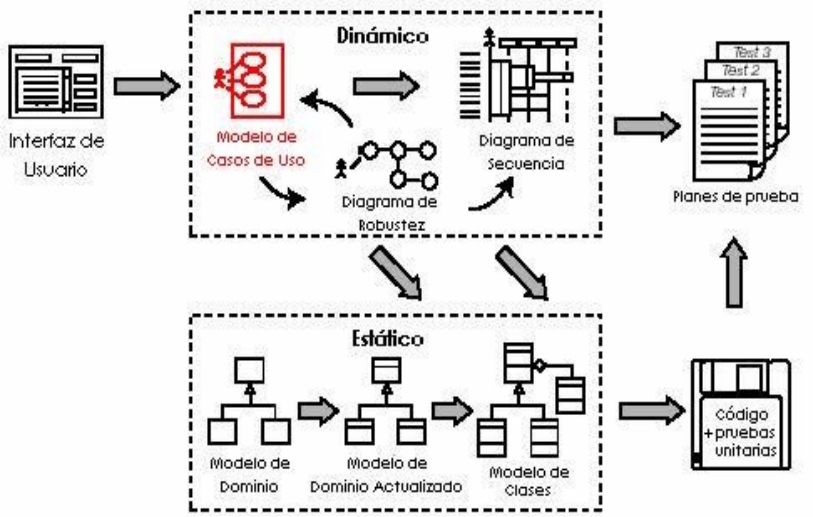

Figura N° 2.3: Modelo de casos de uso (Rosenberg y Sthephens, 2007)

Rosenberg y Stephens (2007), la revisión de requisitos garantiza que el sistema tal y como se describe coincide con los requisitos. Se trata de un período de sesiones de colaboración que impliquen al representante(s) del cliente, los usuarios finales (es decir, las personas que realmente van a utilizar el sistema, o quien está usando el sistema actual que se sustituirá), y las personas de marketing-básicamente, todos los stakeholders que tienen un interés en asegurar que los requisitos encajen con su punto de vista del sistema.

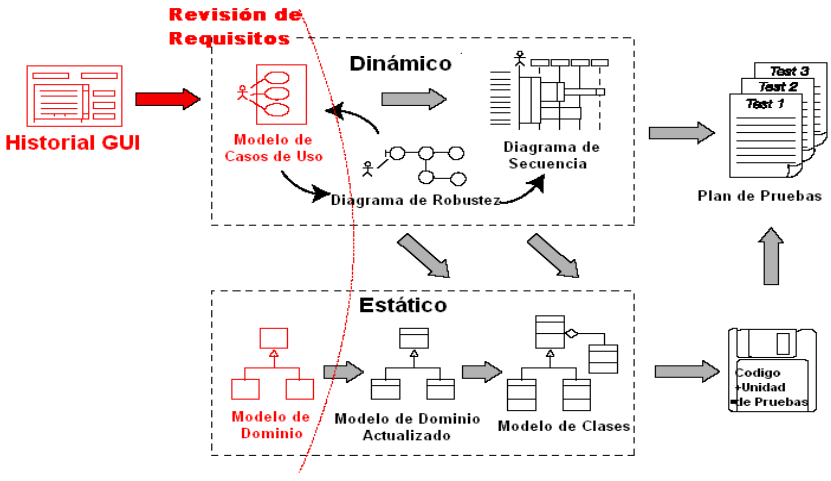

Figura N° 2.4: Revisión de requisitos (Rosenberg y Sthephens, 2007)

Porras (2011), en el diseño preliminar se tiene; a) Dibujar diagrama de robustez.- Es una "imagen del objeto" descripción paso por paso de cada uno de los casos de uso, reescribir los casos de uso a medida que avanza, b) Actualizar modelo de dominio.-

Mientras escribe los casos de uso y dibuja el diagrama de robustez, descubrirá algunas clases “pérdidas”, corregir las ambigüedades y, añadir atributos a los objetos de dominio, c) Nombrar controladores.- Nombre todas las funciones lógicas del software, necesarios para que los casos de uso funcionen, d) Escribir.- Reescribir el borrador de los casos de uso. Inicia la etapa 2 con la revisión del diseño preliminar.

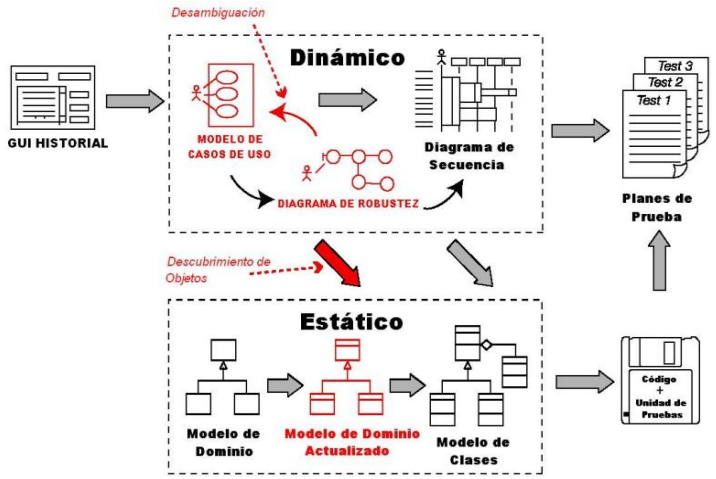

Figura N° 2.5: Análisis de robustez (Rosenberg y Sthephens, 2007)

Para Rosenberg y Stephens (2007), la revisión de diseño preliminar le ayuda a asegurarse de que los diagramas de robustez, el modelo de dominio, y los casos de uso concuerden mutuamente. Esta revisión es el "paso" entre las fases del Diseño Preliminar y el Diseño Detallado, para cada paquete de casos de uso.

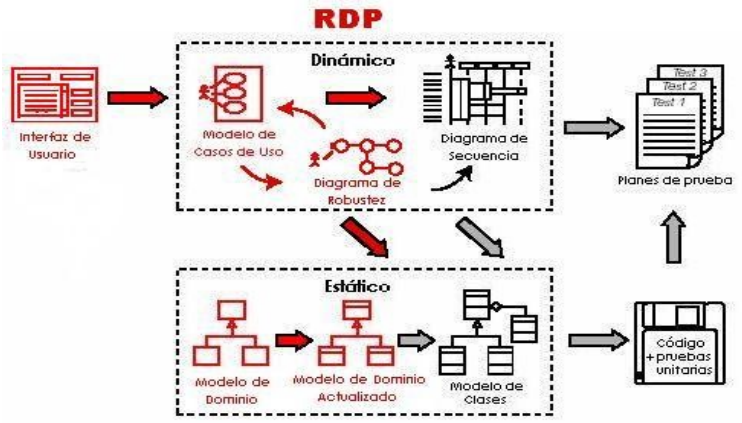

Figura N° 2.6: Revisión de diseño preliminar (Rosenberg y Sthephens, 2007)

Rosenberg y Stephens (2007), en el diseño detallado se tiene; a) Diagrama de Secuencia.- Dibuje un diagrama de secuencia (un diagrama de secuencia por cada caso de uso) para mostrar en detalle cómo va a implementar cada caso de uso. La

función fundamental de los diagramas de secuencia es asignar comportamiento para sus clases,

b) Actualice el modelo de dominio mientras se dibujan los diagramas de secuencia, y añada operaciones a los objetos de dominio. En esta fase, los objetos de dominio son realmente clases de dominio, o entidades, el modelo de dominio debe convertirse rápidamente en un modelo estático, o diagrama de clases-una parte crucial de su diseño detallado, c) Limpiar el modelo estático. Inicia la etapa 3 con la revisión del diseño crítico (RDC).

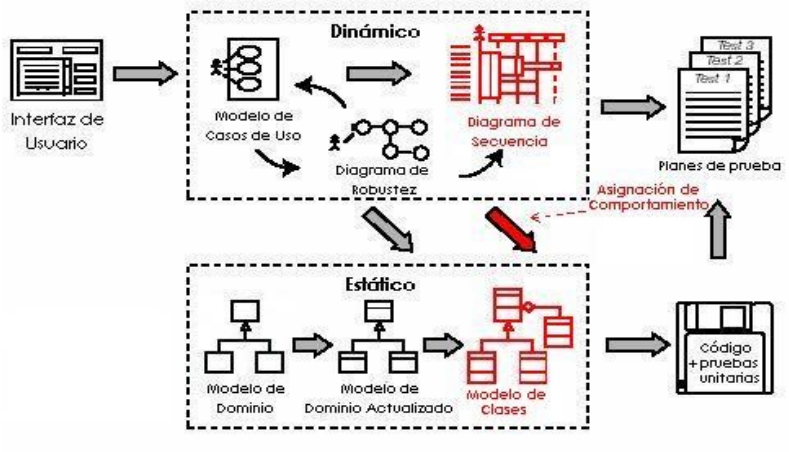

Figura N° 2.7: Diagrama de secuencia (Rosenberg y Sthephens, 2007)

Para Porras (2011), la revisión crítica del diseño es un paso entre el diseño y la implementación, usamos tres criterios; hacer coincidir la descripción de los casos de uso con los diagramas de secuencia, verificar la continuidad de los mensajes y revisar para un buen diseño.

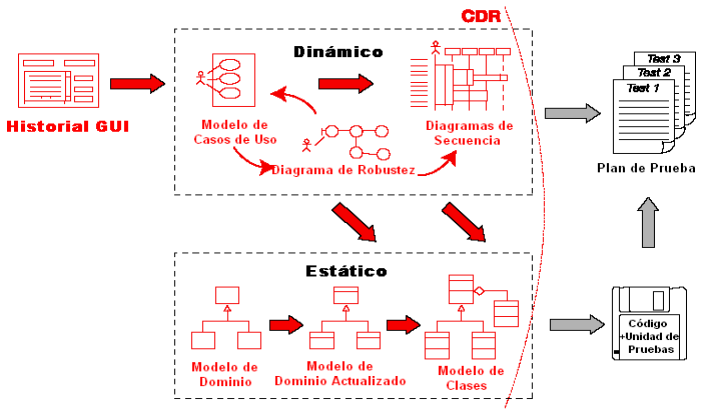

Figura N° 2.8: Revisión del diseño crítico (Rosenberg y Sthephens, 2007)

Porras (2011), en la implementación se tiene; a) Código y pruebas unitarias.- Escribir el código fuente y formular las pruebas unitarias o, escriba las pruebas unitarias y luego el código, b) Integración y pruebas de escenario.- Base las pruebas de integración en los casos de uso, para aprobar los cursos básico y alterno, c) Revisar código y actualizar el modelo.- Luego prepararse para la siguiente iteración del proceso ICONIX, con otro pequeño grupo de casos de uso.

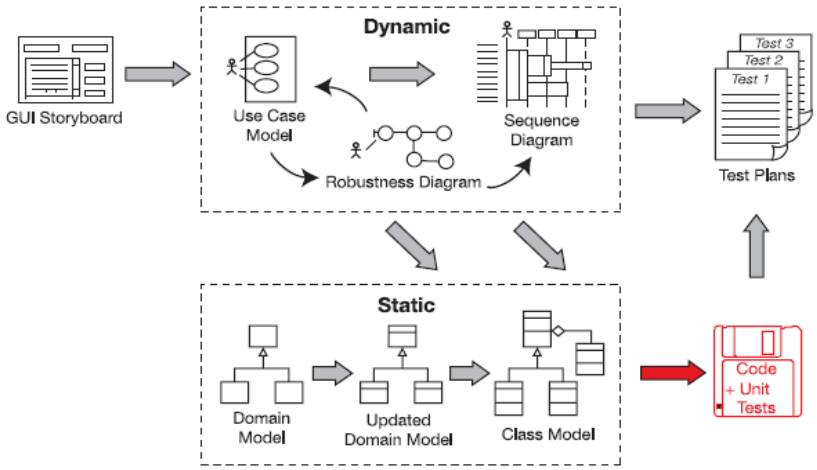

Figura N° 2.9: Implementación (Rosenberg y Sthephens, 2007)

2.2.2. ARQUITECTURA TÉCNICA

La arquitectura técnica debe satisfacer tanto los requisitos como las restricciones de diseño. Se debe indicar qué componentes básicos software, hadware y de comunicación deben adquirirse o desarrollarse (Barranco, 2001).

2.2.3. PATRÓN MVC DEL FRAMEWORK .NET

El marco de ASP.NET MVC es una presentación de poca complejidad y fácil de comprobar que (como las aplicaciones basadas en formularios Web Forms) se integra con las características de ASP.NET existentes, tales como páginas maestras y la autenticación basada en pertenencia. El marco de MVC se define en el ensamblado System.Web.Mvc.

El modelo arquitectónico Modelo-Vista-Controlador (MVC) separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El marco de

ASP.NET MVC proporciona una alternativa al modelo de formularios Web Forms de ASP.NET para crear aplicaciones web (Freeman, 2013).

MVC es un modelo de diseño estándar con el que están familiarizados muchos desarrolladores. Algunos tipos de aplicaciones web salen beneficiados con el marco de MVC. Otras seguirán usando el modelo de la aplicación ASP.NET tradicional que está basado en formularios Web Forms y devoluciones (Naylor, 2009).

El marco de MVC incluye los componentes siguientes:

a. Modelos. Los objetos de modelo son las partes de la aplicación que implementan la lógica del dominio de datos de la aplicación. A menudo, los objetos de modelo recuperan y almacenan el estado del modelo en una base de datos. Por ejemplo, un objeto Product podría recuperar información de una base de datos, trabajar con ella y, a continuación, escribir la información actualizada en una tabla Productos de una base de datos de SQL Server.
b. Vistas. Las vistas son los componentes que muestra la interfaz de usuario de la aplicación. Normalmente, esta interfaz de usuario se crea a partir de los datos de modelo. Un ejemplo sería una vista de edición de una tabla Productos que muestra cuadros de texto, listas desplegables y casillas basándose en el estado actual de un objeto Product.
c. Controladores. Los controladores son los componentes que controlan la interacción del usuario, trabajan con el modelo y por último seleccionan una vista para representar la interfaz de usuario. En una aplicación MVC, la vista solo muestra información; el controlador administra y responde a los datos proporcionados por el usuario y su interacción. Por ejemplo, el controlador administra los valores de la cadena de consulta y pasa estos valores al modelo, que a su vez podría usarlos para consultar la base de datos (Naylor, 2009).

El marco de ASP.NET MVC ofrece la separación de tareas de aplicación (lógica de entrada, lógica de negocios y lógica de la interfaz de usuario), facilidad para pruebas y desarrollo basado en pruebas (TDD). Todos los contratos principales del marco de MVC se basan en interfaz y se pueden probar mediante objetos ficticios, que son objetos simulados que imitan el comportamiento de objetos reales en la aplicación. Puede hacer una prueba unitaria de la aplicación sin tener que ejecutar los controladores

en un proceso de ASP.NET, lo cual hace que las pruebas unitarias sean rápidas y flexibles. Puede usar cualquier marco de pruebas unitarias que sea compatible con .NET Framework (Naylor, 2009).

Los componentes del marco de ASP.NET MVC están diseñados para que se puedan reemplazar o personalizar con facilidad. Puede conectar su propio motor de vista, directiva de enrutamiento de URL, serialización de parámetros de método y acción, y otros componentes. El marco de ASP.NET MVC también admite el uso de los modelos de contenedor Inyección de dependencia (DI) e Inversión de control (IOC). DI permite insertar objetos en una clase, en lugar de depender de que la clase cree el propio objeto. IOC especifica que si un objeto requiere otro objeto, el primer objeto debe obtener el segundo objeto de un origen externo como un archivo de configuración. Esto facilita las pruebas aplicación (Galloway, et. al., 2014).

Amplia compatibilidad para el enrutamiento de ASP.NET, un eficaz componente de asignación de direcciones URL que le permite compilar aplicaciones que tienen direcciones URL comprensibles y que admiten búsquedas. Las direcciones URL no tienen que incluir las extensiones de los nombres de archivo y están diseñadas para admitir patrones de nombres de direcciones URL que funcionan bien para la optimización del motor de búsqueda (SEO) y el direccionamiento de transferencia de estado representacional (REST, Representational State Transfer).

Compatibilidad para usar el marcado en archivos de marcado de páginas de ASP.NET existentes (archivos .aspx), de controles de usuario (archivos .ascx) y de páginas maestras (archivos .master) como plantillas de vista. Puede usar las características de ASP.NET existentes con el marco de ASP.NET MVC, tales como páginas maestras anidadas, expresiones en línea (<%= %>), controles de servidor declarativos, plantillas, enlace de datos, y localización (Naylor, 2009).

Compatibilidad con las características de ASP.NET existentes. ASP.NET MVC le permite usar características, tales como la autenticación de formularios y la autenticación de Windows, la autorización para URL, la pertenencia y los roles, el caching de resultados y datos, la administración de estados de sesión y perfil, el seguimiento de estado, el sistema de configuración y la arquitectura de proveedor.

Muñoz, S. (2018, p. 20), sostiene que:

> Cuando construimos un sitio mediante ASP.NET MVC, separamos el código del lado del servidor en tres partes:
>
> Modelo: Un modelo MVC define un conjunto de clases que representan los tipos de objetos que la aplicación Web administra. Por ejemplo, el modelo de un sitio de comercio electrónico podría incluir una clase modelo Producto que defina propiedades tales como la descripción, número de catálogo, precio y otros. Los Modelos suelen incluir lógica de acceso a datos que lee datos desde una base de datos y escribe datos a esa base de datos. Vistas: Una vista MVC es un componente que construye las páginas Web que conforman la interfaz de usuario de la aplicación Web. Los controladores suelen pasar una instancia de una clase del Modelo a una Vista. La Vista muestra las propiedades de la clase Modelo. Por ejemplo, si el controlador pasa un objeto Producto, la vista puede mostrar el nombre del producto, una imagen y su precio.
>
> Controladores: Un controlador MVC es una clase que se encarga de la interacción del usuario, crea y modifica clases del Modelo y selecciona las Vistas correspondientes. Por ejemplo, cuando un usuario solicita detalles completos acerca de un producto en particular, el controlador crea una nueva instancia de la clase modelo Producto y la pasa a la Vista de detalles para que la muestre al usuario.
>
> Esta separación de código de Modelo, Vista y Controlador garantiza que las aplicaciones MVC tengan una estructura lógica, incluso para los sitios más complejos. También mejora la capacidad de prueba de la aplicación. En última instancia, ASP.NET MVC proporciona un mayor control sobre el código HTML en comparación con Web Pages y Web Forms.

2.2.4. ADO.NET ENTITY FRAMEWORK

ADO.NET Entity Framework admite aplicaciones y servicios centrados en datos, y proporciona una plataforma para la programación con datos que eleva el nivel de abstracción del nivel lógico relacional al nivel conceptual. Al permitir a los desarrolladores trabajar con datos en un nivel de abstracción superior, Entity Framework admite código que es independiente de cualquier motor de almacenamiento de datos o esquema relacional determinados (Naylor, 2009).

Entity Framework, junto con el framework Language-Integrated Query (LINQ), ambos de Microsoft, nos permite abordar el problema de desajuste de frente. Usando Entity Framework, modelamos clases de entidad para nuestra aplicación sobre una superficie de diseño o directamente en código. Luego modelamos relaciones (asociaciones) entre estas entidades. En nuestro código, construimos consultas LINQ para programar contra estas entidades y asociaciones. LINQ nos permite expresar la base de datos relacional establece conceptos directamente en nuestro código mientras trabaja en términos de tipos de entidades y asociaciones. Todo esto ayuda a optimizar nuestra experiencia de desarrollo al tiempo que reduce el esfuerzo general. En lugar de codificar grandes números de construcciones de acceso a datos ADO.NET altamente redundantes, expresamos nuestras necesidades de datos en simples consultas LINQ. En lugar de programar contra el esquema de una base de datos altamente normalizada, codificamos contra las clases de entidad. (Driscoll, Gupta, Vettor, Hirani y Tenny, 2013, p. 1)

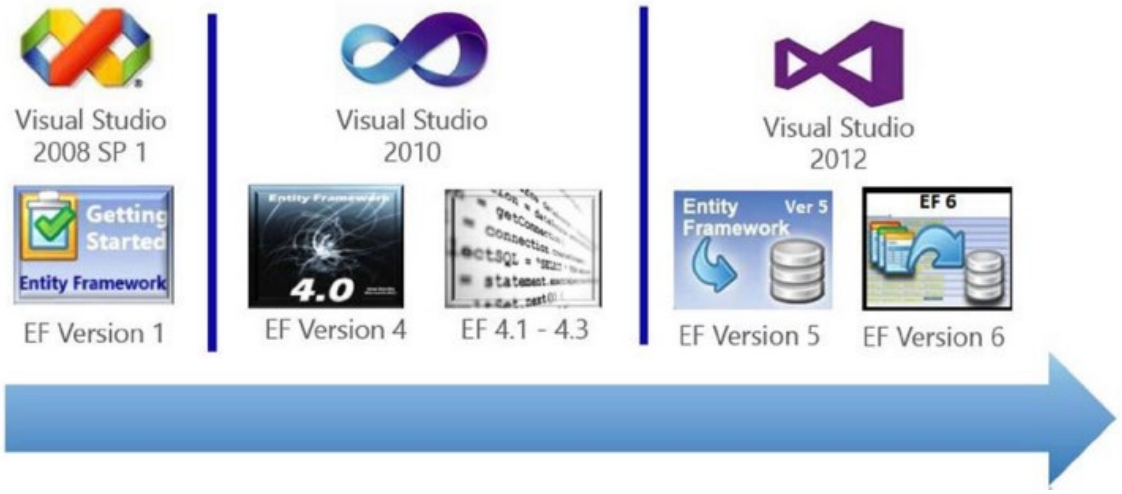

Figura N° 08: Historia de Entity Framework
Fuente: Driscoll, Gupta, Vettor, Hirani y Tenny (2013)

Microsoft (2008) sostiene:

> ADO.NET Entity Framework está diseñado para permitir a los programadores crear aplicaciones de acceso a datos programando con un modelo de la aplicación conceptual en lugar de programar directamente con un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos. Las aplicaciones de Entity Framework ofrecen las siguientes ventajas:

- Las aplicaciones pueden funcionar en términos de un modelo conceptual más centrado en la aplicación, que incluye tipos con herencia, miembros complejos y relaciones.
- Las aplicaciones están libres de dependencias de codificación rígida de un motor de datos o de un esquema de almacenamiento.
- Las asignaciones entre el modelo conceptual y el esquema específico de almacenamiento pueden cambiar sin tener que cambiar el código de la aplicación.
- Los programadores pueden trabajar con un modelo de objeto de aplicación coherente que se puede asignar a diversos esquemas de almacenamiento, posiblemente implementados en sistemas de administración de base de datos diferentes.
- Se pueden asignar varios modelos conceptuales a un único esquema de almacenamiento.

La compatibilidad con Language-Integrated Query proporciona validación de la sintaxis en el momento de la compilación para consultas en un modelo conceptual.

2.2.5. MOTOR DE VISTA RAZOR

"Razor es el motor de vistas creado desde MVC3 que sirve para usar código de VB o C# dentro de las páginas web. Es la evolución de ASPX. Las instrucciones se abren con @{ .. }" (Rincón, 2016, p. 30).

Es un motor de vista introducido por primera vez en MVC 3 y se ha convertido en el motor de vista por defecto desde ese entonces. Provee una sintaxis muy parecida a sistemas script como PHP donde con una simple instrucción se ecribe código de servidor en medio una página html, evitando el uso de controles complicados. Si es C# la misma se nombra como .cshtml, si es VB se nombra .vbhtml (Gonzáles, 2015).

Microsoft (2018) sostiene que "Las vistas de ASP.NET Core MVC usan el motor de vistas de Razor para representar vistas. Razor es un lenguaje de marcado de plantillas compacto, expresivo y fluido para definir vistas que usan código incrustado de C#. Razor se usa para generar dinámicamente contenido web en el servidor. Permite

combinar de manera limpia el código del servidor con el código y el contenido del lado cliente."

2.2.6. CLASES CONTROLADORES C#

Según Anderson (2017), los controladores son "Las clases que controlan las solicitudes del explorador. Recuperan los datos del modelo y llaman a plantillas de vistas que devuelven una respuesta. En una aplicación MVC, la vista solo muestra información; el controlador controla la interacción de los usuarios y los datos que introducen, y responde a ellos. Por ejemplo, el controlador controla los datos de enrutamiento y los valores de cadena de consulta y pasa estos valores al modelo. El modelo puede usar estos valores para consultar la base de datos"

En ASP.NET MVC Framework, los controladores son clases .NET que contienen la lógica necesaria para manejar una solicitud. En el Capítulo 3, expliqué que la función del controlador es encapsular la lógica de la aplicación. Esto significa que los controladores son responsables de procesar las solicitudes entrantes, realizar operaciones en el modelo de dominio y seleccionando vistas para renderizar al usuario (Freeman, 2013, p. 451).

CAPITULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. TIPO DE INVESTIGACIÓN

Se determinó que el "Patrón MVC del Framework .Net es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX".

La **investigación descriptiva** es un tipo de metodología a aplicar para deducir un bien o circunstancia que se esté presentando; se aplica describiendo todas sus dimensiones, en este caso se describe el órgano u objeto a estudiar, nuestra investigación se centra en lo descriptivo, porque estrudiamos la arquitectura técnica de la metodología agil y formal ICONIX.

Según Jiménez (1998), la investigación es **aplicada "**si el problema surge directamente de la práctica social y tienen aplicación en el ámbito donde se realizan". Es obvio, que la aplicación no tiene forzosamente que ser directa en la producción o en los servicios, pero sus resultados se consideran de utilidad para aplicaciones prácticas.

3.2. POBLACIÓN Y MUESTRA

A. POBLACIÓN

La población esta compuesta por las capas de la arquitectura técnica de la metodología ágil y formal ICONIX.

B. MUESTRA

Compuesta por las capas de presentación, lógica de negocio y acceso a datos de la arquitectura técnica de la metodología ágil y formal ICONIX

3.3. VARIABLES E INDICADORES

DEFINICIÓN CONCEPTUAL DE LAS VARIABLES

VARIABLE DESCRIPTIVA

Patrón MVC del Framework .Net.- El modelo arquitectónico Modelo-Vista-Controlador (MVC) separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El marco de ASP.NET MVC proporciona una alternativa al modelo de formularios Web Forms de ASP.NET para crear aplicaciones web.

VARIABLE DE INTERES

Arquitectura técnica de la Metodología Ágil y Formal ICONIX.- La arquitectura técnica es aquella que debe satisfacer tanto los requisitos como las restricciones de diseño. Se debe indicar qué componentes básicos software, hadware y de comunicación deben adquirirse o desarrollarse.

DEFINICIÓN OPERACIONAL DE LAS VARIABLES

VARIABLE DESCRIPTIVA

X: Patrón MVC del Framework .Net

INDICADORES

- X1: ADO.NET Entity Framework
- X2: Motor de Vista Razor
- X3: Clases Controladores C#

VARIABLE DE INTERES

Y: Arquitectura técnica de la metodología ágil y formal ICONIX

INDICADORES:

Y1: Arquitectura Técnica

3.4. HERRAMIENTAS PARA EL TRATAMIENTO DE DATOS E INFORMACIÓN

Las herramientas tecnológicas que se utilizan son seleccionadas tomando en cuenta la facilidad y rapidez en su desarrollo, que facilite administrar con facilidad la complejidad de la aplicación, desarrollo de aplicaciones aplicando estándares y nos permita utilizar técnicas para asegurar la información crítica. Se ha seleccionado las tecnologías según la tabla 3.1.

SOFTWARE	FABRICANTE	SERVICIO
WINDOWS 10 HOME	Microsoft Corporation	Es la versión del sistema operativo Windows para escritorio, ofrece el marco para instalar las herramientas de desarrollo.
VISUAL STUDIO 2017	Microsoft Corporation	Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma.
SQL Server 2014 Management Studio	Microsoft Corporation	Microsoft SQL Server es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft. El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).
C#	Microsoft Corporation	C# es el nuevo lenguaje de propósito general orientado a objetos creado por Microsoft para su nueva plataforma .NET.. Puede usar C# para crear aplicaciones cliente de Windows, servicios web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos y muchas, muchas más cosas. Visual C# proporciona un editor de código avanzado, prácticos diseñadores de interfaz de usuario, un depurador integrado y muchas otras herramientas que facilitan el desarrollo de aplicaciones basadas en el lenguaje C# y .NET Framework.
RAZOR	Microsoft	En una sintaxis basada en C# que

	Corporation	permite usarse como motor de programación en las vistas o plantillas de nuestros controladores.
BOOTSTRAP	Open Source	Es un framework web o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.
JAVASCRIPT	Netscape	Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript podemos crear diferentes efectos e interactuar con nuestros usuarios.

Tabla N° 3.1: Herramientas tecnológicas para el tratamiento de datos.

3.5. TÉCNICAS PARA APLICAR ICONIX

Revisando el marco teórico desarrollado en el capítulo II, sección 2.2.4, formulamos el proceso, que considera las fases para desarrollar la aplicación web usando la metodología Iconix, como se muestra en las tablas 3.2 a 3.9.

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Identificar requisitos	• Requisitos funcionales y no funcionales.	• Entrevista. • Definir lo que el sistema debe hacer.	Usuario Cliente Analista
	• Casos de prueba de aceptación.	• Escribir al menos un caso de prueba para cada requisito.	

Identificar objetos del mundo real y dibujar modelo de dominio	Modelo de dominio	• Identificar clases clave del negocio. • Identificar sustantivos y depurar. • Identificar objetos en requisitos funcionales y asignar al modelo de dominio. • Utilizar agregación y generalización.	Analista
Asignar casos de uso a los requisitos funcionales	Relación entre requisitos funcionales y casos de uso	• Asignar casos de uso a los requisitos funcionales.	
Descubrir los casos de uso	Lista de casos de uso	• Utilizar requisitos funcionales. • Entrevistas.	Usuario Cliente Analista
Dibujar y empaquetar casos de uso	• Paquete de casos de uso. • Diagrama de casos de uso.	• Identificar roles y responsabilidades de actores. • Asociar actores con casos de uso. • Relacionar casos de uso • Agrupar lógicamente casos de uso.	Analista
Realizar prototipo de interfaz grafica	Prototipo GUI	• Utilizar historia de eventos del usuario (storyboards). • Utilizar los requisitos funcionales. • Diseñar interfaz gráfica básica.	Programador Analista
Escribir el primer borrador de casos de uso	Primer borrador de casos de uso	• Utilizar glosario de objetos del modelo de dominio. • Utilizar la regla de dos párrafos. • Escribir el caso de uso como flujos de evento/respuesta. • Escribir el caso de uso con	Analista

		estructura sustantivo-verbo-sustantivo. • Escribir caso de uso en voz activa. • Referenciar por su nombre las pantallas.	

Tabla N° 3.2: Análisis de requisitos (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar el modelo de dominio	Modelo de dominio revisado	• Identificar al menos 80% de clases clave de dominio del problema.	Usuario Cliente Analista Programador
Revisar el prototipo GUI	Prototipo GUI revisado	• Diseñar con precisión la GUI relacionada al caso de uso.	
Revisar la descripción de casos de uso	Casos de uso revisado	• Eliminar clases fuera del dominio del problema. • Cambiar descripción de voz pasiva activa. • Describir todos los cursos alternos. • Asociar todos los requisitos a los casos de uso. • Describir que intentar hacer el usuario para cada caso de uso.	

Tabla N° 3.3: Revisión de requisitos (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar descripción de casos de uso	Casos de uso	• Coincidir la descripción del caso de uso con el diagrama de robustez.	Usuario Cliente Analista Programador
Revisar diagrama de robustez	Diagrama de robustez	• Coincidir el diagrama de robustez con la descripción del caso de uso. • Verificar que el diagrama de robustez cumple las reglas. • Verificar que el diagrama de robustez tiene todos los cursos alternos.	

Revisar modelo de dominio actualizado	Modelo de dominio actualizado	• Coincidir los objetos de entidad del diagrama de robustez con el modelo de dominio actualizado.	

Tabla N° 3.4: Revisión de diseño preliminar (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar descripción de casos de uso	Casos de uso	• Coincidir la descripción del caso de uso con el diagrama de robustez.	Usuario Cliente Analista Programador
Revisar diagrama de robustez	Diagrama de robustez	• Coincidir el diagrama de robustez con la descripción del caso de uso. • Verificar que el diagrama de robustez cumple las reglas. • Verificar que el diagrama de robustez tiene todos los cursos alternos.	
Revisar modelo de dominio actualizado	Modelo de dominio actualizado	• Coincidir los objetos de entidad del diagrama de robustez con el modelo de dominio actualizado.	

Tabla N° 3.5: Revisión de diseño preliminar (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Diseñar diagrama de componentes	Diagrama de componentes	• Entrevistas. • Características del negocio. • Utilizar la arquitectura MVC. • Integrar frameworks.	Cliente Analista Programador Arquitecto de software
Diseñar diagrama de despliegue	Diagrama de despliegue	• Entrevistas. • Características del negocio. • Utilizar modelo cliente servidor.	Cliente Analista Programador Arquitecto de software

Tabla N° 3.6: Arquitectura técnica (Porras, 2011)

TAREA	ARTEFACTO	TECNICA	RESPONSABLE
Dibujar un diagrama de secuencia para cada caso de uso	Diagrama de secuencia	• Copiar la descripción del caso de uso. • Copiar objetos entidad, interfaz y actores del diagrama de robustez. • Verificar que un mensaje del diagrama de secuencia es verbo en el caso de uso. • Hacer refactoring al diagrama de secuencia antes de codificar.	Programador Diseñador
Actualizar el diagrama de clases de un caso de uso	Diagrama de clase	• Asignar operaciones a las clases a partir de mensajes del diagrama secuencia. • Establecer multiplicidad en clases. • Depurar las clases, operaciones y atributos del diagrama de clases.	
Extraer controladores para pruebas unitarias	Lista de controladores	• Identificar controladores para la lógica del negocio desde un diagrama de robustez.	

Tabla N° 3.7: Diseño detallado (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLE
Revisar diagrama de secuencia	Diagrama de secuencia	• Verificar que el diagrama de secuencia coincide con la descripción del caso de uso. • Verificar que el diagrama de secuencia representa los cursos básico y alterno. • Verificar en los mensajes que	Diseñador Programador

		los atributos y valores de retorno son correctos.	
Revisar diagrama de clases	Diagrama de clases	• Verificar que el nombre, atributos y operaciones se asignaron correctamente a las clases. • Asignar requisitos no funcionales a los casos de uso y clases.	
Revisar lista de pruebas unitarias	Lista de controladores	• Actualizar la lista de controladores.	

Tabla N° 3.8: Revisión crítica de diseño (Porras, 2011)

TAREA	ARTEFACTO	TÉCNICA	RESPONSABLES
Implementar la base de datos física	Base de datos física	• Escribir el script usando el modelo de dominio actualizado • Ejecutar el script usando un DBMS	Programador
Implementar código para clases entidad	Código fuente	• Escribir o generar código fuente con una herramienta usando el modelo de dominio actualizado	Programador
Implementar código para las interfaces	Código fuente	• Generar código fuente usando una herramienta	Programador
Crear pruebas unitarias para cada controlador	Prueba unitaria	• Escribir código fuente para una prueba unitaria usando una herramienta	Programador
Implementar código fuente para cada controlador	Código fuente	• Escribir el código fuente siguiendo el flujo normal del diagrama de secuencia • Actualizar el diagrama de secuencia con la codificación	Programador
Ejecutar pruebas unitarias para cada controlador	Reporte de pruebas unitarias	• Ejecutar el módulo de cada prueba unitaria. • Modificar código fuente si la prueba unitaria es incorrecto.	Programador

Ejecutar pruebas de aceptación para cada caso de uso	Reporte de pruebas de aceptación	• Utilizar los casos de prueba de aceptación • Ejecutar el módulo de un caso de uso • Modificar código fuente si la prueba de aceptación muestra resultado incorrecto	Programador Usuario Cliente

Tabla N° 3.9: Implementación (Porras, 2011)

CAPITULO IV

ANÁLISIS Y RESULTADOS DE LA INVESTIGACIÓN

4.1. ARTEFACTOS DE SOFTWARE APLICANDO ICONIX

Según las fases para desarrollar la aplicación web usando la metodología Iconix, desarrollado en las tablas 3.2 a 3.9, descrito en el capítulo II, sección 2.2.4, se obtienen los artefactos análisis de requisitos, revisión de requisitos, diseño preliminar, revisión de diseño preliminar, diseño detallado, revisión crítica de diseño e implementación. A continuación mostramos los resultados de los artefactos de acuerdo a la metodología Iconix.

4.1.1 ANÁLISIS DE REQUISITOS

REQUISITOS FUNCIONALES

Nº REQ.	REQUISITOS FUNCIONALES
1	El sistema debe permitir a que el **usuario** acceda al servicio del software a través de una **cuenta de usuario** que será asignada previamente por el **administrador**.
2	El sistema debe permitir mantener(Registrar, modificar y eliminar) datos del **Estudio Ambiental,** ubigeo
3	El **Registrador** debe permitir mantener (Registrar, modificar y eliminar) datos del **Titular**, Dirección
4	El Registrador debe permitir Mantener (Registrar, modificar y eliminar) datos de la **Resolución**.
5	El Registrador debe permitir Mantener (Registrar, modificar y eliminar) datos de la **consultor**.
6	El sistema debe permitir emitir reporte de **Estudio Ambiental**

7	El Registrador realiza **Búsqueda** de los Estudios ambientales de acuerdo a su interés en la base de datos Registrado**.**
8	El Registrador podrá realizar Búsquedas de los Estudios ambientales de acuerdo al **tipo (minería, electricidad e hidrocarburos)** y **estado (Aprobado, desaprobado, abandono).**
9	El sistema debe permitir mostrar el reporte de estado y tipo del Estudio Ambiental, **fecha**, de ingreso
10	El sistema debe permitir mantener **Ubicación**.
11	El sistema debe permitir visualizar las **Ubicación**.
12	El sistema debe ser capaz de mostrar en la página principal la **ubicación y registro** más recientes.
13	El sistema debe permitir registrar la distribución de los estudios Ambientales en los **estantes**.
14	El sistema debe permitir mantener la distribución de los estudios Ambientales en los **estantes**.
15	El sistema debe permitir visualizar la distribución de los estudios Ambientales en los estantes.
16	El software deberá permitir **mostrar** un listado de los materiales adicionales que contenga el estudio Ambiental (Estudio, Pioneer, fólderes, Anillados, Cds, Planos).
17	El administrador debe ser capaz de crear una cuenta para un usuario, previa **solicitud de acceso**.
18	La aplicación web debe ser administrada dinámicamente por un **administrador** de **cuentas de usuario** previa solicitud de usuario.

Tabla N° 4.1: Requisitos funcionales

REQUISITOS NO FUNCIONALES

Nº REQ.	REQUISITOS NO FUNCIONALES
1	El software debe ejecutarse en cualquier sistema operativo garantizando su portabilidad.
2	El software debe presentar interfaces graficas fáciles de utilizar.
3	El software debe ser personalizable para garantizar el cumplimiento del rol de un usuario.
4	El software debe presentar una arquitectura y codificación usando estándares que permita su operación y mantenimiento adecuado.

Tabla N° 4.2: Requisitos no funcionales

GLOSARIO DE TÉRMINOS

a. Administrador del sistema.
b. Estudio Ambiental. (Aprobado, desaprobado, abandono).
c. Titular.
d. tipo.
e. Dirección
f. Consultor.
g. Estante.
h. Registrador.
i. Fecha.
j. Reporte.
k. Ubigeo.
l. Localidad (Departamento, Provincia, Distrito).
m. Usuario.
n. Usuario Anónimo.
o. Cuenta de Usuario.
p. Resolución.
q. Solicitud de Acceso.
r. Estado.

MODELO DE DOMINIO

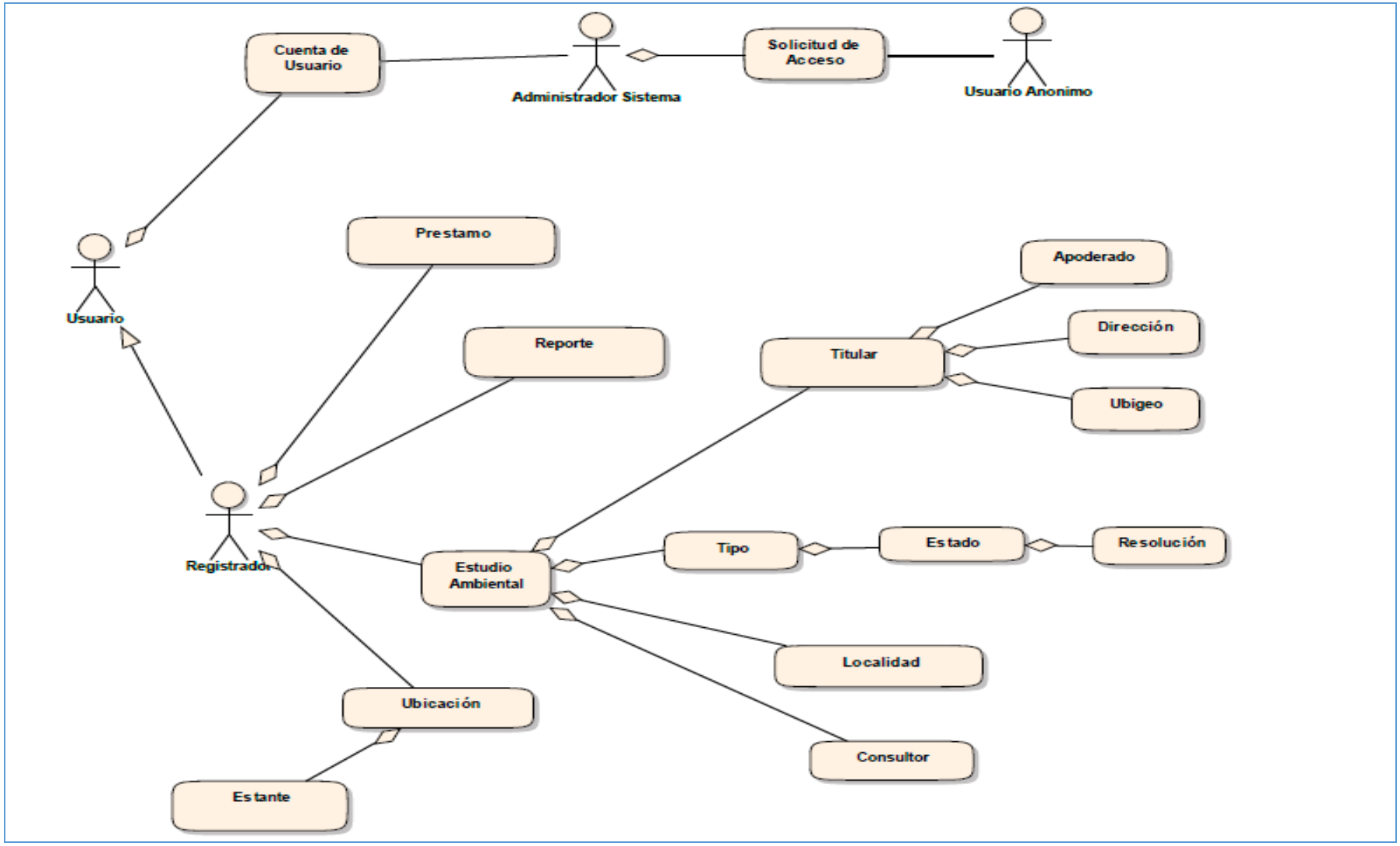

Figura N° 4.1: Modelo de dominio inicial

RELACION ENTRE REQUISITOS FUNCIONALES Y CASOS DE USO

REQUISITOS FUNCIONALES		CASOS DE USO
R1	El sistema debe permitir a que el usuario acceda al servicio del software a través de una cuenta de usuario que será asignada previamente por el administrador.	CU1: Mantener cuenta de usuario. CU2: Autentificar usuario.
R2	El sistema debe permitir mantener(Registrar, modificar y eliminar) datos del Estudio Ambiental, ubigeo	CU3: Mantener Estudio Ambiental.
R3	El Registrador debe permitir mantener (Registrar, modificar y eliminar) datos del Titular, Dirección	CU4: Mantener Titular
R4	El Registrador debe permitir Mantener (Registrar, modificar y eliminar) datos de la Resolución.	CU5: Mantener Resolución
R5	El Registrador debe permitir Mantener (Registrar, modificar y eliminar) datos de la consultor.	CU6: Mantener Consultor
R6	El sistema debe permitir emitir reporte de Estudio Ambiental	CU7: Realizar reporte de Estudio Ambiental
R7	El Registrador realiza Búsqueda de los Estudios ambientales en la base de datos Registrado.	CU8: Realizar Búsqueda de Estudio Ambiental. CU9: Obtener el listado de Estudios Ambientales
R8	El Registrador podrá realizar Búsquedas de los Estudios ambientales de acuerdo al tipo (minería, electricidad e hidrocarburos) y estado (Aprobado, desaprobado, abandono).	CU10: Realizar Búsqueda de Estudio Ambiental por tipo y estado. CU11: Obtener código y datos de Estudio Ambiental
R9	El sistema debe permitir mostrar el reporte de estado y tipo del Estudio Ambiental, fecha, de	CU12: mostrar reporte estudio Ambiental.

	ingreso	
R10	El sistema debe permitir mantener Ubicación.	CU13: Mantener Ubicación
R11	El sistema debe permitir visualizar las Ubicación.	CU14: Visualizar Ubicación.
R12	El sistema debe ser capaz de mostrar en la página principal la ubicación y el registro más recientes.	CU15: Mostrar Ubicación reciente. CU16: Mostrar Registro Reciente.
R13	El sistema debe permitir registrar la distribución de los estudios Ambientales en los estantes.	CU17: Mantener estante CU18: Asignar Ubicación en el Estante
R14	El sistema debe permitir mantener la distribución de los estudios Ambientales en los estantes.	CU19:Mantener disponibilidad Estante
R15	El sistema debe permitir visualizar la distribución de los estudios Ambientales en los estantes.	CU20: Mostrar Distribución de Estudio Ambiental
R16	El software deberá permitir mostrar un listado de los materiales adicionales que contenga el estudio Ambiental (Estudio, Pioneer, fólderes, Anillados, Cds, Planos).	CU21: Mostrar Listado Adicional
R17	El administrador debe ser capaz de crear una cuenta para un usuario, previa solicitud de acceso.	CU22: Crear Cuenta de Usuario CU23: Asignar Roles a Cuenta de Usuario
R18	La aplicación web debe ser administrada dinámicamente por un administrador de cuentas de usuario previa solicitud de usuario.	CU24: Mantener Cuentas de Usuario

Tabla Nº 4.3: Relación entre requisitos funcionales y casos de uso

LISTA DE CASOS DE USO

Nº	CASOS DE USO
CU1	Mantener cuenta de usuario.
CU2	Autentificar usuario.
CU3	Mantener estudio ambiental.
CU4	Mantener titular.
CU5	Mantener resolución.
CU6	Mantener consultor.
CU7	Realizar reporte de estudio ambiental.
CU8	Realizar búsqueda de estudio ambiental.
CU9	Obtener el listado de estudios ambientales.
CU10	Realizar búsqueda de estudio ambiental por tipo y estado.
CU11	Obtener código y datos de estudio ambiental.
CU12	Mostrar reporte estudio ambiental.
CU13	Mantener ubicación.
CU14	Visualizar ubicación.
CU15	Mostrar ubicación reciente.
CU16	Mostrar registro reciente.
CU17	Mantener estante.
CU18	Asignar ubicación en el estante.
CU19	Mantener disponibilidad estante.
CU20	Mostrar distribución de estudio ambiental.
CU21	Mostrar listado adicional.
CU22	Crear cuenta de usuario.
CU23	Asignar roles a Cuenta de Usuario.
CU24	Mantener cuentas de usuario.

Tabla N° 4.4: Lista de casos de uso

PAQUETES DE CASOS DE USO

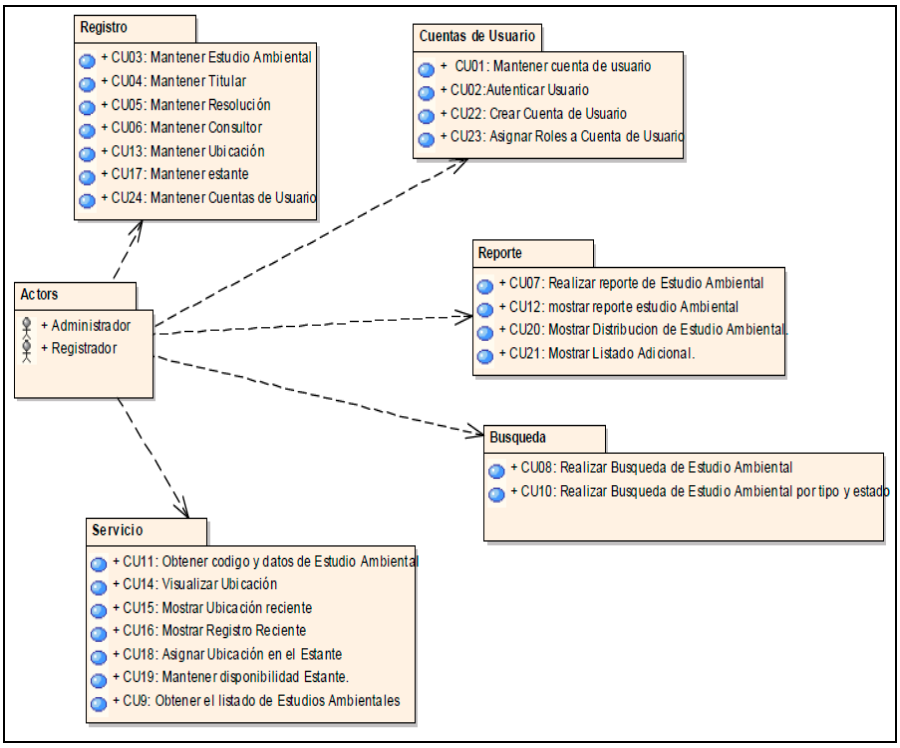

Figura N° 4.2: Paquete de casos de uso

DIAGRAMA DE CASOS DE USO

Cuentas de Usuario

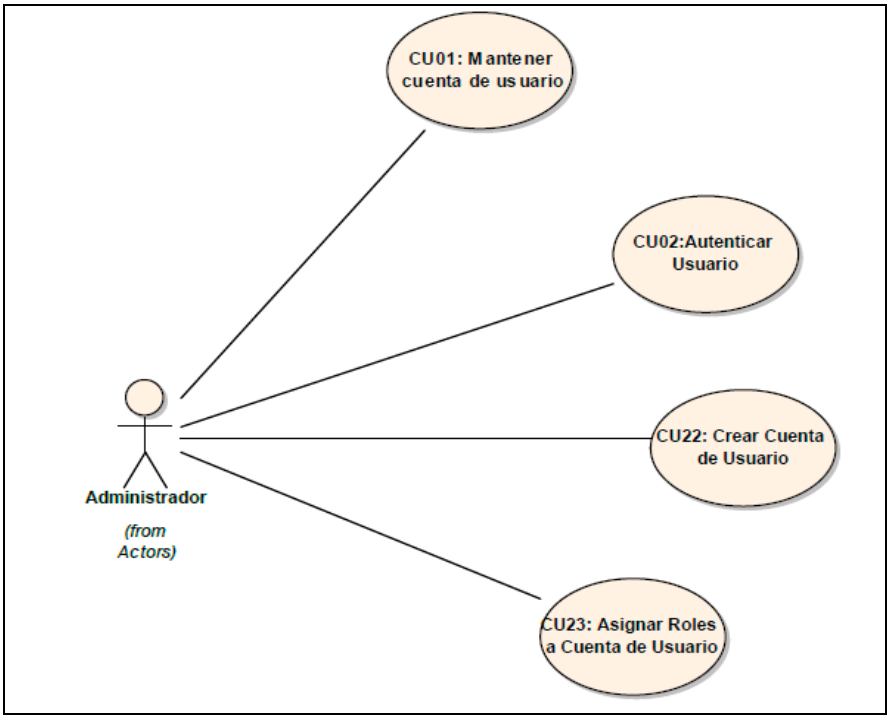

Figura N° 4.3: Paquete “Cuentas de usuario”

Registro

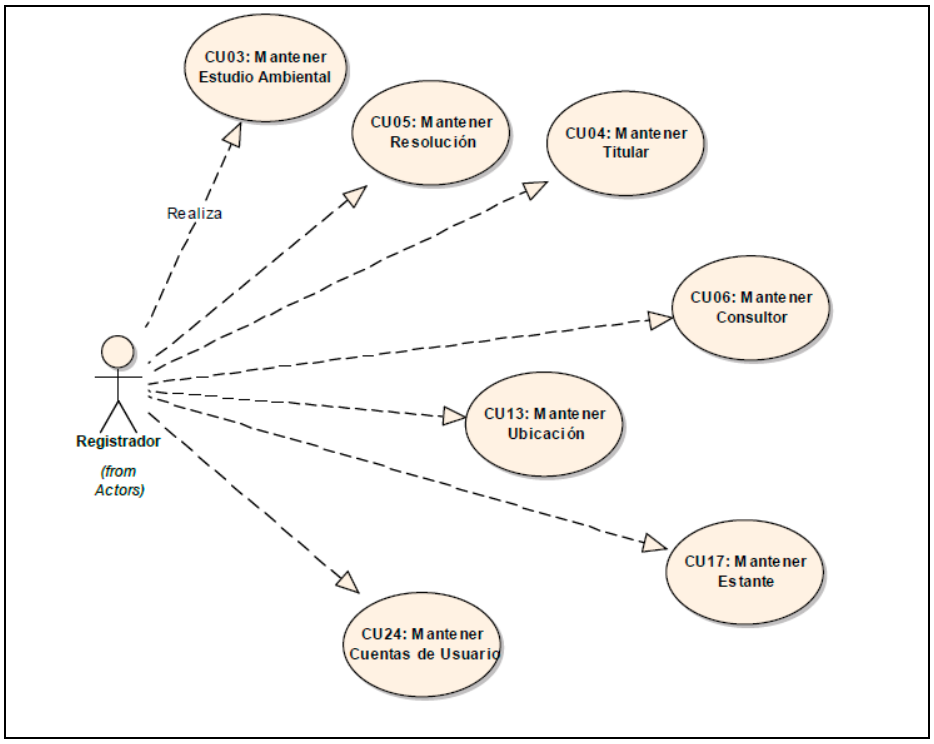

Figura N° 4.4: Paquete “Registro”

Busqueda

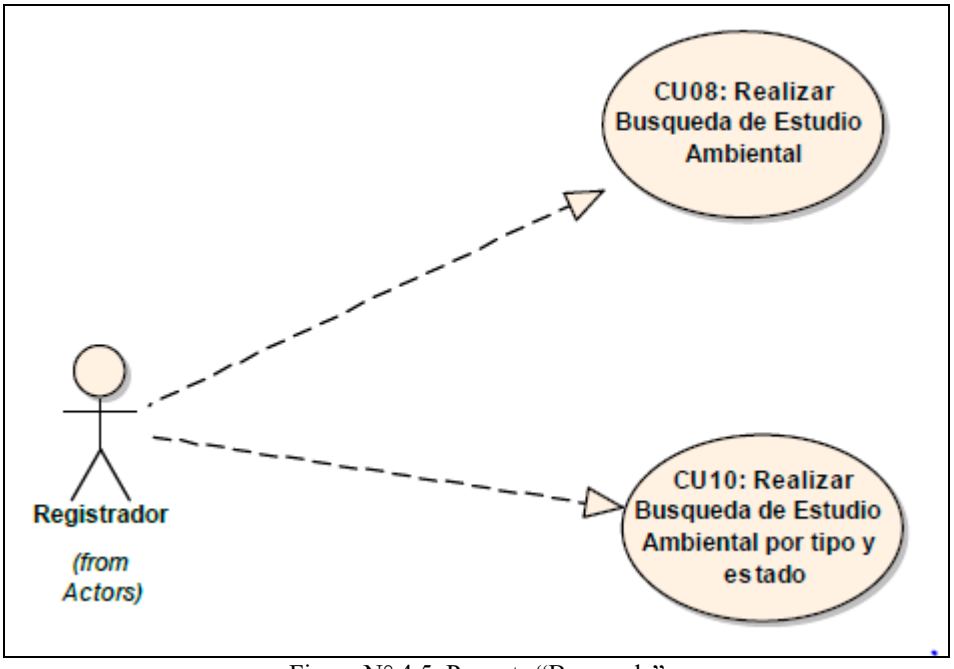

Figura N° 4.5: Paquete “Busqueda”

Servicios

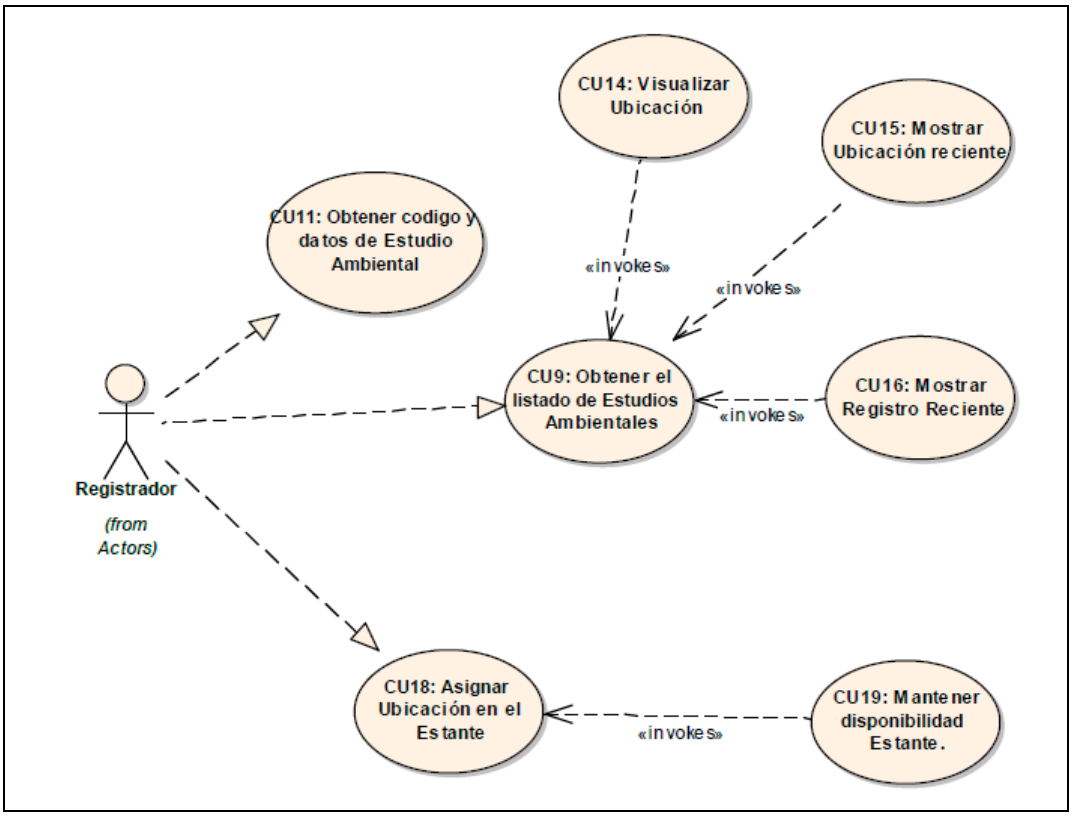

Figura N° 4.6: Paquete “Servicios”

Reporte

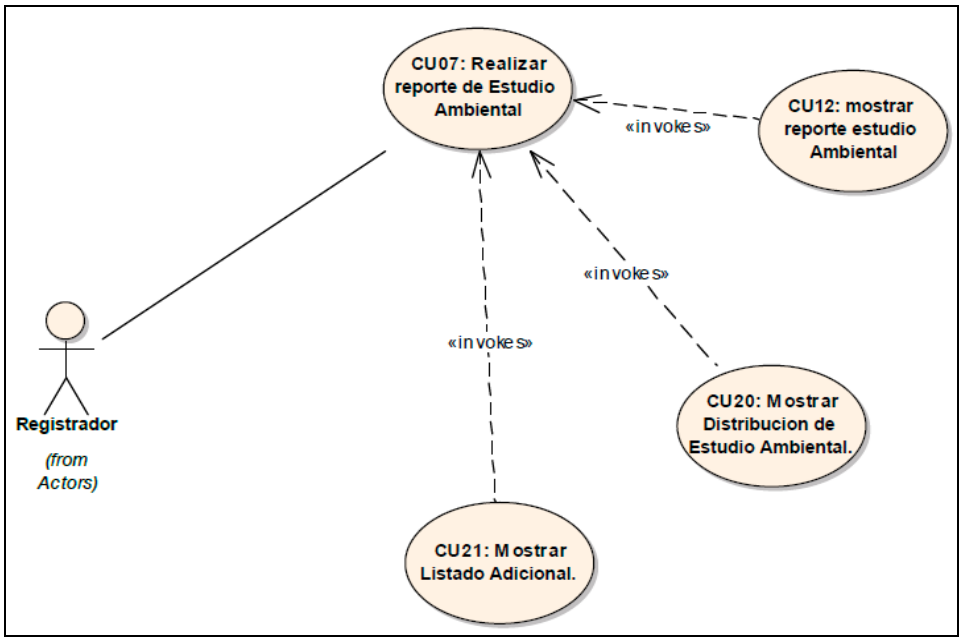

Figura N° 4.7: Paquete “Reporte”

PROTOTIPO DE INTERFAZ

CU 03. Mantener estudio ambiental

Figura N° 4.8: Prototipo de interfaz gráfica “Mantener estudio ambiental”

CU 05. Mantener resolución

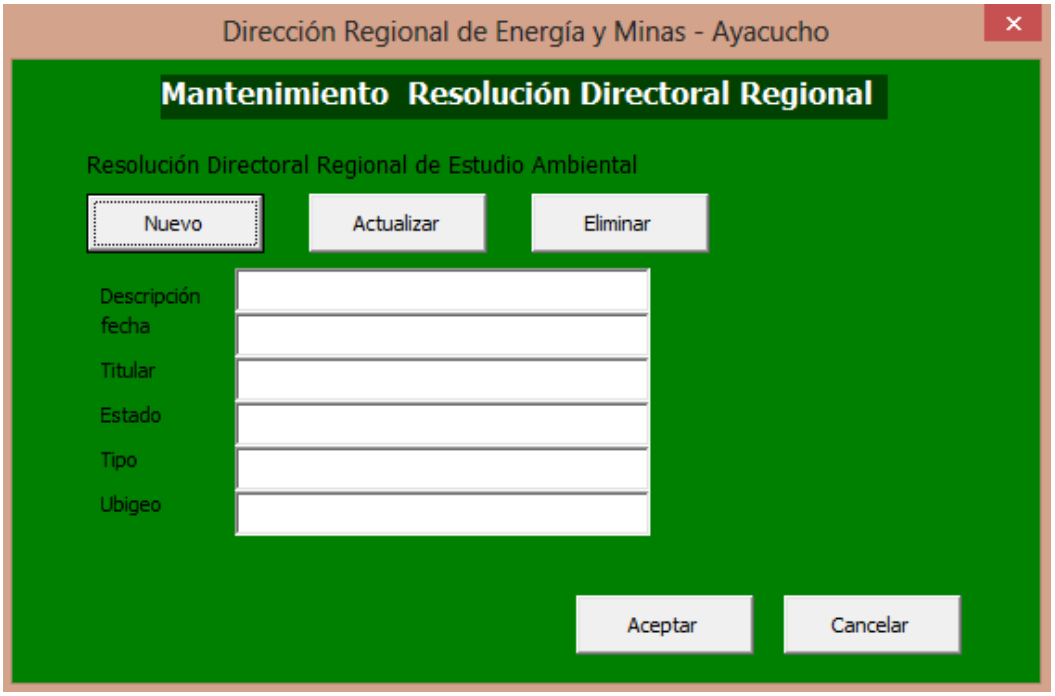

Figura N° 4.9: Prototipo de interfaz gráfica “Mantener resolución”

CU 09. Obtener el listado de estudios ambientales

Figura N° 4.10: Prototipo de interfaz gráfica “Obtener listado de estudio ambiental”

CU 16. Mostrar registro reciente

Figura N° 4.11: Prototipo de interfaz gráfica “Mostrar registro reciente”

PRIMER BORRADOR DE CASOS DE USO

Caso de uso	Descripción
CU03. Mantener estudio ambiental	**Curso Básico:** El Registrador se encuentra en el Menú Registro, hace clic en la opción Estudio Ambiental, el sistema muestra la Pantalla "Mantener Estudio Ambiental". El **Registrador**, para registrar un Estudio Ambiental hace clic en nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos, el Registrador llena los datos; resolución, estudio, titular, Tipo, condición, ubigeo y fecha, hace clic en el botón guardar el sistema guarda los datos del Estudio Ambiental. El **Registrador**, hace clic en buscar para modificar los datos el sistema muestra en la pantalla "estudios ambientales", el **Registrador**, hace clic en el estudio que desea modificar, luego presiona la tecla enter, el sistema carga los datos del Estudio Ambiental en la pantalla “Mantener Estudio Ambiental”, el Registrador modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El **Registrador** hace clic en buscar para eliminar el Estudio Ambiental, el Sistema muestra la pantalla “Estudios Ambientales”, el bibliotecario elige el Estudio Ambiental y realiza la búsqueda del Estudio Ambiental que se desea eliminar, luego presiona la tecla Suprimir; el Sistema elimina el Estudio Ambiental. Curso Alterno: El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles. El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar. El **Registrador** ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Tabla N° 4.5: Primer borrador de caso de uso “Mantener estudio ambiental”

Caso de uso	Descripción
CU05. Mantener resolución	**Curso Básico:** El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Resolución, el sistema muestra la Pantalla "Mantener Resolución Estudio Ambiental". El **Registrador**, para registrar una Resolución Estudio Ambiental hace clic en nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos, el **Registrador** llena los datos; descripción, Nombre, Tipo, Concesión, ubigeo y fecha. Hace clic en el botón guardar el sistema guarda los datos de la Resolución. El **Registrador,** hace clic en buscar para modificar los datos el sistema muestra en la pantalla “Resolución de Estudios Ambientales", el **Registrador**, hace clic en la resolución que desea modificar, luego presiona la tecla enter, el sistema carga los datos de la Resolución Ambiental en la pantalla “Mantener Resolución Ambiental”, el **Registrador** modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El **Registrador** hace clic en buscar para eliminar la Resolución Ambiental, el software muestra la pantalla “Resolución Ambiental”, el bibliotecario elige la Resolución Ambiental y realiza la búsqueda de la Resolución Ambiental que se desea Eliminar, luego presiona la tecla Suprimir; el Sistema elimina la Resolución Ambiental. Curso Alterno: El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles. El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar. El **Registrador** ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Tabla N° 4.6: Primer borrador de caso de uso “Mantener resolución”

Caso de uso	Descripción
CU09. Obtener listado de estudio ambiental	**Curso Básico:** El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el software muestra la pantalla “Estudios Ambientales”. El **Registrador**, hace clic en el botón Listado de estudios Ambientales; el sistema muestra la Pantalla "Listado de Estudios Ambientales", en la cual muestra un listado de los Estudios Ambientales registrados, en la misma pantalla el sistema realiza un resumen de la cantidad de estudios, la cantidad por tipo, y por estado. Curso Alterno: El **Registrador** hace clic en Volver, el sistema Muestra la pantalla anterior y limpia los controles.

Tabla N° 4.7: Primer borrador de caso de uso “Obtener listado de estudio ambiental”

Caso de uso	Descripción
CU16. Mostrar registro reciente	**Curso Básico:** El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el Sistema muestra la pantalla “Estudios Ambientales”. El **Registrador**, hace clic en el botón último Registro; el sistema muestra en pantalla “Ultimo Ingreso Estudios Ambientales registrados ", y los datos, código, nombre, resolución, titular, Estado, Tipo, Ubicación, fecha de registro. Curso Alterno: El **Registrador** hace clic en Volver, muestra la pantalla “Estudios Ambientales”.

Tabla N° 4.8: Primer borrador de caso de uso “Mostrar registro reciente”

4.1.2. REVISIÓN DE REQUISITOS

MODELO DE DOMINIO REVISADO

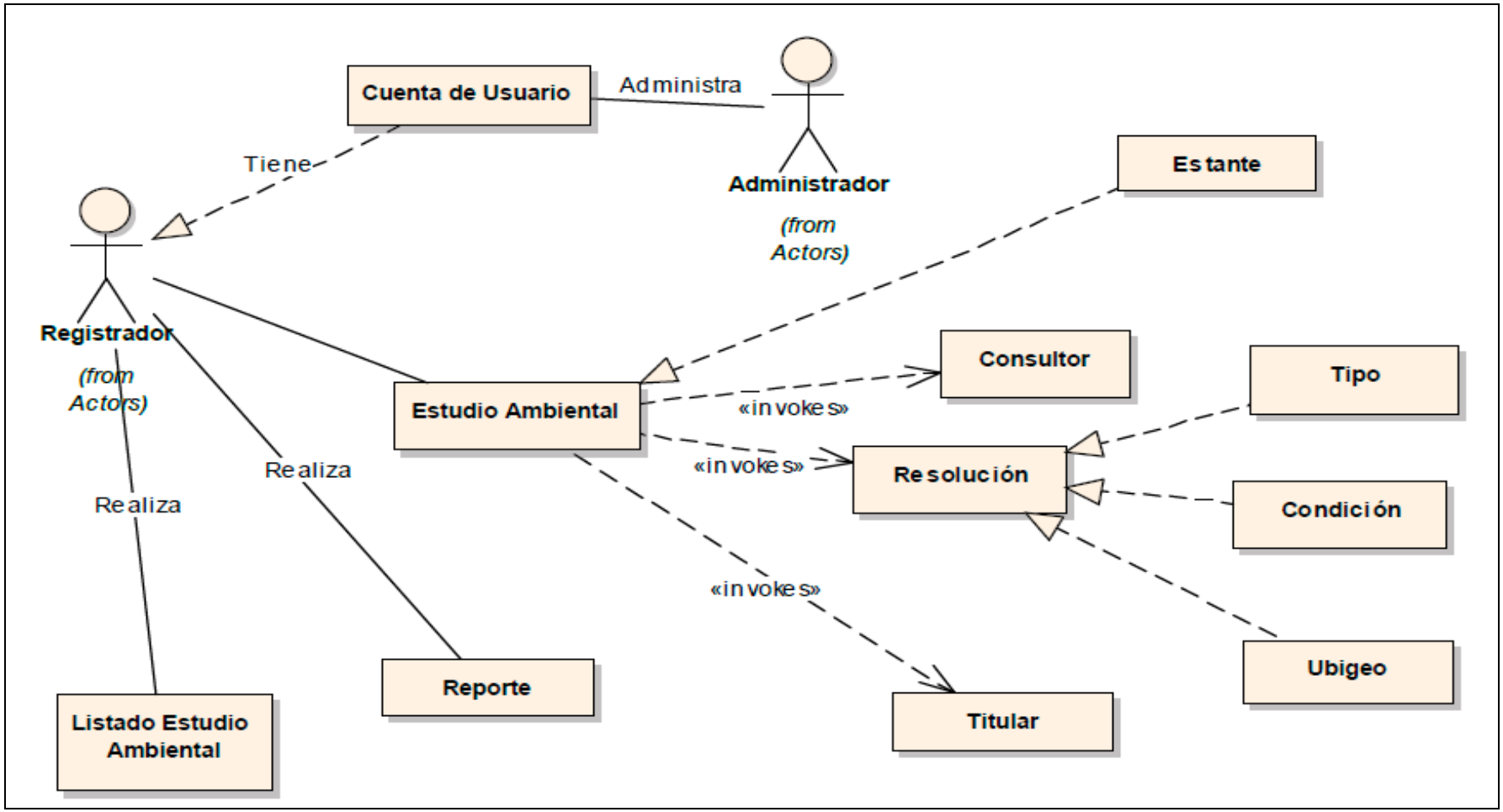

Figura N° 4.12: Modelo de dominio revisado

PROTOTIPO DE GUI REVISADO

Figura N° 4.13: Prototipo de interfaz gráfica revisada "Pantalla inicial de la Dirección Regional de Energía y Minas."

CU 03. Mantener estudio ambiental

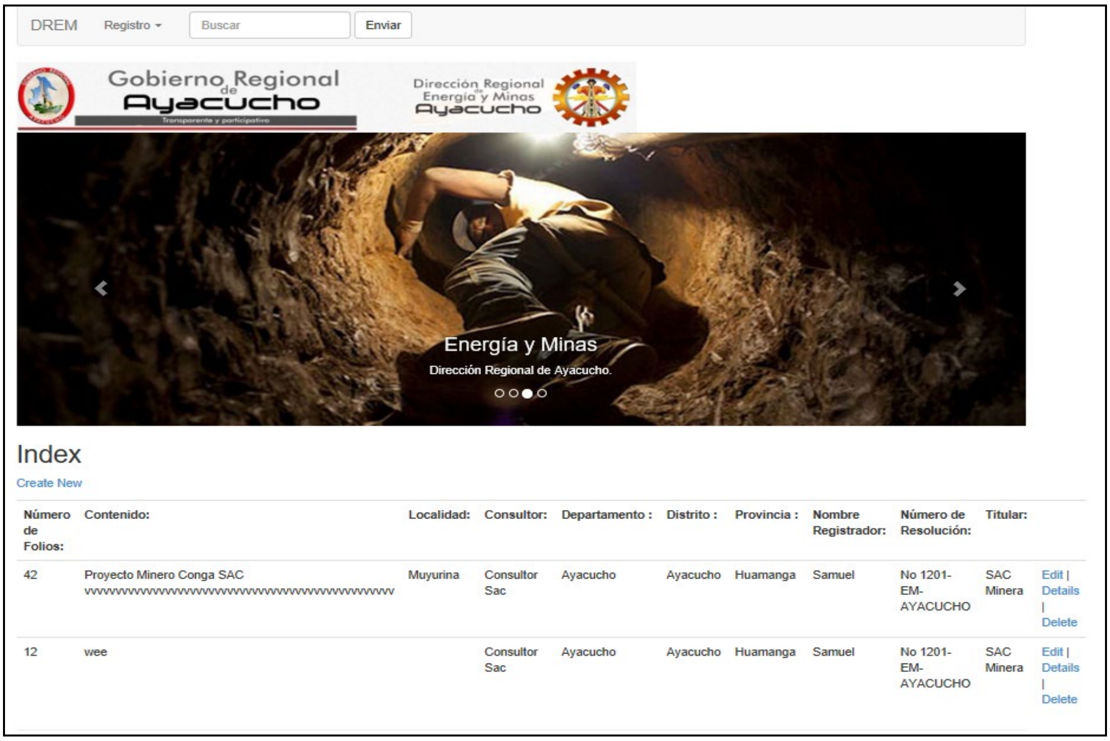

Figura N° 4.14: Prototipo de interfaz gráfica revisada "Mantener estudio ambiental."

CU 05. Mantener resolución

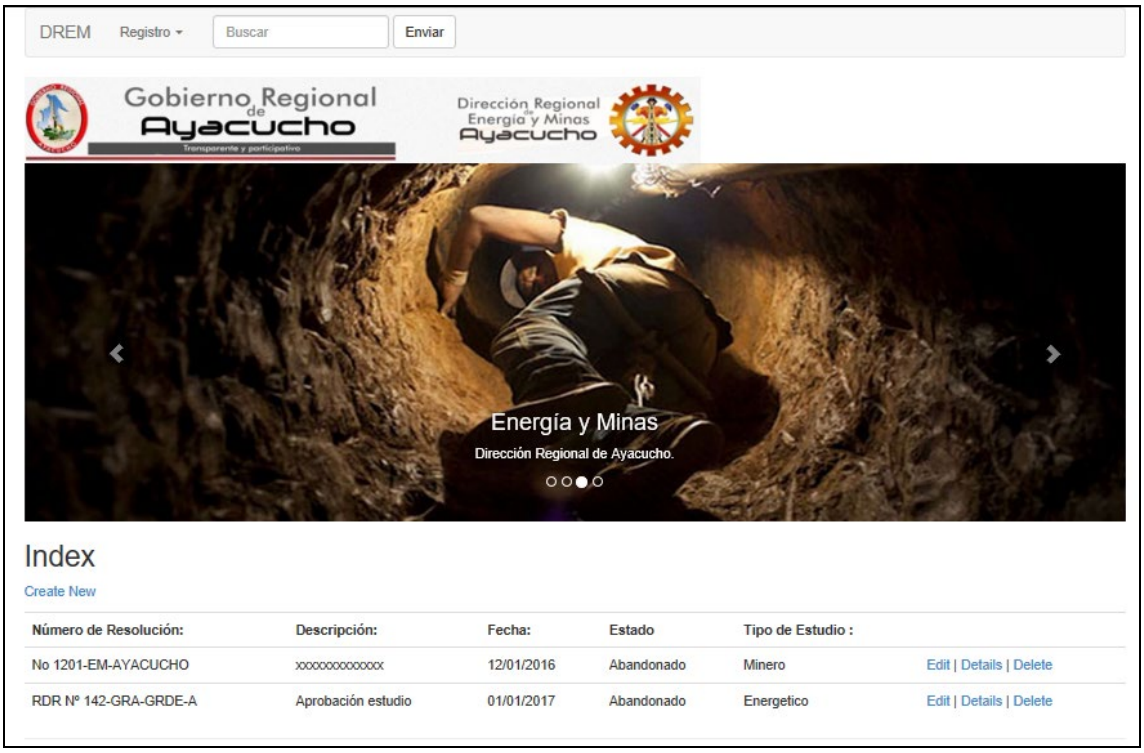

Figura N° 4.15: Prototipo de interfaz gráfica revisada "Mantener resolución."

CU 09. Obtener listado de estudio ambiental

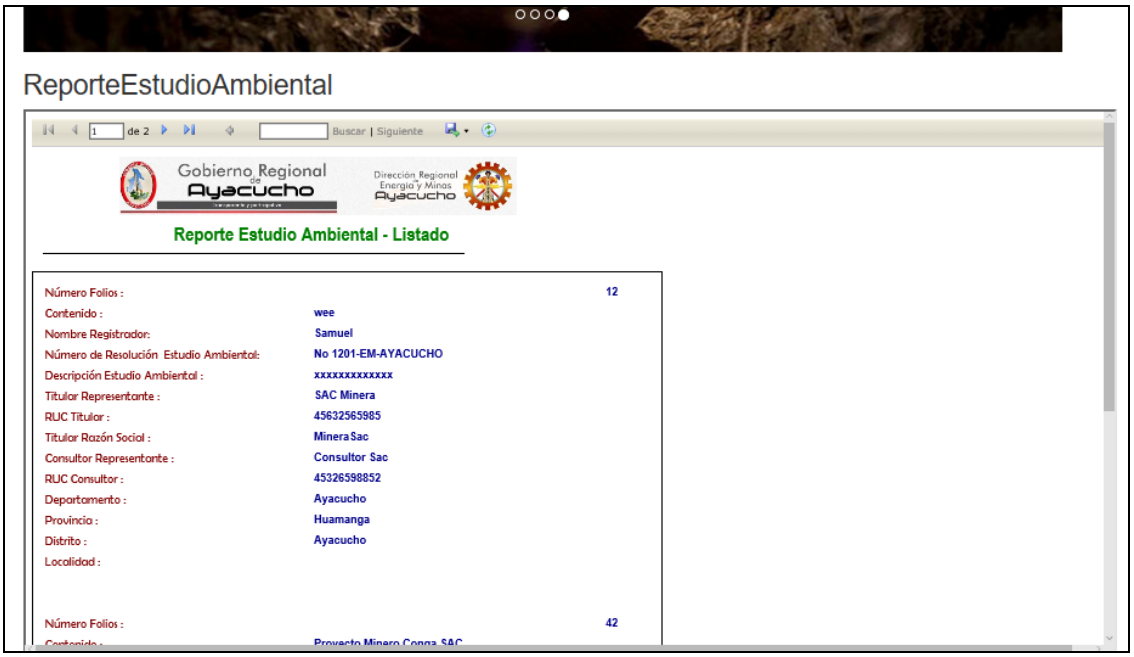

Figura N° 4.16: Prototipo de interfaz gráfica revisada "Obtener listado de estudio ambiental"

CASOS DE USO REVISADO

Caso de uso	Descripción
CU03. Mantener estudio ambiental	**Curso Básico:** El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Estudio Ambiental, el sistema muestra la Pantalla "Mantener Estudio Ambiental". El **Registrador**, para registrar un Estudio Ambiental hace clic en nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos. El **Registrador** llena los datos de resolución, el sistema llena los datos estudio, titular, Tipo, condición, ubigeo, fecha, contenido hace clic en el botón guardar el sistema guarda los datos del Estudio Ambiental. El **Registrador,** hace clic en buscar para modificar los datos el sistema muestra en la pantalla "estudios ambientales", El **Registrador**, hace clic en el estudio que desea modificar, luego presiona la tecla enter, el sistema carga los datos del Estudio Ambiental en la pantalla "Mantener Estudio Ambiental". El **Registrador** modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El **Registrador** hace clic en buscar para eliminar el Estudio Ambiental, el Sistema muestra la pantalla "Estudios Ambientales", el bibliotecario elige el Estudio Ambiental y realiza la búsqueda del Estudio Ambiental que se desea eliminar, luego presiona la tecla Suprimir; el Sistema elimina el Estudio Ambiental. Curso Alterno: El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles. El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar. El **Registrador** ingresa datos no válidos, el Sistema muestra un

	mensaje de validación de campos.

Tabla N° 4.9: Primer borrador de caso de uso “Mantener estudio ambiental”

Caso de uso	Descripción
CU05. Mantener resolución	**Curso Básico:** El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Resolución, el sistema muestra la Pantalla "Mantener Resolución Estudio Ambiental". El **Registrador**, para registrar una Resolución Estudio Ambiental hace clic en el botón nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos, El **Registrador** llena los datos; descripción, Nombre, Tipo, condición, Concesión, ubigeo y fecha. Hace clic en el botón guardar el sistema guarda los datos de la Resolución. El **Registrador,** hace clic en buscar para modificar los datos el sistema muestra en la pantalla “Resolución de Estudios Ambientales", El **Registrador**, hace clic en la resolución que desea modificar, luego presiona la tecla enter, el sistema carga los datos de la Resolución Ambiental en la pantalla “Mantener Resolución Ambiental”, el **Registrador** modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El **Registrador** hace clic en buscar para eliminar la Resolución Ambiental, el software muestra la pantalla “Resolución Ambiental”, el bibliotecario elige la Resolución Ambiental y realiza la búsqueda de la Resolución Ambiental que se desea Eliminar, luego presiona la tecla Suprimir; el Sistema elimina la Resolución Ambiental. Curso Alterno: El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles. El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar.

	El **Registrador** ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Tabla N° 4.10: Primer borrador de caso de uso “Mantener resolución”

Caso de uso	Descripción
CU09. Obtener listado de estudios ambientales	**Curso Básico:** El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el software muestra la pantalla “Estudios Ambientales”. El **Registrador**, hace clic en el botón Listado de estudios Ambientales; el sistema muestra la Pantalla "Listado de Estudios Ambientales", en la cual muestra un listado de los Estudios Ambientales registrados, en la misma pantalla el sistema realiza un resumen de la cantidad de estudios, la cantidad por tipo, y por estado. Curso Alterno: El **Registrador** hace clic en Volver, el sistema Muestra la pantalla anterior y limpia los controles.

Tabla N° 4.11: Primer borrador de caso de uso “Obtener listado de estudio ambiental”

Caso de uso	Descripción
CU16. Mostrar registro reciente	**Curso Básico:** El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el Sistema muestra la pantalla “Estudios Ambientales”. El **Registrador**, hace clic en el botón último Registro; el sistema muestra en pantalla “Ultimo Ingreso Estudios Ambientales registrados ", y los datos. Curso Alterno: El **Registrador** hace clic en Volver, muestra la pantalla “Estudios Ambientales”.

Tabla N° 4.12: Primer borrador de caso de uso “Mostrar registro reciente”

4.1.3 DISEÑO PRELIMINAR

CASOS DE USO DESAMBIGUADO Y DIAGRAMA DE ROBUSTEZ

CU 03. Mantener estudio ambiental

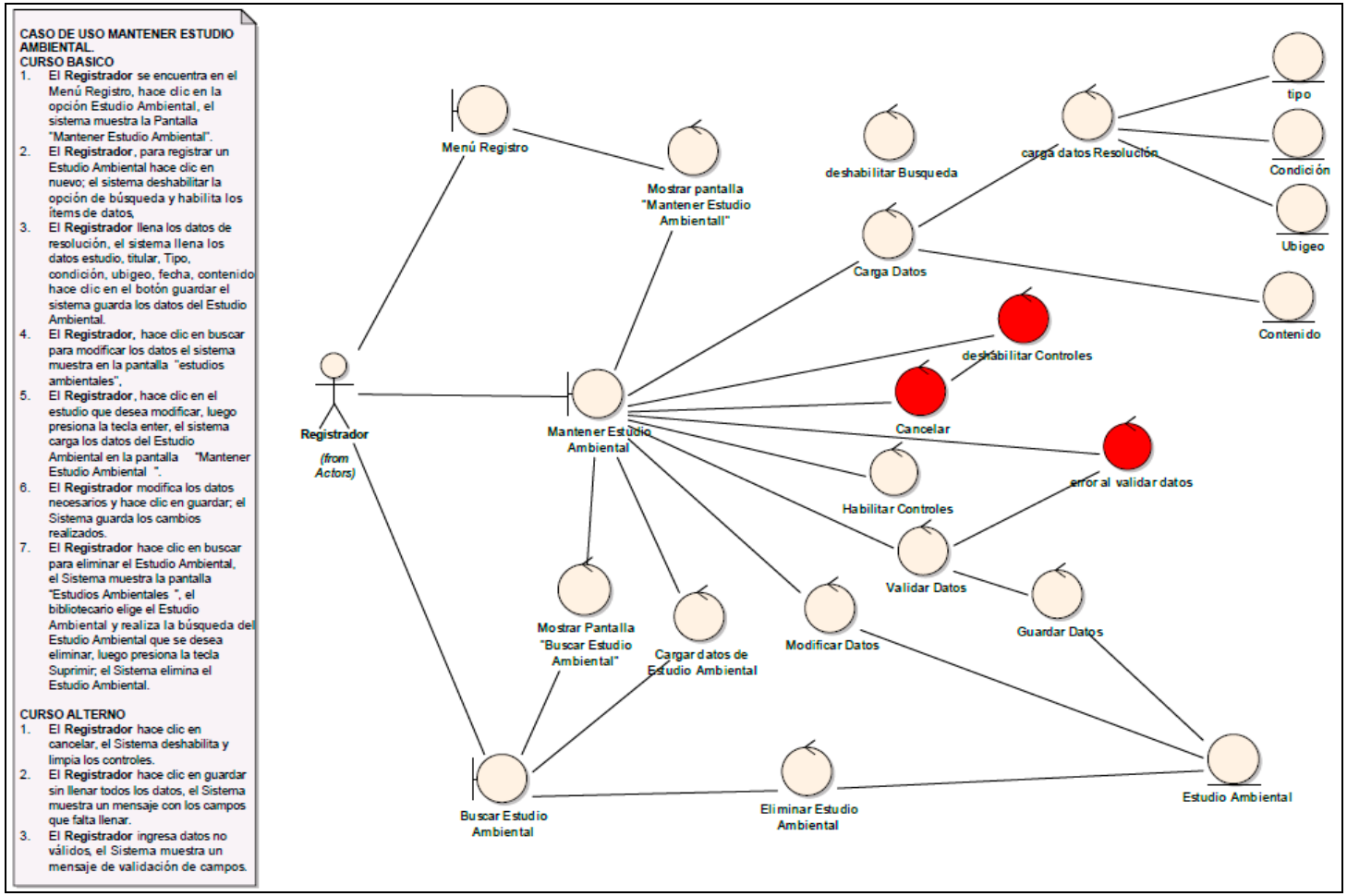

Figura N° 4.17: Diagrama de robustez de “Mantener estudio ambiental”

CU 05. Mantener resolución

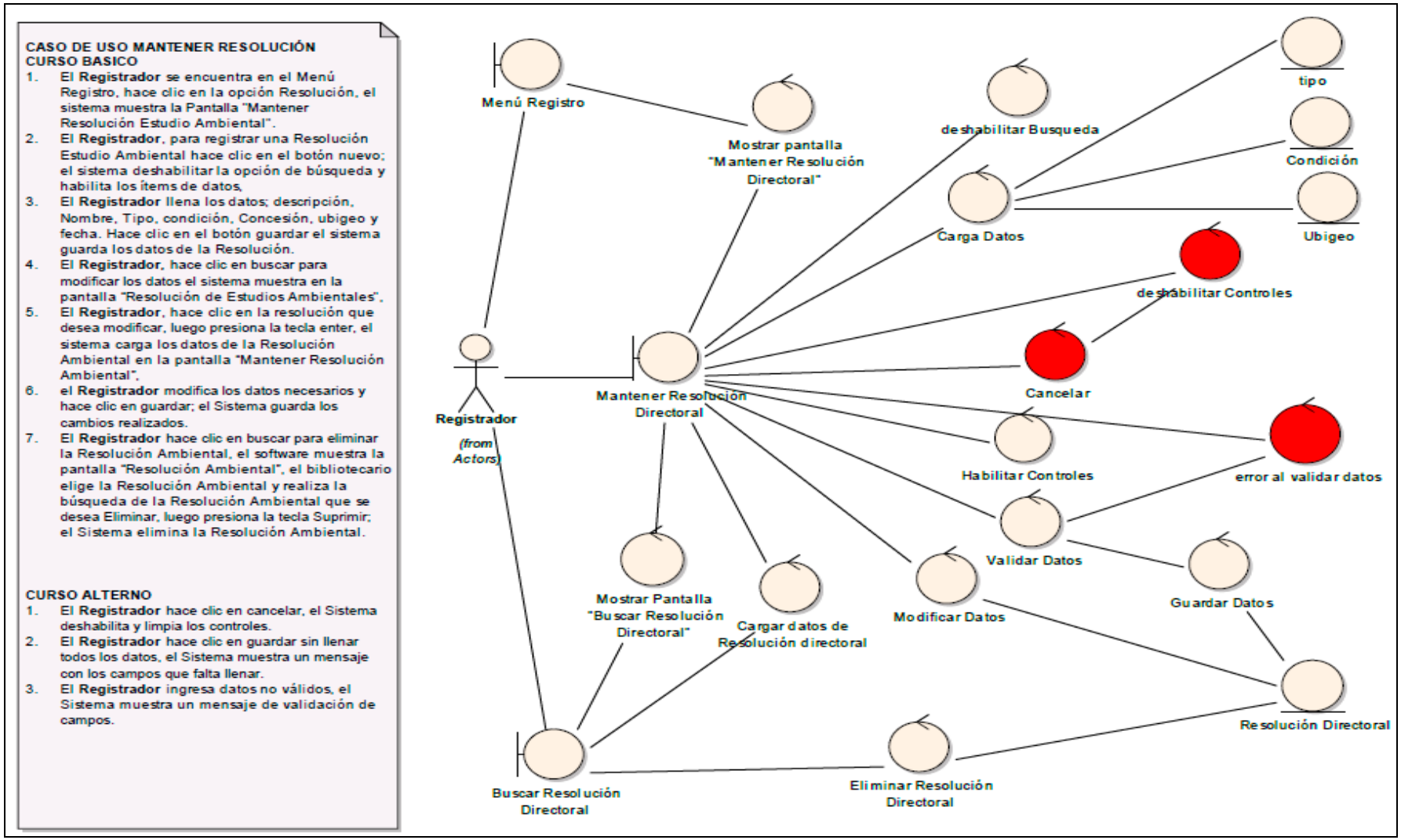

Figura N° 4.18: Diagrama de robustez de “Mantener resolución”

CU 09. Obetener Listado de Estudios Ambientales

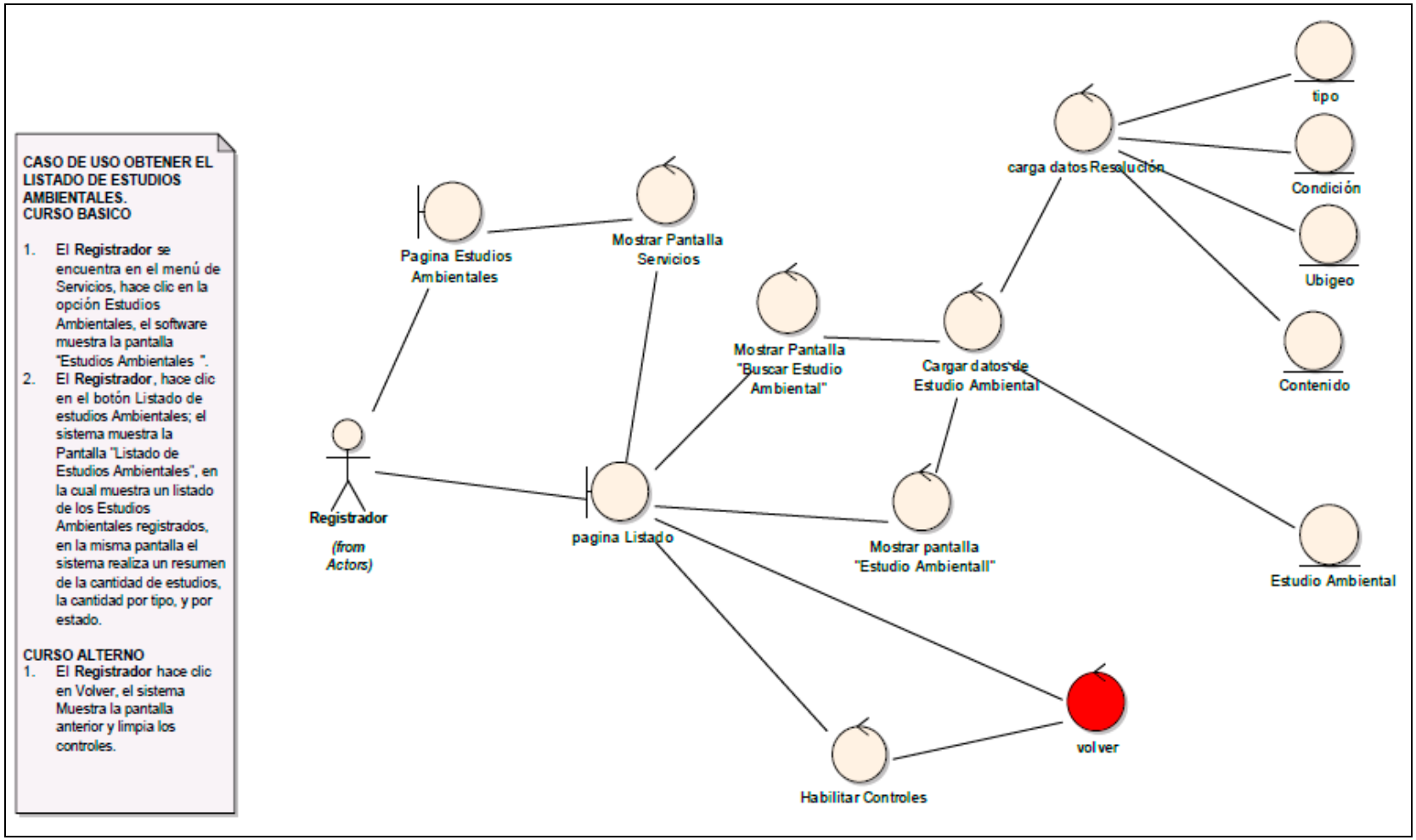

Figura N° 4.19: Diagrama de robustez de “Obtener el listado de estudios ambientales”

4.1.4. REVISIÓN DE DISEÑO PRELIMINAR

REVISIÓN DE CASOS DE USO

CU 03. Mantener estudio ambiental

Caso de uso	Descripción
CU03. Mantener estudio ambiental	**Curso Básico:** El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Estudio Ambiental, el sistema muestra la Pantalla "Mantener Estudio Ambiental". El Registrador, para registrar un Estudio Ambiental hace clic en nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos. El Registrador llena los datos de resolución, el sistema llena los datos estudio, titular, Tipo, condición, ubigeo, fecha, contenido hace clic en el botón guardar el sistema guarda los datos del Estudio Ambiental. El Registrador, hace clic en buscar para modificar los datos el sistema muestra en la pantalla "estudios ambientales", El Registrador, hace clic en el estudio que desea modificar, luego presiona la tecla enter, el sistema carga los datos del Estudio Ambiental en la pantalla “Mantener Estudio Ambiental”. El Registrador modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El Registrador hace clic en buscar para eliminar el Estudio Ambiental, el Sistema muestra la pantalla “Estudios Ambientales”, el bibliotecario elige el Estudio Ambiental y realiza la búsqueda del Estudio Ambiental que se desea eliminar, luego presiona la tecla Suprimir; el Sistema elimina el Estudio Ambiental. Curso Alterno: El Registrador hace clic en cancelar, el Sistema deshabilita y limpia los controles. El Registrador hace clic en guardar sin llenar todos los datos, el

	Sistema muestra un mensaje con los campos que falta llenar. El Registrador ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Tabla N° 4.13: Primer borrador de caso de uso “Mantener estudio ambiental”

CU 05. Mantener resolución

Caso de uso	Descripción
CU05. Mantener resolución	**Curso Básico:** El Registrador se encuentra en el Menú Registro, hace clic en la opción Resolución, el sistema muestra la Pantalla "Mantener Resolución Estudio Ambiental". El Registrador, para registrar una Resolución Estudio Ambiental hace clic en el botón nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos. El Registrador llena los datos; descripción, Nombre, Tipo, condición, Concesión, ubigeo y fecha. Hace clic en el botón guardar el sistema guarda los datos de la Resolución. El Registrador, hace clic en buscar para modificar los datos el sistema muestra en la pantalla “Resolución de Estudios Ambientales", El Registrador, hace clic en la resolución que desea modificar, luego presiona la tecla enter, el sistema carga los datos de la Resolución Ambiental en la pantalla “Mantener Resolución Ambiental”, El Registrador modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados. El Registrador hace clic en buscar para eliminar la Resolución Ambiental, el software muestra la pantalla “Resolución Ambiental”, el bibliotecario elige la Resolución Ambiental y realiza la búsqueda de la Resolución Ambiental que se desea Eliminar, luego presiona la tecla Suprimir; el Sistema elimina la Resolución Ambiental. Curso Alterno: El Registrador hace clic en cancelar, el Sistema deshabilita y

	limpia los controles. El Registrador hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar. El Registrador ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Tabla N° 4.14: Primer borrador de caso de uso “Mantener resolución”

CU 09. Obtener listado de estudios ambientales

Caso de uso	Descripción
CU09. Obtener listado de estudios ambientales	**Curso Básico:** El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el software muestra la pantalla “Estudios Ambientales”. El Registrador, hace clic en el botón Listado de estudios Ambientales; el sistema muestra la Pantalla "Listado de Estudios Ambientales", en la cual muestra un listado de los Estudios Ambientales registrados, en la misma pantalla el sistema realiza un resumen de la cantidad de estudios, la cantidad por tipo, y por estado. Curso Alterno: El **Registrador** hace clic en Volver, el sistema Muestra la pantalla anterior y limpia los controles.

Tabla N° 4.15: Primer borrador de caso de uso “Obtener listado de estudios ambientales”

4.1.5. ARQUITECTURA TÉCNICA

La presente investigación se basó en la adaptabilidad del Patrón MVC del Framework .Net a la arquitectura técnica de la metodología ICONIX, es por eso que en esta fase se proponen las tecnologías para cada uno de las capas del Patrón MVC.

a) MODELO

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class Consultor
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Consultor()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Consultor:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public int IdConsultor { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Consultor:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Representante { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "RUC:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        [RegularExpression("\\d{11}", ErrorMessage = "RUC  - 11 digitos")]
        public string Ruc { get; set; }
```

```
        [System.ComponentModel.DataAnnotations.Display(Name = "Razón Social:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string RazonSocial { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Teléfono:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Telefono { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Email:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Email no válido")]
        public string Email { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
    }
}
```

Tabla N° 4.16: Código usando Entity Framework para generar la clase Consultor.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Departamento
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Departamento()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
```

```
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Departamento :")]
        public int IdDepartamento { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Departamento :")]
        public string NombreDepartamento { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
    }
}
```

Tabla N° 4.17: Código usando Entity Framework para generar la clase Departamento.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Distrito
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Distrito()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Distrito :")]
        public int IdDistrito { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Distrito :")]
        public string NombreDistrito { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
```

```
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
    }
}
```

Tabla N° 4.18: Código usando Entity Framework para generar la clase Distrito.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class EstadoEstudio
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public EstadoEstudio()
        {
            this.Resolucion = new HashSet<Resolucion>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Estado del Estudio
:")]
        public int IdEstadoEstudio { get; set; }
        public string Estado { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Resolucion> Resolucion { get; set; }
    }
}
```

Tabla N° 4.19: Código usando Entity Framework para generar la clase Estado Estudio.

```
namespace WebApplication1.Models
{
    using System;
```

```
    using System.Collections.Generic;

    public partial class EstadoTrabajo
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public EstadoTrabajo()
        {
            this.Registrador = new HashSet<Registrador>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Estado de Trabajo
:")]
        public int IdEstadoTrabajo { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Condición Laboral
:")]
        public string CondicionLaboral { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Registrador> Registrador { get; set; }
    }
}
```

Tabla N° 4.20: Código usando Entity Framework para generar la clase Estado Trabajo.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class EstudioAmbiental
    {
        public int IdEstudioAmbiental { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Número de
```

```
Folios:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public Nullable<int> NumFolios { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Contenido:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Contenido { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Registrador:")]
        public Nullable<int> IdRegistrador { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Resolución:")]
        public Nullable<int> IdResolucion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Titular:")]
        public Nullable<int> IdTitular { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Consultor:")]
        public Nullable<int> IdConsultor { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Departamento:")]
        public Nullable<int> IdDepartamento { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Provincia:")]
        public Nullable<int> IdProvincia { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Distrito:")]
        public Nullable<int> IdDistrito { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Localidad:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Localidad { get; set; }


        public virtual Consultor Consultor { get; set; }
        public virtual Departamento Departamento { get; set; }
        public virtual Distrito Distrito { get; set; }
        public virtual Provincia Provincia { get; set; }
        public virtual Registrador Registrador { get; set; }
        public virtual Resolucion Resolucion { get; set; }
```

```
        public virtual Titular Titular { get; set; }
    }
}
```

Tabla N° 4.21: Código usando Entity Framework para generar la clase Estudio Ambiental.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Provincia
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Provincia()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Provincia :")]
        public int IdProvincia { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Provincia :")]
        public string NombreProvincia { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
    }
}
```

Tabla N° 4.22: Código usando Entity Framework para generar la clase Provincia.

```
namespace WebApplication1.Models
{
    using System;
```

```
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

public partial class Registrador
{
    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
    public Registrador()
    {
        this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
    }

    public int IdRegistrador { get; set; }
    [System.ComponentModel.DataAnnotations.Display(Name = "Nombre
Registrador:")]
    [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
    public string Nombre { get; set; }
    [System.ComponentModel.DataAnnotations.Display(Name = "Apellido
Paterno:")]
    [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
    public string ApellidoPaterno { get; set; }
    [System.ComponentModel.DataAnnotations.Display(Name = "Apellido
Materno:")]
    [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
    public string ApellidoMaterno { get; set; }
    [System.ComponentModel.DataAnnotations.Display(Name = "DNI:")]
    [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
    [RegularExpression("\\d{8}", ErrorMessage = "DNI Incorrecto")]
    public string Dni { get; set; }
```

```
        [System.ComponentModel.DataAnnotations.Display(Name = "Dirección:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Direccion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Teléfono:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Telefono { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Email:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Email no válido")]
        public string Email { get; set; }
        public Nullable<int> IdSexo { get; set; }
        public Nullable<int> IdEstadoTrabajo { get; set; }

        public virtual EstadoTrabajo EstadoTrabajo { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
        public virtual Sexo Sexo { get; set; }
    }
}
```

Tabla N° 4.23: Código usando Entity Framework para generar la clase Registrador.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class Resolucion
    {
```

```
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Resolucion()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
        }

        public int IdResolucion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Número de
Resolución:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string NumeroResolucion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Descripción:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Descripcion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Fecha:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio:")]
        [DataType(DataType.Date, ErrorMessage = "Email no válido")]
        public Nullable<System.DateTime> Fecha { get; set; }
        public Nullable<int> IdTipoEstudio { get; set; }
        public Nullable<int> IdEstadoEstudio { get; set; }

        public virtual EstadoEstudio EstadoEstudio { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
        public virtual TipoEstudio TipoEstudio { get; set; }
    }
}
```

Tabla N° 4.24: Código usando Entity Framework para generar la clase Resolución.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Sexo
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Sexo()
        {
            this.Registrador = new HashSet<Registrador>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Sexo :")]
        public int IdSexo { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Sexo :")]
        public string TipoSexo { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Registrador> Registrador { get; set; }
    }
}
```

Tabla N° 4.25: Código usando Entity Framework para generar la clase Sexo.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class TipoEstudio
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
```

```
        public TipoEstudio()
        {
            this.Resolucion = new HashSet<Resolucion>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Tipo de Estudio :")]
        public int IdTipoEstudio { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Tipo de Estudio :")]
        public string EstudioTipo { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Resolucion> Resolucion { get; set; }
    }
}
```

Tabla N° 4.26: Código usando Entity Framework para generar la clase Tipo Estudio.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;

    public partial class TipoPersona
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public TipoPersona()
        {
            this.Titular = new HashSet<Titular>();
        }
        [System.ComponentModel.DataAnnotations.Display(Name = "Tipo de Persona
:")]
        public int IdTipoPersona { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Tipo de Persona
```

```
:")]
        public string PersonaTipo { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<Titular> Titular { get; set; }
    }
}
```

Tabla N° 4.27: Código usando Entity Framework para generar la clase Tipo Persona.

```
namespace WebApplication1.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;

    public partial class Titular
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2214:DoNotCallOverridableMethodsInConstructors")]
        public Titular()
        {
            this.EstudioAmbiental = new HashSet<EstudioAmbiental>();
        }

        public int IdTitular { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Titular:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Representante { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "RUC:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
```

```
        [RegularExpression("\\d{11}", ErrorMessage = "RUC Incorrecto")]
        public string Ruc { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Razón Social:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string RazonSocial { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Tipo de Persona:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public Nullable<int> IdTipoPersona { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Dirección:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Direccion { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Email :")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        [DataType(DataType.EmailAddress, ErrorMessage = "Email no válido")]
        public string Email { get; set; }
        [System.ComponentModel.DataAnnotations.Display(Name = "Teléfono:")]
        [System.ComponentModel.DataAnnotations.Required(ErrorMessage = "Contenido
Vacio")]
        public string Telefono { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
        public virtual ICollection<EstudioAmbiental> EstudioAmbiental { get; set; }
        public virtual TipoPersona TipoPersona { get; set; }
    }
}
```

Tabla N° 4.28: Código usando Entity Framework para generar la clase Tipo Titular.

b) VISTA

```
@model WebApplication1.Models.Consultor

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Nuevo</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Consultor</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Representante, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Representante, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Representante, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Ruc, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.EditorFor(model => model.Ruc, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Ruc, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.RazonSocial, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.RazonSocial, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.RazonSocial, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Telefono, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Telefono, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Telefono, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.29: Código usando Razor para generar la vista Crear Consultor.

```
@model WebApplication1.Models.Consultor

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar Consultor</h2>
```

```
<h3>¿Estás seguro que quieres eliminar?</h3>
<div>

    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Representante)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Representante)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Ruc)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Ruc)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.RazonSocial)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.RazonSocial)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Telefono)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.Telefono)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Regresar", "Index")
        </div>
    }
</div>
```

Tabla N° 4.30: Código usando Razor para generar la vista Eliminar Consultor.

```
@model WebApplication1.Models.Consultor

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

```
<h2>Detalles  - Consultor</h2>

<div>

    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Representante)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Representante)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Ruc)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Ruc)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.RazonSocial)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.RazonSocial)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Telefono)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.Telefono)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.IdConsultor }) |
    @Html.ActionLink("Regresar", "Index")
</p>
```

Tabla N° 4.31: Código usando Razor para generar la vista Detalle Consultor.

```
@model WebApplication1.Models.Consultor

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Editar  Consultor</h2>


@using (Html.BeginForm())
{
```

```
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.IdConsultor)

        <div class="form-group">
            @Html.LabelFor(model => model.Representante, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Representante, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Representante, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Ruc, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Ruc, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Ruc, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.RazonSocial, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
            <div class="col-md-10">
                @Html.EditorFor(model => model.RazonSocial, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.RazonSocial, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Telefono, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Telefono, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Telefono, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
```

```
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.32: Código usando Razor para generar la vista Editar Consultor.

```
@model IEnumerable<WebApplication1.Models.Consultor>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Mantener Consultor</h2>

<p>
    @Html.ActionLink("Crear Nuevo", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Representante)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Ruc)
```

```
        </th>
        <th>
            @Html.DisplayNameFor(model => model.RazonSocial)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Telefono)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Representante)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Ruc)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.RazonSocial)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Telefono)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id=item.IdConsultor }) |
            @Html.ActionLink("Ver Dettale", "Details", new { id=item.IdConsultor }) |
```

```
            @Html.ActionLink("Eliminar", "Delete", new { id=item.IdConsultor })
        </td>
    </tr>
}

</table>
```

Tabla N° 4.33: Código usando Razor para generar la vista Mantener Consultor.

```
@model WebApplication1.Models.EstudioAmbiental

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Nuevo Estudio Ambiental</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.NumFolios, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumFolios, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NumFolios, "", new { @class
```

```
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Contenido, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Contenido, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Contenido, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdRegistrador, "Registrador", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdRegistrador", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdRegistrador, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdResolucion, "Resolución:", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdResolucion", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdResolucion, "", new {
```

```
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdTitular, "Titular:", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdTitular", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdTitular, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdConsultor, "Consultor:", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdConsultor", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdConsultor, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdDepartamento, "Departamento:", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdDepartamento", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdDepartamento, "", new {
```

```
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdProvincia, "Provincia:", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdProvincia", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdProvincia, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdDistrito, "Distrito:", htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdDistrito", null, htmlAttributes: new { @class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdDistrito, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Localidad, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Localidad, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Localidad, "", new { @class
```

```
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.34: Código usando Razor para generar la vista Crear Estudio Ambiental.

```
@model WebApplication1.Models.EstudioAmbiental

@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar</h2>

<h3>¿Estás seguro que quieres eliminar?</h3>
<div>
```

```
<hr />
<dl class="dl-horizontal">
    <dt>
        @Html.DisplayNameFor(model => model.NumFolios)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.NumFolios)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Contenido)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Contenido)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Localidad)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Localidad)
    </dd>

    <dt>
        @Html.DisplayNameFor(model => model.Consultor.Representante)
    </dt>

    <dd>
        @Html.DisplayFor(model => model.Consultor.Representante)
    </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model                                          =>
model.Departamento.NombreDepartamento)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Departamento.NombreDepartamento)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Distrito.NombreDistrito)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Distrito.NombreDistrito)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Provincia.NombreProvincia)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Provincia.NombreProvincia)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Registrador.Nombre)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Registrador.Nombre)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.Resolucion.NumeroResolucion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Resolucion.NumeroResolucion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Titular.Representante)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Titular.Representante)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Regresar", "Index")
        </div>
    }
</div>
```

Tabla N° 4.35: Código usando Razor para generar la vista Borrar Estudio Ambiental.

```
@model WebApplication1.Models.EstudioAmbiental

@{
```

```
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Detalles Estudio Ambiental</h2>

<div>

    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.NumFolios)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumFolios)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Contenido)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Contenido)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Localidad)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Localidad)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.Consultor.Representante)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Consultor.Representante)
        </dd>

        <dt>
            @Html.DisplayNameFor(model                                    =>
model.Departamento.NombreDepartamento)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Departamento.NombreDepartamento)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Distrito.NombreDistrito)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Distrito.NombreDistrito)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Provincia.NombreProvincia)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Provincia.NombreProvincia)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.Registrador.Nombre)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Registrador.Nombre)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Resolucion.NumeroResolucion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Resolucion.NumeroResolucion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Titular.Representante)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Titular.Representante)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.IdEstudioAmbiental }) |
    @Html.ActionLink("Regresar", "Index")
</p>
```

Tabla N° 4.36: Código usando Razor para generar la vista Detalle Estudio Ambiental.

```
@model WebApplication1.Models.EstudioAmbiental

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Editar Estudio Ambiental</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.IdEstudioAmbiental)

        <div class="form-group">
            @Html.LabelFor(model => model.NumFolios, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumFolios, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NumFolios, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Contenido, htmlAttributes: new { @class =
```

```
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Contenido, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Contenido, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdRegistrador, "IdRegistrador",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdRegistrador", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdRegistrador, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdResolucion, "IdResolucion",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdResolucion", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdResolucion, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdTitular, "IdTitular", htmlAttributes: new {
```

```
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdTitular", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdTitular, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdConsultor, "IdConsultor", htmlAttributes:
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdConsultor", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdConsultor, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdDepartamento, "IdDepartamento",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdDepartamento", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.IdDepartamento, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdProvincia, "IdProvincia", htmlAttributes:
```

```
new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdProvincia", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdProvincia, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdDistrito, "IdDistrito", htmlAttributes: new
{ @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdDistrito", null, htmlAttributes: new { @class =
"form-control" })
                @Html.ValidationMessageFor(model => model.IdDistrito, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Localidad, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Localidad, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Localidad, "", new { @class
= "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
```

```
            <input type="submit" value="Save" class="btn btn-default" />
        </div>
    </div>
</div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.37: Código usando Razor para generar la vista Editar Estudio Ambiental.

```
@model IEnumerable<WebApplication1.Models.EstudioAmbiental>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Mantener Estudio Ambiental</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.NumFolios)
        </th>
        <th>
```

```
            @Html.DisplayNameFor(model => model.Contenido)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Localidad)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Consultor.Representante)
        </th>
        <th>
            @Html.DisplayNameFor(model                                    =>
model.Departamento.NombreDepartamento)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Distrito.NombreDistrito)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Provincia.NombreProvincia)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Registrador.Nombre)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Resolucion.NumeroResolucion)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Titular.Representante)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
```

```
            @Html.DisplayFor(modelItem => item.NumFolios)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Contenido)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Localidad)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Consultor.Representante)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Departamento.NombreDepartamento)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Distrito.NombreDistrito)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Provincia.NombreProvincia)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Registrador.Nombre)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Resolucion.NumeroResolucion)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Titular.Representante)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id=item.IdEstudioAmbiental }) |
            @Html.ActionLink("Detalles", "Details", new { id=item.IdEstudioAmbiental })
|
```

```
            @Html.ActionLink("Eliminar", "Delete", new { id=item.IdEstudioAmbiental })
        </td>
    </tr>
}

</table>
```

Tabla N° 4.38: Código usando Razor para generar la vista Mantener Estudio Ambiental.

```
@model WebApplication1.Models.Resolucion

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Nuevo</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.NumeroResolucion, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumeroResolucion, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NumeroResolucion, "", new
```

```
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Descripcion, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Descripcion, new { htmlAttributes = new
{ @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Descripcion, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Fecha, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Fecha, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Fecha, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdTipoEstudio, "IdTipoEstudio",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdTipoEstudio", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.IdTipoEstudio, "", new {
```

```
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdEstadoEstudio, "IdEstadoEstudio",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdEstadoEstudio", null, htmlAttributes: new {
@class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdEstadoEstudio, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.39: Código usando Razor para generar la vista Crear Resolución.

```
@model WebApplication1.Models.Resolucion
```

```
@{
    ViewBag.Title = "Delete";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Eliminar Resolución Directoral</h2>

<h3>¿Estás seguro que quieres eliminar?</h3>
<div>

    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.NumeroResolucion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumeroResolucion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Descripcion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Descripcion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Fecha)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.Fecha)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EstadoEstudio.Estado)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EstadoEstudio.Estado)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TipoEstudio.EstudioTipo)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TipoEstudio.EstudioTipo)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Regresar", "Index")
        </div>
    }
</div>
```

Tabla N° 4.40: Código usando Razor para generar la vista Borrar Resolución.

```
@model WebApplication1.Models.Resolucion

@{
    ViewBag.Title = "Details";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Detalles Resolucion</h2>

<div>
    <h4></h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.NumeroResolucion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.NumeroResolucion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Descripcion)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Descripcion)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Fecha)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.Fecha)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.EstadoEstudio.Estado)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.EstadoEstudio.Estado)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.TipoEstudio.EstudioTipo)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.TipoEstudio.EstudioTipo)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Editar", "Edit", new { id = Model.IdResolucion }) |
    @Html.ActionLink("Regresar", "Index")
</p>
```

Tabla N° 4.41: Código usando Razor para generar la vista Detalle Resolución.

```
@model WebApplication1.Models.Resolucion

@{
    ViewBag.Title = "Edit";
    Layout = "~/Views/Shared/_Layout.cshtml";
```

```
}

<h2>Editar Resolucion</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">

        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.IdResolucion)

        <div class="form-group">
            @Html.LabelFor(model => model.NumeroResolucion, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.NumeroResolucion, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.NumeroResolucion, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Descripcion, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Descripcion, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Descripcion, "", new {
```

```
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Fecha, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Fecha, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Fecha, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdTipoEstudio, "IdTipoEstudio",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdTipoEstudio", null, htmlAttributes: new { @class
= "form-control" })
                @Html.ValidationMessageFor(model => model.IdTipoEstudio, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.IdEstadoEstudio, "IdEstadoEstudio",
htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("IdEstadoEstudio", null, htmlAttributes: new {
@class = "form-control" })
                @Html.ValidationMessageFor(model => model.IdEstadoEstudio, "", new {
```

```
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Regresar", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Tabla N° 4.42: Código usando Razor para generar la vista Editar Resolución.

```
@model IEnumerable<WebApplication1.Models.Resolucion>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Mantener Resolución Directoral Regional</h2>

<p>
    @Html.ActionLink("Nuevo", "Create")
</p>
```

```
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.NumeroResolucion)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Descripcion)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Fecha)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.EstadoEstudio.Estado)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.TipoEstudio.EstudioTipo)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.NumeroResolucion)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Descripcion)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Fecha)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.EstadoEstudio.Estado)
```

```
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.TipoEstudio.EstudioTipo)
        </td>
        <td>
            @Html.ActionLink("Editar", "Edit", new { id=item.IdResolucion }) |
            @Html.ActionLink("Detalles", "Details", new { id=item.IdResolucion }) |
            @Html.ActionLink("Eliminar", "Delete", new { id=item.IdResolucion })
        </td>
    </tr>
}

</table>
```

Tabla N° 4.43: Código usando Razor para generar la vista Mantener Resolución.

```
@{
    ViewBag.Title = "ReporteEstudioAmbiental";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>ReporteEstudioAmbiental</h2>
<iframe id="myReport" width="1200" height="600"
src="~/Reportes/ReportesEnergiaMinas.aspx">

</iframe>
```

Tabla N° 4.44: Código usando Razor para generar la vista Reporte Estudio Ambiental.

c) CONTROLADOR

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
```

```
using System.Net;
using System.Web;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class ConsultorsController : Controller
    {
        private BDEstudioAmbientalEntities db = new BDEstudioAmbientalEntities();

        // GET: Consultors
        public ActionResult Index()
        {
            return View(db.Consultor.ToList());
        }

        // GET: Consultors/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Consultor consultor = db.Consultor.Find(id);
            if (consultor == null)
            {
                return HttpNotFound();
            }
            return View(consultor);
        }

        // GET: Consultors/Create
```

```
        public ActionResult Create()
        {
            return View();
        }

        // POST: Consultors/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "IdConsultor,Representante,Ruc,RazonSocial,Telefono,Email")] Consultor consultor)
        {
            if (ModelState.IsValid)
            {
                db.Consultor.Add(consultor);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(consultor);
        }

        // GET: Consultors/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Consultor consultor = db.Consultor.Find(id);
            if (consultor == null)
```

```
            {
                return HttpNotFound();
            }
            return View(consultor);
        }

        // POST: Consultors/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "IdConsultor,Representante,Ruc,RazonSocial,Telefono,Email")] Consultor consultor)
        {
            if (ModelState.IsValid)
            {
                db.Entry(consultor).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(consultor);
        }

        // GET: Consultors/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Consultor consultor = db.Consultor.Find(id);
            if (consultor == null)
```

```
            {
                return HttpNotFound();
            }
            return View(consultor);
        }

        // POST: Consultors/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Consultor consultor = db.Consultor.Find(id);
            db.Consultor.Remove(consultor);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }}
```

Tabla N° 4.45: Código C# del controlador Consultor.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
```

```
using System.Web;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
   public class EstudioAmbientalsController : Controller
   {
      private BDEstudioAmbientalEntities db = new BDEstudioAmbientalEntities();

      // GET: EstudioAmbientals
      public ActionResult Index()
      {
         var estudioAmbiental = db.EstudioAmbiental.Include(e => e.Consultor).Include(e => e.Departamento).Include(e => e.Distrito).Include(e => e.Provincia).Include(e => e.Registrador).Include(e => e.Resolucion).Include(e => e.Titular);
         return View(estudioAmbiental.ToList());
      }

      // GET: EstudioAmbientals/Details/5
      public ActionResult Details(int? id)
      {
         if (id == null)
         {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         EstudioAmbiental estudioAmbiental = db.EstudioAmbiental.Find(id);
         if (estudioAmbiental == null)
         {
            return HttpNotFound();
         }
         return View(estudioAmbiental);
```

```
        }

        // GET: EstudioAmbientals/Create
        public ActionResult Create()
        {
            ViewBag.IdConsultor = new SelectList(db.Consultor, "IdConsultor", "Representante");
            ViewBag.IdDepartamento = new SelectList(db.Departamento, "IdDepartamento", "NombreDepartamento");
            ViewBag.IdDistrito = new SelectList(db.Distrito, "IdDistrito", "NombreDistrito");
            ViewBag.IdProvincia = new SelectList(db.Provincia, "IdProvincia", "NombreProvincia");
            ViewBag.IdRegistrador = new SelectList(db.Registrador, "IdRegistrador", "Nombre");
            ViewBag.IdResolucion = new SelectList(db.Resolucion, "IdResolucion", "NumeroResolucion");
            ViewBag.IdTitular = new SelectList(db.Titular, "IdTitular", "Representante");
            return View();
        }

        // POST: EstudioAmbientals/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "IdEstudioAmbiental,NumFolios,Contenido,IdRegistrador,IdResolucion,IdTitular,IdConsultor,IdDepartamento,IdProvincia,IdDistrito,Localidad")] EstudioAmbiental estudioAmbiental)
        {
            if (ModelState.IsValid)
```

```
            {
                db.EstudioAmbiental.Add(estudioAmbiental);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdConsultor = new SelectList(db.Consultor, "IdConsultor", "Representante", estudioAmbiental.IdConsultor);
            ViewBag.IdDepartamento = new SelectList(db.Departamento, "IdDepartamento", "NombreDepartamento", estudioAmbiental.IdDepartamento);
            ViewBag.IdDistrito = new SelectList(db.Distrito, "IdDistrito", "NombreDistrito", estudioAmbiental.IdDistrito);
            ViewBag.IdProvincia = new SelectList(db.Provincia, "IdProvincia", "NombreProvincia", estudioAmbiental.IdProvincia);
            ViewBag.IdRegistrador = new SelectList(db.Registrador, "IdRegistrador", "Nombre", estudioAmbiental.IdRegistrador);
            ViewBag.IdResolucion = new SelectList(db.Resolucion, "IdResolucion", "NumeroResolucion", estudioAmbiental.IdResolucion);
            ViewBag.IdTitular = new SelectList(db.Titular, "IdTitular", "Representante", estudioAmbiental.IdTitular);
            return View(estudioAmbiental);
        }

        // GET: EstudioAmbientals/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            EstudioAmbiental estudioAmbiental = db.EstudioAmbiental.Find(id);
            if (estudioAmbiental == null)
            {
```

```
            return HttpNotFound();
        }
        ViewBag.IdConsultor = new SelectList(db.Consultor, "IdConsultor", "Representante", estudioAmbiental.IdConsultor);
        ViewBag.IdDepartamento = new SelectList(db.Departamento, "IdDepartamento", "NombreDepartamento", estudioAmbiental.IdDepartamento);
        ViewBag.IdDistrito = new SelectList(db.Distrito, "IdDistrito", "NombreDistrito", estudioAmbiental.IdDistrito);
        ViewBag.IdProvincia = new SelectList(db.Provincia, "IdProvincia", "NombreProvincia", estudioAmbiental.IdProvincia);
        ViewBag.IdRegistrador = new SelectList(db.Registrador, "IdRegistrador", "Nombre", estudioAmbiental.IdRegistrador);
        ViewBag.IdResolucion = new SelectList(db.Resolucion, "IdResolucion", "NumeroResolucion", estudioAmbiental.IdResolucion);
        ViewBag.IdTitular = new SelectList(db.Titular, "IdTitular", "Representante", estudioAmbiental.IdTitular);
        return View(estudioAmbiental);
    }

    // POST: EstudioAmbientals/Edit/5
    // To protect from overposting attacks, please enable the specific properties you want to bind to, for
    // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Edit([Bind(Include = "IdEstudioAmbiental,NumFolios,Contenido,IdRegistrador,IdResolucion,IdTitular,IdConsultor,IdDepartamento,IdProvincia,IdDistrito,Localidad")] EstudioAmbiental estudioAmbiental)
    {
        if (ModelState.IsValid)
        {
            db.Entry(estudioAmbiental).State = EntityState.Modified;
```

```
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.IdConsultor = new SelectList(db.Consultor, "IdConsultor", "Representante", estudioAmbiental.IdConsultor);
        ViewBag.IdDepartamento = new SelectList(db.Departamento, "IdDepartamento", "NombreDepartamento", estudioAmbiental.IdDepartamento);
        ViewBag.IdDistrito = new SelectList(db.Distrito, "IdDistrito", "NombreDistrito", estudioAmbiental.IdDistrito);
        ViewBag.IdProvincia = new SelectList(db.Provincia, "IdProvincia", "NombreProvincia", estudioAmbiental.IdProvincia);
        ViewBag.IdRegistrador = new SelectList(db.Registrador, "IdRegistrador", "Nombre", estudioAmbiental.IdRegistrador);
        ViewBag.IdResolucion = new SelectList(db.Resolucion, "IdResolucion", "NumeroResolucion", estudioAmbiental.IdResolucion);
        ViewBag.IdTitular = new SelectList(db.Titular, "IdTitular", "Representante", estudioAmbiental.IdTitular);
        return View(estudioAmbiental);
    }

    // GET: EstudioAmbientals/Delete/5
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        EstudioAmbiental estudioAmbiental = db.EstudioAmbiental.Find(id);
        if (estudioAmbiental == null)
        {
            return HttpNotFound();
        }
        return View(estudioAmbiental);
```

```
        }

        // POST: EstudioAmbientals/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            EstudioAmbiental estudioAmbiental = db.EstudioAmbiental.Find(id);
            db.EstudioAmbiental.Remove(estudioAmbiental);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Tabla N° 4.46: Código C# del controlador Estudio Ambiental.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
```

```
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
   public class RegistradorsController : Controller
   {
      private BDEstudioAmbientalEntities db = new BDEstudioAmbientalEntities();

      // GET: Registradors
      public ActionResult Index()
      {
         var registrador = db.Registrador.Include(r => r.EstadoTrabajo).Include(r => r.Sexo);
         return View(registrador.ToList());
      }

      // GET: Registradors/Details/5
      public ActionResult Details(int? id)
      {
         if (id == null)
         {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         Registrador registrador = db.Registrador.Find(id);
         if (registrador == null)
         {
            return HttpNotFound();
         }
         return View(registrador);
      }

      // GET: Registradors/Create
      public ActionResult Create()
```

```
        {
            ViewBag.IdEstadoTrabajo = new SelectList(db.EstadoTrabajo, "IdEstadoTrabajo", "CondicionLaboral");
            ViewBag.IdSexo = new SelectList(db.Sexo, "IdSexo", "TipoSexo");
            return View();
        }

        // POST: Registradors/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "IdRegistrador,Nombre,ApellidoPaterno,ApellidoMaterno,Dni,Direccion,Telefono,Email,IdSexo,IdEstadoTrabajo")] Registrador registrador)
        {
            if (ModelState.IsValid)
            {
                db.Registrador.Add(registrador);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdEstadoTrabajo = new SelectList(db.EstadoTrabajo, "IdEstadoTrabajo", "CondicionLaboral", registrador.IdEstadoTrabajo);
            ViewBag.IdSexo = new SelectList(db.Sexo, "IdSexo", "TipoSexo", registrador.IdSexo);
            return View(registrador);
        }

        // GET: Registradors/Edit/5
        public ActionResult Edit(int? id)
```

```
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Registrador registrador = db.Registrador.Find(id);
            if (registrador == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdEstadoTrabajo = new SelectList(db.EstadoTrabajo, "IdEstadoTrabajo", "CondicionLaboral", registrador.IdEstadoTrabajo);
            ViewBag.IdSexo = new SelectList(db.Sexo, "IdSexo", "TipoSexo", registrador.IdSexo);
            return View(registrador);
        }

        // POST: Registradors/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "IdRegistrador,Nombre,ApellidoPaterno,ApellidoMaterno,Dni,Direccion,Telefono,Email,IdSexo,IdEstadoTrabajo")] Registrador registrador)
        {
            if (ModelState.IsValid)
            {
                db.Entry(registrador).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
```

```
            ViewBag.IdEstadoTrabajo = new SelectList(db.EstadoTrabajo,
"IdEstadoTrabajo", "CondicionLaboral", registrador.IdEstadoTrabajo);
            ViewBag.IdSexo = new SelectList(db.Sexo, "IdSexo", "TipoSexo",
registrador.IdSexo);
            return View(registrador);
        }

        // GET: Registradors/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Registrador registrador = db.Registrador.Find(id);
            if (registrador == null)
            {
                return HttpNotFound();
            }
            return View(registrador);
        }

        // POST: Registradors/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Registrador registrador = db.Registrador.Find(id);
            db.Registrador.Remove(registrador);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
```

```
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }}}
```

Tabla N° 4.47: Código C# del controlador Registrador.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace WebApplication1.Controllers
{
    public class ReporteEstudioAmbientalController : Controller
    {
        // GET: ReporteEstudioAmbiental
        public ActionResult ReporteEstudioAmbiental()
        {
            return View();
        }
    }
}
```

Tabla N° 4.48: Código C# del controlador Reporte Estudio Ambiental.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
```

```
using System.Web;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
   public class ResolucionsController : Controller
   {
      private BDEstudioAmbientalEntities db = new BDEstudioAmbientalEntities();

      // GET: Resolucions
      public ActionResult Index()
      {
         var resolucion = db.Resolucion.Include(r => r.EstadoEstudio).Include(r => r.TipoEstudio);
         return View(resolucion.ToList());
      }

      // GET: Resolucions/Details/5
      public ActionResult Details(int? id)
      {
         if (id == null)
         {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
         }
         Resolucion resolucion = db.Resolucion.Find(id);
         if (resolucion == null)
         {
            return HttpNotFound();
         }
         return View(resolucion);
      }
```

```
        // GET: Resolucions/Create
        public ActionResult Create()
        {
            ViewBag.IdEstadoEstudio = new SelectList(db.EstadoEstudio, "IdEstadoEstudio", "Estado");
            ViewBag.IdTipoEstudio = new SelectList(db.TipoEstudio, "IdTipoEstudio", "EstudioTipo");
            return View();
        }

        // POST: Resolucions/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "IdResolucion,NumeroResolucion,Descripcion,Fecha,IdTipoEstudio,IdEstadoEstudio")] Resolucion resolucion)
        {
            if (ModelState.IsValid)
            {
                db.Resolucion.Add(resolucion);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdEstadoEstudio = new SelectList(db.EstadoEstudio, "IdEstadoEstudio", "Estado", resolucion.IdEstadoEstudio);
            ViewBag.IdTipoEstudio = new SelectList(db.TipoEstudio, "IdTipoEstudio", "EstudioTipo", resolucion.IdTipoEstudio);
            return View(resolucion);
        }
```

```
        // GET: Resolucions/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Resolucion resolucion = db.Resolucion.Find(id);
            if (resolucion == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdEstadoEstudio = new SelectList(db.EstadoEstudio, "IdEstadoEstudio", "Estado", resolucion.IdEstadoEstudio);
            ViewBag.IdTipoEstudio = new SelectList(db.TipoEstudio, "IdTipoEstudio", "EstudioTipo", resolucion.IdTipoEstudio);
            return View(resolucion);
        }

        // POST: Resolucions/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "IdResolucion,NumeroResolucion,Descripcion,Fecha,IdTipoEstudio,IdEstadoEstudio")] Resolucion resolucion)
        {
            if (ModelState.IsValid)
            {
                db.Entry(resolucion).State = EntityState.Modified;
```

```
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        ViewBag.IdEstadoEstudio = new SelectList(db.EstadoEstudio,
"IdEstadoEstudio", "Estado", resolucion.IdEstadoEstudio);
        ViewBag.IdTipoEstudio = new SelectList(db.TipoEstudio, "IdTipoEstudio",
"EstudioTipo", resolucion.IdTipoEstudio);
        return View(resolucion);
    }

    // GET: Resolucions/Delete/5
    public ActionResult Delete(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Resolucion resolucion = db.Resolucion.Find(id);
        if (resolucion == null)
        {
            return HttpNotFound();
        }
        return View(resolucion);
    }

    // POST: Resolucions/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public ActionResult DeleteConfirmed(int id)
    {
        Resolucion resolucion = db.Resolucion.Find(id);
        db.Resolucion.Remove(resolucion);
        db.SaveChanges();
```

```
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Tabla N° 4.49: Código C# del controlador Resolución.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using WebApplication1.Models;

namespace WebApplication1.Controllers
{
    public class TitularsController : Controller
    {
        private BDEstudioAmbientalEntities db = new BDEstudioAmbientalEntities();

        // GET: Titulars
        public ActionResult Index()
```

```
        {
            var titular = db.Titular.Include(t => t.TipoPersona);
            return View(titular.ToList());
        }

        // GET: Titulars/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Titular titular = db.Titular.Find(id);
            if (titular == null)
            {
                return HttpNotFound();
            }
            return View(titular);
        }

        // GET: Titulars/Create
        public ActionResult Create()
        {
            ViewBag.IdTipoPersona = new SelectList(db.TipoPersona, "IdTipoPersona",
"PersonaTipo");
            return View();
        }

        // POST: Titulars/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = "IdTitular,Representante,Ruc,RazonSocial,IdTipoPersona,Direccion,Email,Telefono")] Titular titular)
        {
            if (ModelState.IsValid)
            {
                db.Titular.Add(titular);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.IdTipoPersona = new SelectList(db.TipoPersona, "IdTipoPersona", "PersonaTipo", titular.IdTipoPersona);
            return View(titular);
        }

        // GET: Titulars/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Titular titular = db.Titular.Find(id);
            if (titular == null)
            {
                return HttpNotFound();
            }
            ViewBag.IdTipoPersona = new SelectList(db.TipoPersona, "IdTipoPersona", "PersonaTipo", titular.IdTipoPersona);
            return View(titular);
        }
```

```
        // POST: Titulars/Edit/5
        // To protect from overposting attacks, please enable the specific properties you want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include = "IdTitular,Representante,Ruc,RazonSocial,IdTipoPersona,Direccion,Email,Telefono")] Titular titular)
        {
            if (ModelState.IsValid)
            {
                db.Entry(titular).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.IdTipoPersona = new SelectList(db.TipoPersona, "IdTipoPersona", "PersonaTipo", titular.IdTipoPersona);
            return View(titular);
        }

        // GET: Titulars/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Titular titular = db.Titular.Find(id);
            if (titular == null)
            {
                return HttpNotFound();
```

```
            }
            return View(titular);
        }

        // POST: Titulars/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Titular titular = db.Titular.Find(id);
            db.Titular.Remove(titular);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Tabla N° 4.50: Código C# del controlador Titular.

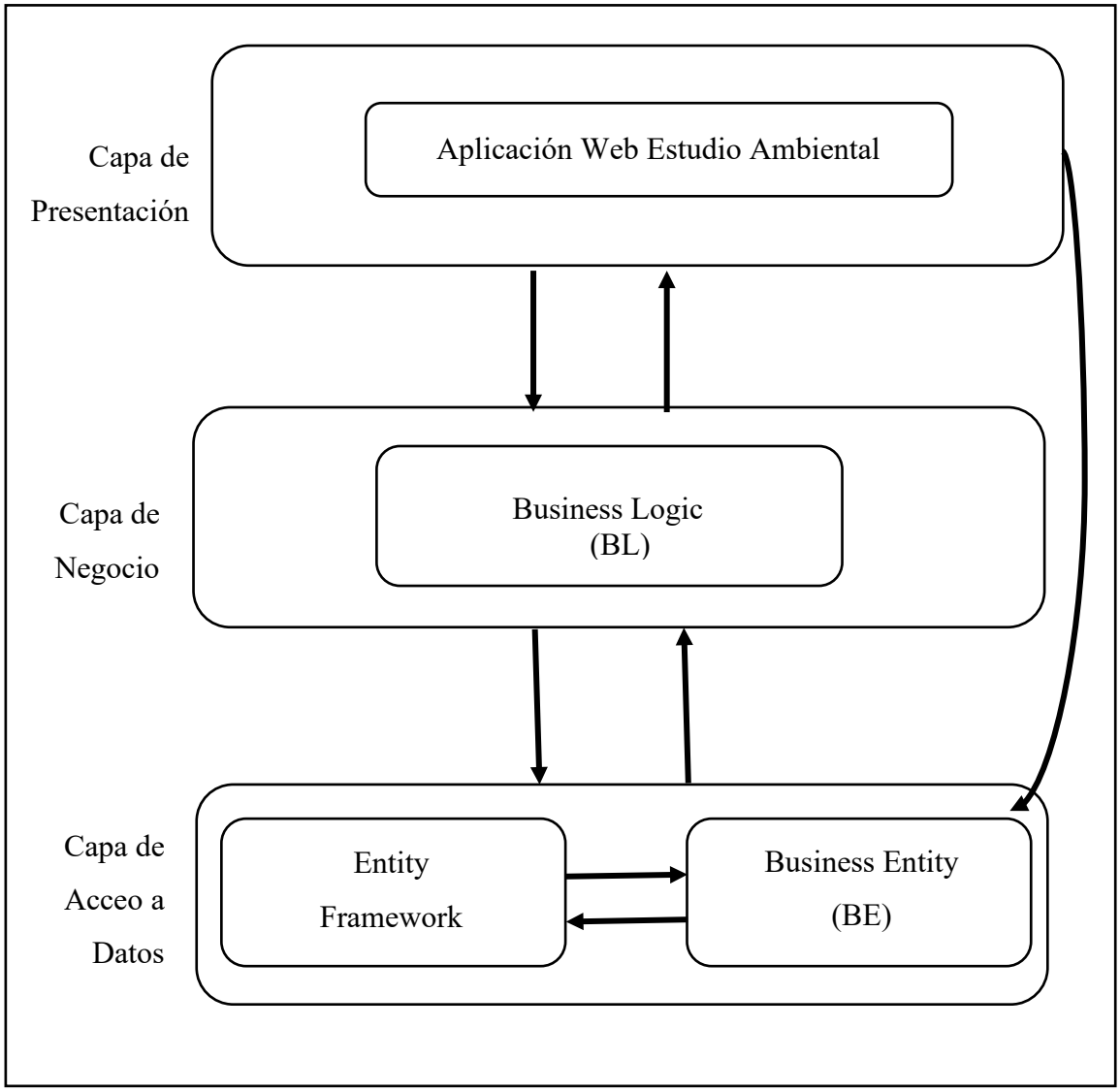

Figura N° 4.18: Arquitectura técnica. Diagrama de componentes

Figura N° 4.19: Arquitectura técnica. Diagrama de despliegue

4.1.6. DISEÑO DETALLADO

DIAGRAMA DE SECUENCIA

CU 03. Mantener estudio ambiental

SO DE USO MANTENER ESTUDIO BIENTAL.

RSO BASICO

1) El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Estudio Ambiental, el sistema muestra la Pantalla "Mantener Estudio Ambiental".
2) El **Registrador**, para registrar un Estudio Ambiental hace clic en nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos,
3) El **Registrador** llena los datos de resolución, el sistema llena los datos estudio, titular, Tipo, condición, ubigeo, fecha, contenido hace clic en el botón guardar el sistema guarda los datos del Estudio Ambiental.
4) El **Registrador,** hace clic en buscar para modificar los datos el sistema muestra en la pantalla "estudios ambientales",
5) El **Registrador**, hace clic en el estudio que desea modificar, luego presiona la tecla enter, el sistema carga los datos del Estudio Ambiental en la pantalla "Mantener Estudio Ambiental".
6) El **Registrador** modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados.
7) El **Registrador** hace clic en buscar para eliminar el Estudio Ambiental, el Sistema muestra la pantalla "Estudios Ambientales", el bibliotecario elige el Estudio Ambiental y realiza la búsqueda del Estudio Ambiental que se desea eliminar, luego presiona la tecla Suprimir; el Sistema elimina el Estudio Ambiental.

RSO ALTERNO

1) El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles.
2) El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar.
3) El **Registrador** ingresa datos no válidos, el Sistema muestra un mensaje de validación de campos.

Figura N° 4.20: Diagrama de secuencia de "Mantener estudio ambiental"

CU 05. Mantener resolución

CASO DE USO MANTENER RESOLUCIÓN
CURSO BASICO

1) El **Registrador** se encuentra en el Menú Registro, hace clic en la opción Resolución, el sistema muestra la Pantalla "Mantener Resolución Estudio Ambiental".
2) El **Registrador**, para registrar una Resolución Estudio Ambiental hace clic en el botón nuevo; el sistema deshabilitar la opción de búsqueda y habilita los ítems de datos,
3) El **Registrador** llena los datos; descripción, Nombre, Tipo, condición, Concesión, ubigeo y fecha. Hace clic en el botón guardar el sistema guarda los datos de la Resolución.
4) El **Registrador,** hace clic en buscar para modificar los datos el sistema muestra en la pantalla "Resolución de Estudios Ambientales",
5) El **Registrador**, hace clic en la resolución que desea modificar, luego presiona la tecla enter, el sistema carga los datos de la Resolución Ambiental en la pantalla "Mantener Resolución Ambiental",
6) el **Registrador** modifica los datos necesarios y hace clic en guardar; el Sistema guarda los cambios realizados.
7) El **Registrador** hace clic en buscar para eliminar la Resolución Ambiental, el software muestra la pantalla "Resolución Ambiental", el bibliotecario elige la Resolución Ambiental y realiza la búsqueda de la Resolución Ambiental que se desea Eliminar, luego presiona la tecla Suprimir; el Sistema elimina la Resolución Ambiental.

CURSO ALTERNO

1) El **Registrador** hace clic en cancelar, el Sistema deshabilita y limpia los controles.
2) El **Registrador** hace clic en guardar sin llenar todos los datos, el Sistema muestra un mensaje con los campos que falta llenar.
3) El **Registrador** ingresa datos no válidos, el Sistema muestra

Figura N° 4.21: Diagrama de secuencia de "Mantener resolución"

CU 09. Obtener listado de estudios ambientales

CASO DE USO OBTENER EL LISTADO DE ESTUDIOS AMBIENTALES.
CURSO BASICO

1) El **Registrador** se encuentra en el menú de Servicios, hace clic en la opción Estudios Ambientales, el software muestra la pantalla “Estudios Ambientales".
2) El **Registrador**, hace clic en el botón Listado de estudios Ambientales; el sistema muestra la Pantalla "Listado de Estudios Ambientales", en la cual muestra un listado de los Estudios Ambientales registrados, en la misma pantalla el sistema realiza un resumen de la cantidad de estudios, la cantidad por tipo, y por estado.

CURSO ALTERNO

1) El **Registrador** hace clic en Volver, el sistema Muestra la pantalla anterior y limpia los controles.

Figura N° 4.22: Diagrama de secuencia de “Obtener listado de estudios ambientales”

DIAGRAMA DE CLASES DE DISEÑO

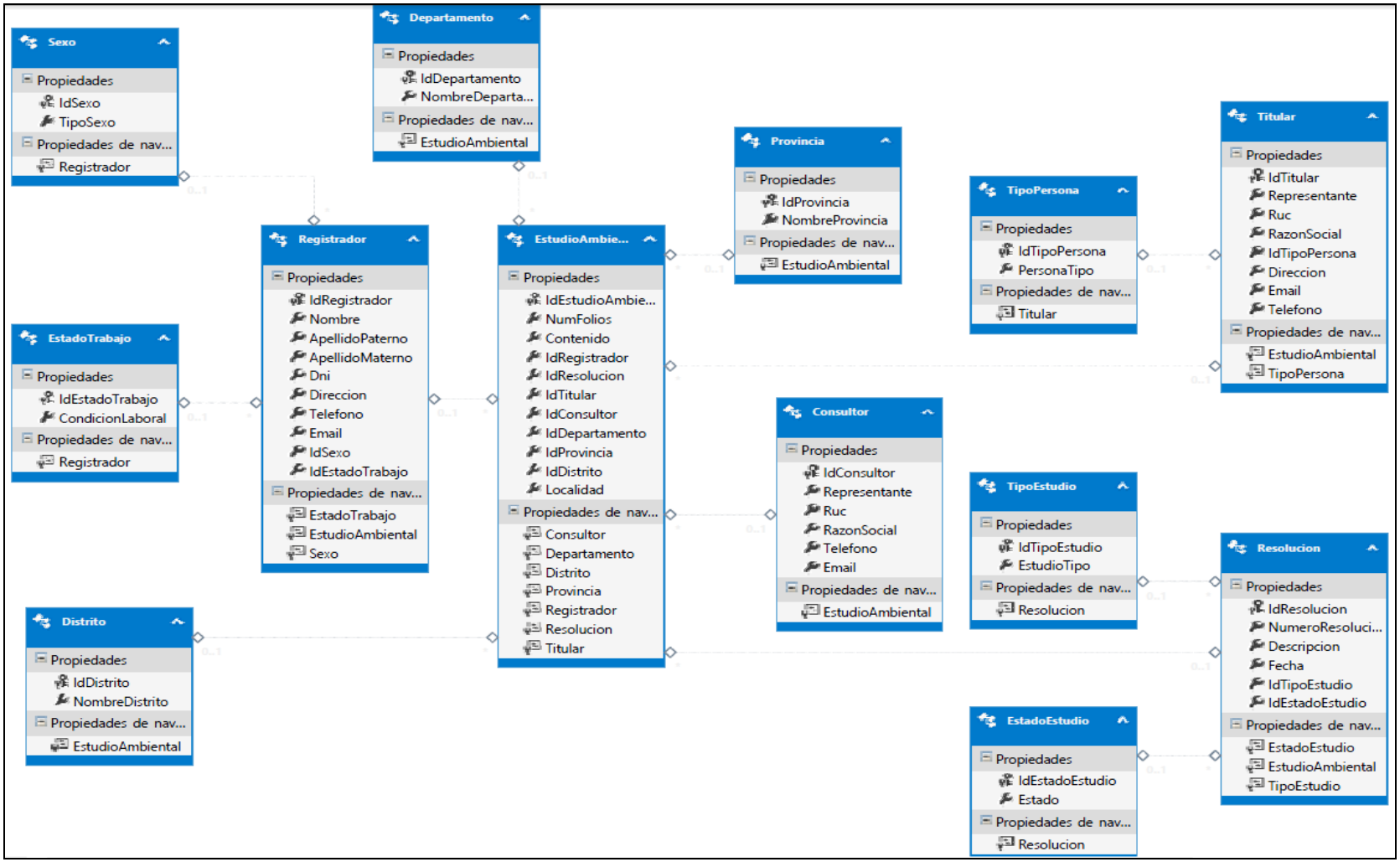

Figura N° 4.23: Diagrama de clases de diseño

4.1.7. IMPLEMENTACIÓN

BASE DE DATOS FÍSICA

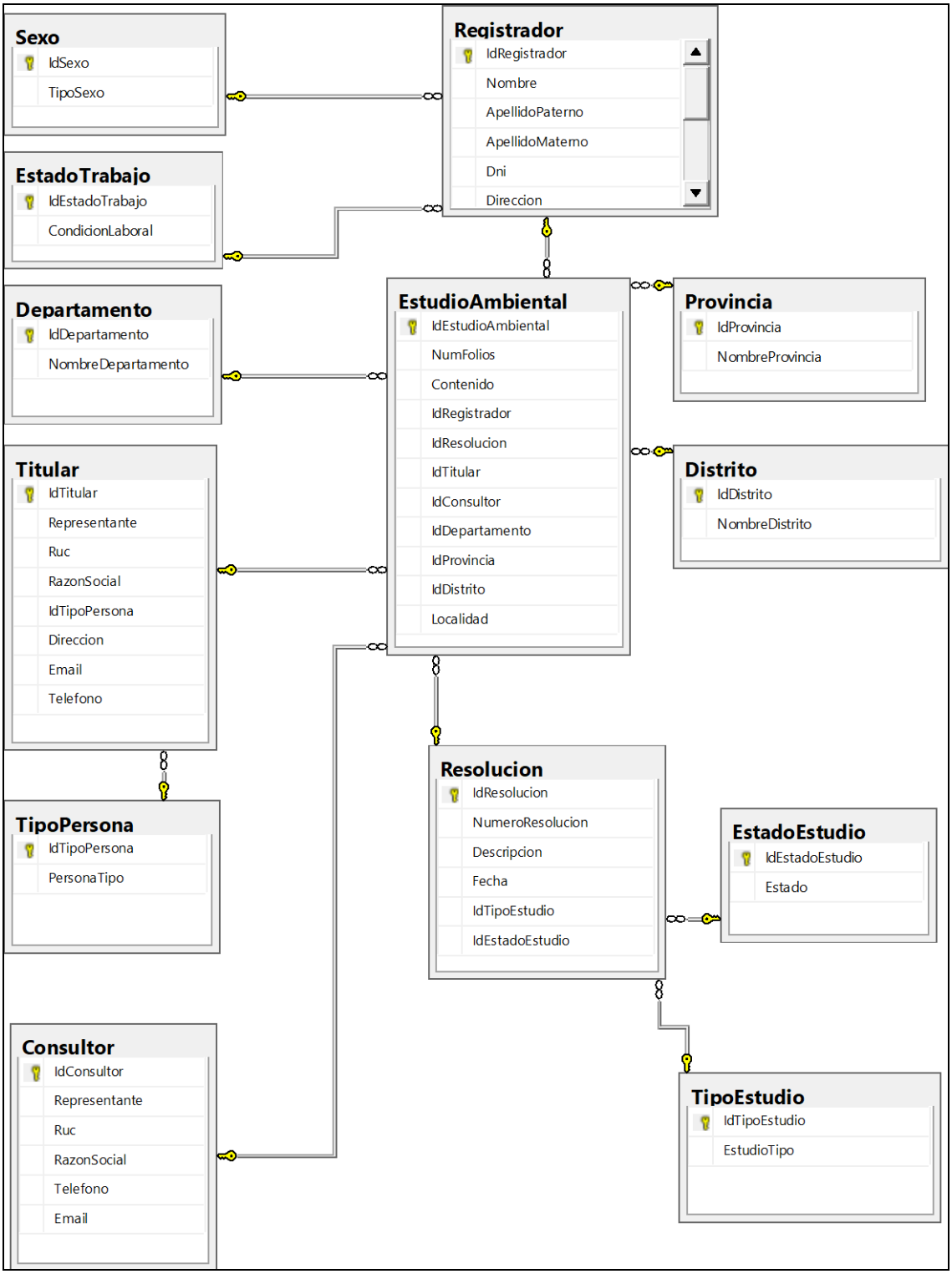

Figura N° 4.24: Base de datos física

4.2. RESULTADOS

Se propone adaptar el Patrón MVC del framework .net a la arquitectura técnica de la metodología ágil y formal ICONIX a través del caso de estudio: registro de estudios ambientales del DREMA.

a) Se propone usar Entity Framework del Patrón MVC del Framework .Net en la capa de Modelo, debido a la gran facilidad que nos da para mapear las clases de nuestro programa orientado a objetos y tablas en la base de datos, ya que Entity Framework crea las entidades orientadas a objetos (las clases) de manera automática, y las actualizara en caso de que haya cambios en la base de datos.

b) Se propone usar Razor del Patrón MVC del Framework .Net en la capa de Vista, debido a que este motor es como un lenguaje que nos permite crear código HTML, con ayuda de este motor podemos llenar grillas, combos, menus, así mismo pudimos crear controles HTML (formularios, inputs, labels,etc.). A diferencia de aspx, en la que podemos "jalar" controles desde nuestra caja de herramientas y automáticamente se nos genera todo el código necesario, lo cual es ventaja y desventaja al mismo tiempo: ventaja porque creamos un control en segundos y desventaja porque el no todo el código generado es necesario o para poder cambiar algún simple atributo se debe hacer desde el código en servidor, en Razor, en cambio, debemos digitar todo el código (y sin opción a vista previa), y claro que intellisense nos ayudara a que esto no sea un calvario, pero podremos manipular más a fondo el código generado. Esta es su mayor ventaja pero como no todo puede ser perfecto, razor nos obliga a conocer mas a fondo el lenguaje HTML, JS y CSS, lo cual en el fondo tampoco es tan malo, pero nos hará sufrir un poco más al inicio

c) Se propone usar Clases Controladores C# del Patrón MVC del Framework .Net en la capa de Controlador, debido a que C# es un lenguaje de programación de alto nivel y orientado a objetos. Es normalmente más eficiente que Java y tiene características útiles como la sobrecarga de operadores. C# está basado en C++ pero tiene varias ventajas sobre este lenguaje más antiguo: tiene tipado seguro, orientación a objetos más comprensiva, y la sintaxis ha sido simplificada en importantes formas. Y lo más importante, C# interactúa excepcionalmente bien

con otros lenguajes de la plataforma .NET. Por esta razón, C# es una mejor opción para construir aplicaciones para .NET.

4.3. ANALISIS DE RESULTADOS

Basandose en la correcta ejecución de la aplicación de Estudios Ambientales en la DREMA (actualmente en uso), en el numeral 4.1) y 4.1.5), del Capitulo IV-artefactos de software aplicando programación extrema y arquitectura técnica respectivamente, se concluye que al el Modelo ADO.NET Entity Framework del Patrón MVC del Framework .Net, el Motor de Vista Razor del Patrón MVC del Framework .Net, las Clases Controladores C# del Patrón MVC del Framework .Net son mecanismos de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

1. De acuerdo al numeral 4.3), del Capítulo IV - Análisis de Resultados, se afirma coherentemente con la hipótesis específica propuesta, que el Entity Framework del Patrón MVC del framework .Net es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

2. De acuerdo al numeral 4.3), del Capítulo IV - Análisis de Resultados, coherentemente con la hipótesis específica propuesta, se afirma que el Motor de Vista Razor del Patrón MVC del framework .Net es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

3. De acuerdo al numeral 4.3), del Capítulo IV - Análisis de Resultados, coherentemente con la hipótesis específica propuesta, se afirma las Clases Controladores C# del Patrón MVC del framework .Net es un mecanismo de adaptabilidad a la Arquitectura Técnica de la Metodología Ágil y Formal ICONIX.

5.2. RECOMENDACIONES

1. Se recomienda realizar la investigación de adaptabilidad de la metodología agil y formal ICONIX para un entorno libre de código abierto.

2. Se recomienda contruir una herramienta automatizada de generación de código para la arquitectura técnica de la metodología agil y formal ICONIX.

3. Se recomienda el análisis de la arquitectura técnica para un entorno de N capas.

BIBLIOGRAFÍA

Alfaro, C. (2012). *Metodología de investigación científica aplicado a la ingeniería.* Proyecto de investigación. Universidad Nacional del Callao, Lima, Perú.

Alvira, M. (2002).*Perspectiva cualitativa / perspectiva cuantitativa en la metodología sociológica*. México. México: Mc Graw Hill.

Anderson, R. (2017). *Agregar un controlador a una aplicación de ASP.NET Core MV*.Recuperado el 22 de octubre de 2018, de https://docs.microsoft.com/es-es/aspnet/core/tutorials/first-mvc-app/adding-controller?view=aspnetcore-2.1

Barranco, J. (2001). *Metodología del Análisis Estructurado (2ª Ed.)*.España: Universidad Pontificia Comillas.

Driscoll, B., Gupta, N., Vettor, R., Hirani, Z. y Tenny, L. (2013). *Entity Framework 6 Recipes*. EEUU: Apress.

Freeman, A. (2013). Pro ASP.NET MVC 5.EEUU: Apress

Galloway, J. & Wilson, B. & Allen, K. & Matson, D. (2014). *Professional ASP.NET MVC 5*. EEUU: Apress.

Gonzáles, H. (2015). *MVC 4 con .Net desde cero: Guía práctica para implementar MVC 4 con C# y Visual Studio 2012/2013 (Spanish Edition)* .Recuperado el 22 de octubre de 2018, de https://books.google.com.pe/books?id=hAYtDwAAQBAJ&pg=PT83&dq=MOTOR+DE+VISTA+RAZOR&hl=es&sa=X&ved=0ahUKEwjR-rzdwaDeAhVIFJAKHU4QDfcQ6AEIMjAC#v=onepage&q=MOTOR%20DE%20VISTA%20RAZOR&f=false

Jiménez, R. (1998). *Metodología de la Investigación. Elementos básicos para la investigación clínica*. La Habana, Cuba: Editorial Ciencias Médicas.

Microsft (2018). *Información general de ASP.NET Core MVC.* Recuperado de https://docs.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-2.2#razor-view-engine

Microsft (2008). *ADO.NET Entity Framework.* Recuperado de https://docs.microsoft.com/es-es/previous-versions/visualstudio/visual-studio-2008/bb399572(v%3dvs.90)

Naylor, L. (2009). *ASP.NET MVC with Entity Framework and CSS*. EEUU: Apress.

Muñoz, M. (2018). *Programador Jr. De Aplicaciones ASP.NET MVC: Manual de estudiante.* Independently published

Rincón, L. (2016). *Desarrollo y servicios web* .Recuperado el 22 de octubre de 2018, de http://escher.puj.edu.co:8000/lfrincon/lfrp-web/raw/48c8c9ec795016a887565e797f4b7252ba4bf62c/Material/Clases/Sesion29.%20Framework%20.NET.pdf

Porras, E. (2011). *La Metodología Ágil y Formal ICONIX para el Desarrollo de Software: Teoría y Práctica.* Ayacucho, Perú: Ami Ayacucho.

Rosenberg, D., Stephens, M. (2007). *Use Case Driven Object Modeling with UML: Teory and Practice (1° Ed.).* EEUU: Apress.

ACTA DE SUSTENTACIÓN DE TESIS TITULADO:

"ADAPTABILIDAD DEL PATRON MVC DEL FRAMEWORK .NET IA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA AGIL Y FORMAL ICONIX. CASO DE ESTUDIO: REGISTRO DE ESTUDIO AMBIENTALES DEL DREMA",

EN EL AUDITORIO DE LA FACUITAD DE INGENIERIA DE MINAS GEOLOGIA Y CIVIL (H-205), SIENDO LAS 10:30AM DEL DÍA 04 DE ABRIL DEL 2019, SE REUNIERON EN EL AUDITORIO (H-2) LOS MIEMBROS DEL JURADO DE TESIS, SEGUN RESOLUCIÓN DECANAL N° 069 - 2019-FIMGC-D. DE FECHA 01 DE ABRIL DEL 2019, INTEGRADO POR:

DR. Ing. JAIME A. HUAMAN MONTES (PRESIDENTE)

Msc. Ing EFRAIN E. PORRAS FLORES (MIEMBRO)

Mg. Ing. HUBNER SANAMPA PATILLA (MIEMBRO),

EJERCIENDO COMO ASESOR EL Mg. ING. HUBNER SANAMPA PATILLA,

EL ACTO DE SUSTENTACIÓN, CON EL QUORUM RESPECTIVO SE DIO POR INICIO A HORAS 10:30 AM, DE LA MAÑANA CON PARTICIPACIÓN DEL SR. PRESIDENTE DR. Ing. JAIME HUAMAN MONTES, QUIEN INVITO AL SECRETARIO DOCENTE A DAR LECTURA A LA RESOLUCIÓN DECANAL N° 069-2019-FIMGC-D. DE FECHA 01 DE ABRIL DEL 2019, QUE CONSIDERA EL ACTO DE SUSTENTACIÓN TITULADO: "ADAPTABILIDAD DEL PATRON MVC DEL FRAMEWORK.NET IA ARQUITECTURA TÉCNICA DELA METODOLOGÍA AGIL Y FORMAL ICONIX. CASO DE ESTUDIO: REGISTRO DE ESTUDIO AMBIENTALES DEL DREMA", PRESENTADO POR EL BACHILLER SAMUEL KALIP YARCURI PAREDES,

LUEGO EL PRESIDENTE INVITA AL SR. SAMUEL K. YARCURI PAREDES, A DAR INICIO CON EXPOSICIÓN POR UN TIEMPO DE 40 MINUTOS, PARA LO CUAL CUENTA CON UN PROYECTOR MULTIMEDIA PARA PARTICIPAR.

CULMINADA LA HORA DISPUESTA PARA LA PRESENTACIÓN DE SU INFORME, EL PRESIDENTE INVITA A LOS JURADOS A REALIZAR LAS PREGUNTAS Y OBSERVACIONES EN EL ORDEN SIGUIENTE,

MSC Ing. EFRAIN E. PORRAS FLORES

Mg. Ing. HUBNER JANAMPA PATILLA

DR. Ing. JAIME A. HUAMAN MONTES

CONCLUIDA LA RUEDA DE PREGUNTAS Y OBSERVACIONES EL PRESIDENTE DEL JURADO INVITA AL SEÑOR SUSTENTANTE Y PUBLICO A RETIRARSE DEL AMBIENTE PARA PROCEDER LA CALIFICACIÓN POR CADA UNO DE LOS MIEMBROS DEL JURADO DE ACUERDO A LOS SIGUIENTES CRITERIOS

A: PRESENTACION DEL TRABAJO

B: METODOLOGIA Y APORTE TECNICO CIENTIFICO (CONTENIDO)

C: EXPOSICIÓN

D: RESPUESTAS A PREGUNTAS

LA CALIFICACION SE REALIZA EN FORMA PRIVADA POR CADA UNO DE LOS MIEMBROS DEL JURADO DE ACUERDO AL SIGUIENTE

DETALLE	A	B	C	D	PROM
1 DR. Ing. JAIME A. HUAMAN MONTES	13	13	13	13	13
2 MSC Ing. EFRAIN E. PORRAS FLORES	14	13	12	11	13
3 Mg. Ing. HUBNER JANAMPA PATILLA	15	14	14	14	14

PROMEDIO FINAL: 13

LUEGO EL PRESIDENTE DEL JURADO CALIFICADOR POR INTERMEDIO DEL SECRETARIO DOCENTE, INVITA AL SR. SAMUEL KAUP YURCURI PAREDES, Y AL PÚBLICO EN GENERAL A REINGRESAR AL AMBIENTE PARA PODER DAR RESULTADO DE LA CALIFICACIÓN CON LA NOTA PROMEDIO DE TRECE (13)

FINALMENTE, EL PRESIDENTE DEL JURADO CALIFICADOR EL DR. JAIME HUAMAN MONTES, MANIFIESTA QUE A PARTIR DE LA FECHA, LA UNIVERSIDAD NACIONAL SAN CRISTOBAL DE HUAMANGA Y LA FACULTAD DE INGENIERIA MINAS GEOLOGIA Y CIVIL CUENTA CUENTA CON UN NUEVO INGENIERO DE SISTEMAS DESEANDOLE EXITO, EN SU VIDA PROFESIONAL

EN CUANTO EL INFORME DE TESIS SE ARGUIVA LAS OBSERVACIONES Y CORRECCIONES RESULTAS, EL SUSTENTANTE DEBERA EDITAR UN TOTAL DE 05 EJEMPLARES CON LOS RESPECTIVOS CD'S, PAR IMPLEMENTAR LA BIBLIOTECA Y AULA VIRTUAL.

Dr. Ing. Jaime A. HUAMAN MONTES
PRESIDENTE

Mg. Ing. Hadmer JANAMPA PATILLA
MIEMBRO ASESOR

MSc. Ing. Efrain E. PORRAS FLORES
MIEMBRO

Ing. Christian LEZAMA CUELLAR
SECRETARIO DOCENTE

BACH. Samuel K. YARCURI PAREDES
SUSTENTANTE

Del acuerdo constatado en el Acta, levantado el 04 de abril del 2019 en la Sustentación de tesis presentado por el Bachiller en Ingeniería de Sistemas Sr. Samuel Kalip YARCURI PAREDES, Tesis Titulado: "**ADAPTABILIDAD DEL PATRON MVC DEL FRAMEWORK .NET A LA ARQUITECTURA TÉCNICA DE LA METODOLOGÍA ÁGIL Y FORMAL ICONIX. CASO DE ESTUDIO: REGISTRO DE ESTUDIOS AMBIENTALES DEL DREMA**", fue calificado con nota de TRECE(13), por lo que se da la respectiva **APROBACIÓN.**

RECOMENDADO : 17 DE ENERO DEL 2019

APROBADO : 04 DE ABRIL DEL 2019

Dr. Ing. Jaime A. HUAMÁN MONTES
Presidente

MSc. Ing. Efrain E. PORRAS FLORES
Miembro

Mg. Ing. Hubner Janampa Patilla
Presidente

Ing. Christian LEZAMA CUELLAR
Secretario Docente