ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO PARA LA GESTIÓN Y SEGUIMIENTO DE COMPROMISOS DE CENACE

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

MATÍAS ROMERO ARMAS

DIRECTOR: DAVID RAUL MEJÍA NAVARRETE, M.Sc.

Quito, 23 agosto 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Matías Romero Armas, bajo mi supervisión.

M.Sc. Raúl David Mejía Navarrete
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Matías Romero Armas, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

Matías Romero Armas

DEDICATORIA

A mis padres Yolanda y Rómulo, que gracias a su amor y apoyo diario me ayudaron a ser una mejor persona y crecer académicamente. Muchas gracias por todas las enseñanzas y valores que he aprendido gracias a ustedes y sobretodo con su ejemplo.

A mis hermanos Silvana y David, que son mi ejemplo a seguir debido a su dedicación y esfuerzo. Gracias por siempre estar dispuestos a ayudarme en cualquier ámbito que sea necesario.

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios por darme las fuerzas y valentía de seguir uno de mis sueños y poder culminar una meta más en mi vida.

A mis padres y hermanos, que desde el inicio estuvieron ayudándome en todo lo que necesitaba y siempre estaban pendientes de mí, más de lo que yo he estado pendiente de ellos, los amo mucho.

A mis tíos y primos, por sus buenos deseos en el transcurso de mi carrera universitaria y por todos los momentos de risas, juegos y diversión en familia que amenizaron mi época de la universidad.

A mis amigos de la universidad con los cuales compartí buenos y malos momentos, pero siempre nos hemos estado apoyándonos desde el día en que nos conocimos: Cinthya O., Jorge C., María Emilia V., Samuel J., Steven M., Adrián M., Jhon J., Álvaro T., Elizabeth Y., Jhostin C., Marco O., etc.

A mis amigos de Redes: Zahid C., Carlos V., Vanessa E. Marcelo C., Joseph B., Bryan S., Steban M. con los cuales tuve muchas clases, trabajos grupales, paseos, giras y siempre me sacaban de quicio, pero sin ustedes mi estadía en la universidad no sería la misma.

A mis amigos que me apoyaron y estuvieron presentes en momentos muy difíciles: Jordy G., Melvin C., Gaby C., Leonela B. Luis V, Doris A.

A todos los docentes y personal administrativo de la Escuela Politécnica Nacional por impartir sus conocimientos durante mi carrera universitarias.

A mi director, Ing. David Mejía por su tiempo y paciencia durante el desarrollo de mi Trabajo de Titulación y por las enseñanzas en clases tanto para la vida profesional como personal.

Al personal de CENACE por brindarme las facilidades y la información necesaria para el desarrollo de manera correcta de este Trabajo de Titulación.

ÍNDICE DE CONTENIDO

RESUMEN

En el presente Trabajo de Titulación se presenta el diseño e implementación de un prototipo para la gestión y seguimiento de compromisos de CENACE. El uso del software permitirá agilizar las búsquedas de los compromisos y realizar un seguimiento de los mismos, lo cual genera beneficios para el área de análisis de operaciones al momento de realizar reportes. El prototipo estará conformado por un servicio WCF (Windows *Communication Foundation*) basado en SOAP (*Simple Object Access Protocol*) y por un gestor cliente que consumirá dicho servicio a través de una aplicación de escritorio en Windows.

En el primer capítulo se presenta los objetivos generales, específicos y el alcance de este Trabajo de Titulación; así como también se describen los fundamentos teóricos utilizados para el diseño e implementación del prototipo.

En el segundo capítulo se muestra las fases del prototipo: La fase de diseño a través del levantamiento de requerimientos y diagramas relacional, casos de uso, actividades, secuencia, clases y generación de *sketches*; y la fase de implementación de cada una de las capas del prototipo.

En el tercer capítulo se detalla los resultados obtenidos de las pruebas de funcionamiento y pruebas usuario aplicadas al prototipo desarrollado.

En el cuarto capítulo se presenta las conclusiones y recomendaciones obtenidas al finalizar este Trabajo de Titulación.

Finalmente, se encuentra los anexos. Entre los anexos se muestra las encuestas para el levantamiento de requerimientos, detalles y ampliación de los diferentes diagramas realizados en la etapa de diseño, códigos del prototipo desarrollado, manual de usuario y encuestas de satisfacción.

PALABRAS CLAVE: gestión de compromisos, seguimiento de compromisos, servicio WCF basado en SOAP, IIS, sistema distribuido, arquitectura basada en 4 capas.

ABSTRACT

This degree work describes the design and implementation of a prototype for the management and follow-up of CENACE commitments. The use of the software will speed up the searches for commitments and track them, which generates benefits for the analysis area when making reports. The prototype will be made up of a WCF service based on SOAP and a client manager that will consume the service through a Windows desktop application.

The first chapter presents the general and specific objectives and the scope of this degree project; as well as the theoretical foundations used for the design and implementation of the prototype.

The second chapter shows the phases of the prototype: The design phase through the survey of requirements and relational, use cases, activities, sequence, class diagrams and generation of sketches; and the implementation phase of each of the prototype layers.

The third chapter details the results obtained from the performance tests and user tests applied to the developed prototype.

The fourth chapter presents the conclusions and recommendations obtained at the end of this degree project.

Finally, there are the annexes. Among the annexes we find the surveys of requirements, details and extension of the different diagrams made in the design stage, codes of the developed prototype, user manual and satisfaction surveys.

KEYWORDS: commitment management, commitment tracking, WCF service based on SOAP, IIS, distributed system, 4 tier based architecture.

1. INTRODUCCIÓN

En [1] se indica: “Al menos el 40% de todas las empresas morirán en los próximos 10 años, si no descubren cómo cambiar toda su empresa para adaptarse a las nuevas tecnologías”, por ello que se ha visto la necesidad de que ciertos procesos de las empresas deban adaptarse a las nuevas tecnologías para brindar mayor rentabilidad, competitividad y eficiencia.

“El Operador Nacional de Electricidad CENACE es una entidad estratégica del sector eléctrico ecuatoriano, el cual opera y administra el funcionamiento técnico y comercial del Sistema Nacional Interconectado” [2]. Como parte de esta entidad se encuentra la Subgerencia Nacional de Análisis de Operaciones, la cual no cuenta con un sistema informático que le permita gestionar la información de los compromisos generados en las distintas reuniones del área.

Por ello, a través de este Trabajo de Titulación se plantea desarrollar un prototipo para la gestión y seguimiento de los compromisos de CENACE, el cual se espera permita agilizar las búsquedas de los compromisos y sus principales métricas, lo cual genera beneficios para el área de análisis de operaciones, sobre todo al momento de realizar reportes.

En este capítulo se presentarán los objetivos generales y específicos, así como el alcance del Trabajo de Titulación, también se presentará un marco teórico que sirve como base para el desarrollado de este Trabajo de Titulación, en donde se tratará la arquitectura basada en capas, la base de datos, el método de acceso a los datos DAO[1] para separar la capa lógica de negocio de la capa de los datos, los servicios WCF[2], los servidores IIS[3] para alojar los servicios web y finalmente se explicará la metodología de desarrollo ágil Kanban.

1.1 OBJETIVOS

El objetivo general del este Trabajo de Titulación es desarrollar un prototipo de sistema para la gestión y seguimiento de compromisos de CENACE.

[1] DAO (*Data Access Object*): es un componente que otorga una interfaz entre la aplicación y uno o varios dispositivos de almacenamiento de datos.

[2] WCF (Windows *Communication Foundation*): permite simplificar la programación de red al momento de crear servicios.

[3] IIS (*Internet Information Services*): es un servidor web para Microsoft Windows.

Los objetivos específicos son los siguientes:

- Analizar las herramientas necesarias para el desarrollo del prototipo propuesto.
- Diseñar el prototipo con base en los requerimientos levantados para la empresa.
- Implementar el prototipo de acuerdo al análisis de requerimientos.
- Analizar los resultados de las pruebas de funcionalidad.

1.2 ALCANCE

En este Trabajo de Titulación se plantea desarrollar un prototipo de sistema distribuido que permita al usuario llevar un control de seguimiento de los compromisos propuestos en las distintas reuniones de la entidad en el área de Análisis de Operaciones. El prototipo de sistema distribuido estará formado por un servicio web desarrollado con WCF y será consumido por el usuario a través de una aplicación de escritorio en Windows.

La aplicación de escritorio contará con varias ventanas que permitirá al usuario poder registrar los compromisos, que contendrá la información como: tipo de reunión a la que está asociada, detalle del compromiso, fecha plazo, uno o varios responsables debidamente identificados con el nombre, correo electrónico y área de trabajo. Además, los usuarios podrán realizar un seguimiento de los compromisos, en donde pueden visualizar los compromisos próximos a caducar, podrán colocar el porcentaje de cumplimiento de un compromiso y registrar observaciones sobre los mismos; también se podrá realizar el CRUD (crear, visualizar, actualizar y eliminar) de los compromisos, del seguimiento de compromisos y de los responsables.

En el presente Trabajo de Titulación se utilizará una arquitectura de cuatro capas (Datos, Acceso a Datos, Negocio y Presentación) como se muestra en la Figura 1.1 y será desarrollado con base en la metodología Kanban. Para concluir, se generará un producto final demostrable.

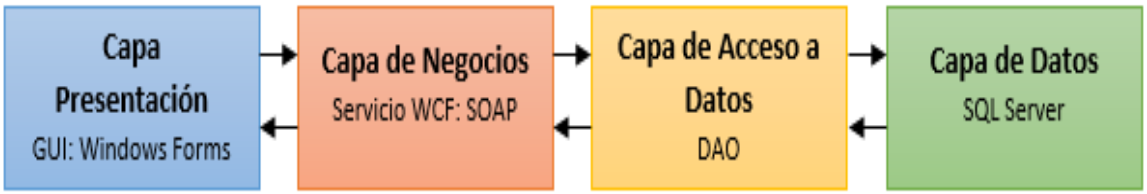

Figura 1.1 Arquitectura de cuatro capas.

1.3 MARCO TEÓRICO

En esta sección se indicará el marco teórico y las distintas tecnologías utilizadas para el desarrollo de este Trabajo de Titulación.

1.3.1 ARQUITECTURA BASADA EN CAPAS

La arquitectura basada en capas se basa principalmente en dividir la aplicación en un cierto número de capas y en cada capa definir una responsabilidad o rol específico de forma jerárquica, es decir las capas deben ser colocadas una encima de otra [3].

El dividir una aplicación en capas permite que cada capa cumpla un rol específico y tenga una interacción con la capa superior e inferior, permitiendo que las aplicaciones sean más fáciles de entender, probar y mantener [4].

En este Trabajo de Titulación se desarrollará un prototipo de aplicación con una arquitectura cliente-servidor y la funcionalidad será estructurada en 4 capas, como se muestra en la Figura 1.2.

La capa de presentación es la capa más cercana al usuario y por lo cual su responsabilidad es de interactuar con el mismo. Esta capa suele desarrollarse para ser ejecutada desde un navegador web, una aplicación de escritorio o una interfaz gráfica de usuario. Para la capa presentación, en este Trabajo de Titulación se desarrollará una aplicación de escritorio para el sistema operativo Windows [5].

La capa de lógica de negocio, es la capa intermedia entre la capa de presentación y la capa de acceso a datos, esta capa se encarga de cumplir las reglas de negocio y los requerimientos que el usuario necesita de la aplicación. Esta capa incluye la lógica del funcionamiento de la aplicación, la administración de datos, entre otras.

La capa de acceso a datos proporciona un mecanismo para que la capa de negocio interactúe con los datos, para que el usuario pueda agregar información sobre nuevas reuniones y compromisos asociados a este y que con ello pueda gestionar la información accediendo a esta, modificándola o eliminándola; además de dar un seguimiento a los compromisos almacenados en la base de datos.

Finalmente, la capa de datos gestiona la fuente de información. En ella se encuentran las tablas y datos que serán administrados por la aplicación. En esta capa se puede emplear

un sistema de gestión de base de datos relacionales como SQL[4] Server, PostgreSQL, MySQL, Oracle, entre otros [6]. Para este Trabajo de Titulación se va a utilizar Microsoft SQL Server Management Studio, el cual es un entorno integrado de gestión de base de datos.

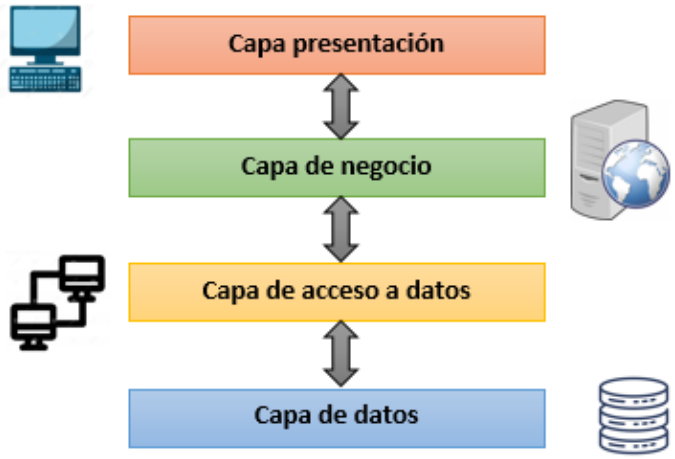

Figura 1.2 Arquitectura basada en 4 capas.

1.3.2 BASE DE DATOS

Un sistema de base de datos es un conjunto de archivos relacionados entre sí y de programas que permite a los usuarios poder acceder y modificar estos archivos.

Una base de datos es el método más usado para el almacenamiento de datos. El objetivo principal de un SGBD[5] es almacenar y recuperar la información de manera más práctica y eficiente. En la actualidad, las bases de datos son el pilar de muchas empresas ya que en ella guardan información relevante de sus procesos [7].

1.3.2.1 Modelo de datos

El modelo de datos es una herramienta que permite describir los datos, las relaciones, la semántica y las restricciones de consistencia. Un modelo de datos es una representación simple de las estructuras de datos [8].

Los componentes básicos de un modelo de datos son:

[4] SQL (*Structured Query Languaje*): es un lenguaje diseñado para administrar y recuperar la información de un sistema de gestión de base de datos.
[5] SGBD (*Sistema Gestor de Base de Datos*): es un conjunto de programas que permiten gestionar y administrar la información que se encuentre en una base de datos.

Entidad: es un objeto (persona, evento, lugar) sobre el cual se almacenarán sus datos.

Atributo: es una característica de una entidad.

Relaciones: son las asociaciones entre entidades.

Restricciones: son reglas colocadas a los datos, lo que permite la integridad de los datos almacenados.

Existen dos tipos de modelos de datos, el modelo entidad-relación[6] y el modelo relacional. En este Trabajo de Titulación se utilizará el modelo relacional.

1.3.2.2 Modelo relacional

En el modelo relacional, los datos y las relaciones entre ellos se representan por medio de un grupo de tablas. El fundamento de este modelo son las relaciones entre entidades. Cada tabla está compuesta por varias columnas, que representa los atributos de la entidad y con filas en donde se almacenan los datos [9].

Para representar una relación entre las tablas se utilizan las claves, las cuales pueden ser primarias o secundarias (foráneas). “La clave primaria de una entidad es una columna que permite identificar de manera única a una fila; mientras que la clave secundaria es una columna que hace referencia a una clave primaria de otra tabla previamente creada” [9].

Como se puede observar en la Figura 1.3, en el modelo relacional los datos se almacenan en filas.

Por ejemplo, en la tabla `compromisos – reunión`, se observa que los datos de la primera fila, tiene como `id-compromiso` el valor 1, y esta es una cave primaria, en el atributo `responsable-compromiso` se indica Jonathan Álvarez, en `detalle-compromiso` se indica “realizar la actividad 1”, cuyo `plazo` es hasta el 31-12-2021 y la reunión asociada (`id-reunion`) es a la reunión cuya clave primaria tiene valor 1, la misma que se encuentra definida en la tabla `reunión`, por lo que la reunión tiene como `fecha` 01-04-2021; mientras que los datos de la sexta fila, tiene como `id-compromiso` el valor 6, y esta es una cave primaria, en el atributo `responsable-compromiso` se indica Raúl Gómez, en `detalle-compromiso` se indica “realizar la actividad 6”, cuyo `plazo` es hasta

[6] Modelo entidad-relación: es una herramienta para la representación de datos de una base de datos utilizando únicamente como componentes del mismo las entidades y relaciones entre ellas, mas no los atributos de las entidades.

el 21-10-2021 y la reunión asociada (`id-reunion`) es a la reunión cuya clave primaria tiene valor 3, la misma que se encuentra definida en la tabla `reunión`, por lo que la reunión tiene como `fecha` 11-07-2021;

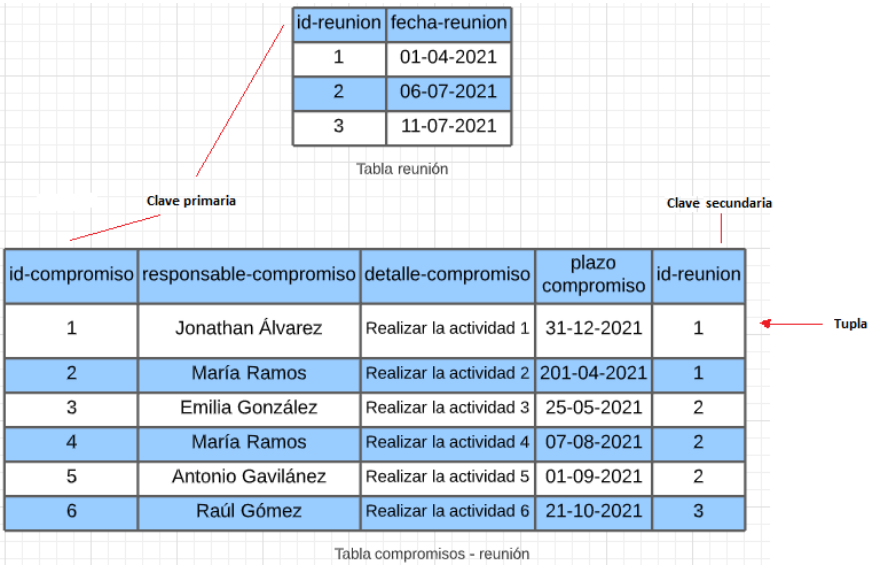

Figura 1.3 Modelo relacional con atributos

1.3.3 LENGUAJE DE CONSULTA DE BASES DE DATOS

Para poder comunicarse con el SGBD, el usuario debe usar un lenguaje de consulta de base de datos. El lenguaje de consulta de base de datos permite al usuario la definición y manipulación de los datos. Existen muchos lenguajes de consulta de base de datos, pero el más utilizado para base de datos relacionales es el lenguaje SQL [10].

SQL tiene tres tipos de instrucciones, el tipo DDL[7], lo cual se utiliza para la descripción de la base de datos, como por ejemplo la creación de tablas, el tipo DML[8], lo cual se utiliza para la manipulación de datos, como por ejemplo para la recuperación de información almacenada, la inserción de información nueva a la base de datos, la eliminación o la modificación de la información de la base de datos; y el tipo DCL[9] los cuales se emplean para controlar el acceso a la información almacenada en la base de datos [9].

[7] DDL (*Data Definition Lenguage*): es un subconjunto del lenguaje SQL que se usa para describir los datos y las relaciones entre ellos.

[8] DML (*Data Management Language*): es parte del lenguaje SQL que permite a los usuarios realizar tareas de consulta o manipulación de la información almacenada en una base de datos.

[9] DCL (*Data Control Language*): es una porción del lenguaje de SQL que permiten controlar el acceso a la información almacenada en la base de datos

1.3.3.1 Comandos

SQL posee instrucciones del tipo DDL, DML y DCL, denominados comandos. Los comandos DDL permiten crear bases de datos y estructuras para el almacenamiento de datos; los comandos DML permiten insertar, consultar, eliminar y modificar los datos de una base de datos; mientras que los comandos DCL se emplean para definir permisos.

En la Tabla 1.1, Tabla 1.2 y Tabla 1.3 se presenta un conjunto de comandos DDL, DML y DCL respectivamente.

Tabla 1.1 Comandos DDL de SQL

Comando	Descripción
CREATE	Crea nueva base de datos, tablas y vistas
DROP	Elimina tablas
ALTER	Modifica tablas o campos

Tabla 1.2 Comandos DML de SQL

Comando	Descripción
SELECT	Consultar registro según un criterio
INSERT	Insertar datos a una tabla
UPDATE	Modificar los campos y registros
DELETE	Eliminar registros

Tabla 1.3 Comandos DCL de SQL

Comando	Descripción
GRANT	Otorgar permisos
REVOKE	Eliminar permisos

1.3.3.2 Cláusulas

Las cláusulas son instrucciones usadas como condicionales para definir un grupo específico de datos. En la Tabla 1.4 se presentan ejemplos de cláusulas.

1.3.3.3 Operaciones

Son operadores lógicos y de comparación, en la Tabla 1.5 se presentan ejemplos.

Tabla 1.4 Cláusulas de SQL

Comando	Descripción
`FROM`	Especifica una tabla
`GROUP BY`	Agrupa los registros
`HAVING`	Especifica una condición que cumpla una selección de registros
`ORDER BY`	Ordena registros
`WHERE`	Condición que deben cumplir los registros dentro de una tabla específica (cláusula `FROM`)

Tabla 1.5 Operaciones de SQL

Comando	Descripción
`AND`	Operador conjunción
`OR`	Operador disyunción
`BETWEEN`	Intervalo
`< , > , =`	Menor, mayor e igual

1.3.4 MÉTODO DE ACCESO A DATOS

La información es algo fundamental en las aplicaciones. La información se puede almacenar en muchos formatos, como por ejemplo una base de datos que se accede mediante lenguaje SQL, archivos etiquetados como XML[10] o JSON[11], base de datos para dispositivos móviles, entre otros. y esta variedad de formatos no debe afectar a las funcionalidades de la aplicación [11].

DAO (*Data Access Object*) es un patrón el cual ofrece un objeto de acceso a los datos que actúa como una pasarela entre los datos de la capa de la base de datos y la capa de la lógica de negocio. [12]

La capa de acceso a datos encapsula la lógica necesaria para que los datos se puedan transferir desde la capa de datos hacia la capa de lógica de negocio o viceversa y para ello debe tener métodos con los cuales se puedan realizar operaciones como añadir, modificar, buscar y borrar elementos [11].

[10] XML (*Extensible Markup Language*): es un lenguaje de marcado que define reglas para la codificación de documentos.

[11] JSON (*JavaScript Object Notation*), es un formato de intercambio de datos.

Como se observa en la Figura 1.4 la estructura del patrón DAO consta de cuatro elementos [12]:

Objeto de Negocio: Objeto de la capa de negocio que necesita acceder a la fuente de datos.

Objeto DAO: Se encarga de encapsular el acceso a la fuente de datos y para ello crea y utiliza un objeto de transferencia como contenedor para mover los datos.

Objeto de Transferencia: Objeto de contenedor para la movilización de la información desde la capa de datos hacia la capa lógica de negocio y viceversa.

Fuente de datos: Lugar donde esta almacenada la información.

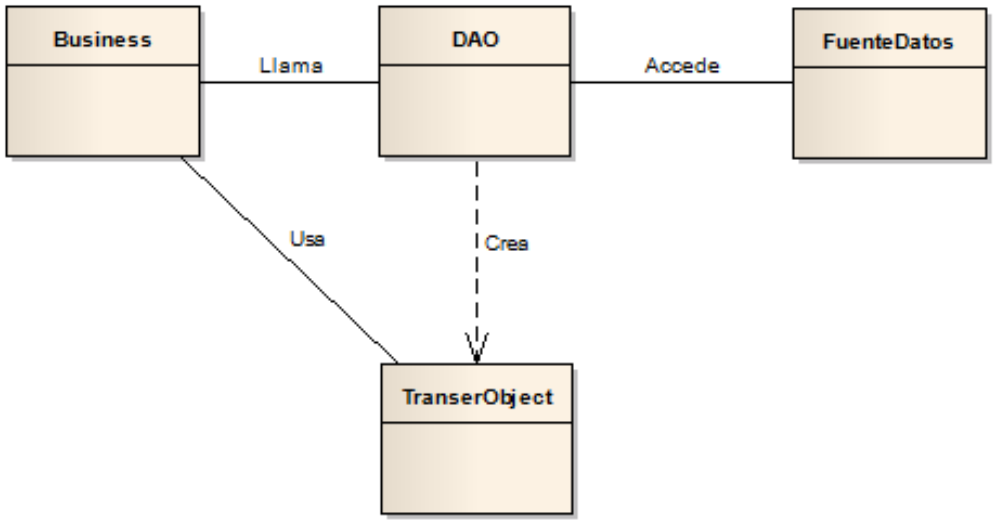

Figura 1.4 Estructura del patrón DAO

1.3.5 SERVICIOS WCF

Windows *Communication Foundation* (WCF) es un *framework*[12] para crear aplicaciones orientadas a servicios. WCF permite enviar datos como mensajes asincrónicos desde un punto de conexión de un servicio a otro [13].

WCF hace que el desarrollo de *endpoints*[13] sea mucho más fácil, por lo que "WCF está diseñado para ofrecer un enfoque manejable para crear servicios web y clientes de servicios web" [14].

[12] *Framework:* es una herramienta que se utiliza para agilizar el desarrollo de un software.

[13] *Endpoint:* es un dispositivo informático que se comunica con una red a la que está conectado, como por ejemplo: computadoras, celulares, estaciones de trabajo, servidores, etc.

1.3.5.1 Arquitectura WCF

La Figura 1.5 presenta las capas principales de la arquitectura de WCF.

En la capa aplicación se encuentra todas las funcionalidades que el servicio ofrece, en esta capa se implementan todos los métodos definidos dentro de la capa contratos, en un lenguaje de programación como C# [14].

En la capa contratos se definen varios aspectos del sistema. Dentro de esta capa se tiene cuatro tipos de contratos. El contrato de datos indica todos los datos que un servicio puede crear o consumir; el contrato de mensaje permite controlar partes específicas del mensaje utilizado por SOAP[14]; el contrato de servicio indica las firmas de los métodos del servicio y se crea como una interfaz en uno de los lenguajes compatibles como Visual Basic o C#, y las políticas y vinculaciones indican las condiciones necesarias para comunicarse con un servicio [15].

En la capa tiempo de ejecución del servicio se controla el comportamiento que ocurre durante la ejecución de un servicio. “Esta capa ayuda a limitar cuantos mensajes se procesan en tiempo de ejecución si la demanda del servicio crece, además controla el comportamiento del servicio cuando ocurren errores, etc.” [15].

La capa mensajería se conforma por canales, los cuales son componentes que procesan un mensaje usados para acceder al servicio [15].

La capa hospedaje y activación proporciona las opciones de arranque y alojamiento del servicio. En su forma final, el servicio es un programa el cual debe ejecutarse en un ejecutable administrado por un agente externo como IIS o WAS[15] [15].

Figura 1.5 Arquitectura WCF

[14] SOAP *(Simple Object Access Protocol):* es un protocolo que nos permite realizar servicios web sin estado.

[15] WAS (*WebSphere Application Server*): es un servidor de aplicaciones de software.

1.3.6 INTERNET INFORMATION SERVICES

IIS es un conjunto de servicios que transforman un sistema operativo Microsoft Windows en un servidor capaz de ofrecer servicios web, FTP[16], SMTP[17], entre otros. IIS es apto tanto como servidor web en Internet como para una intranet, siendo más utilizado dentro de una intranet [16]. Un servidor web permite almacenar páginas web a las que se accede usando un navegador web.

1.3.7 METODOLOGÍA DE DESARROLLO KANBAN

La metodología de desarrollo Kanban es una metodología ágil y su objetivo es gestionar la realización de tareas hasta su finalización, es decir divide el trabajo en tareas pequeñas. Esta metodología de gestión de proyectos permite visualizar el flujo de trabajo por medio de tarjetas [13].

Kanban se basa en los principios fundamental los cuales son: visualizar lo que se va a realizar el día de hoy, es decir ver todas las tareas que se desarrollaran en un día; limitar la cantidad de trabajo en progreso, para que no se acumulen tareas y el flujo de tareas continúe; y, en el mejorar el flujo, es decir que cuando una tarea se ha culminado, se procede a realizar la siguiente tarea prioritaria [13].

Las tareas se presentan en el tablero Kanban, como se muestra en la Figura 1.6.

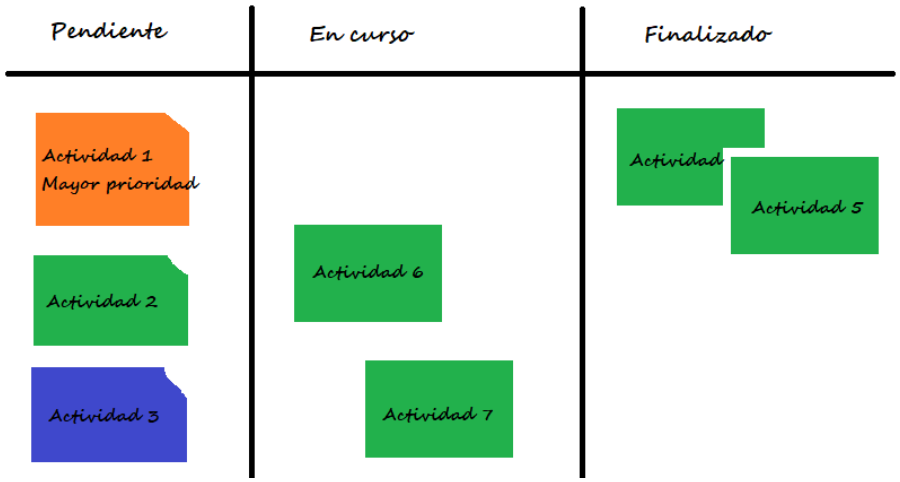

Figura 1.6 Tablero Kanban

[16] FTP (*File Transfer Protocol*): es un protocolo que permite la transferencia de archivos entre un dispositivo a otro.

[17] SMTP (*Simple Mail Transfer Protocol*): es un protocolo de comunicación de los servidores de correos para el envío y recepción de correos electrónicos.

Este tablero es un sistema de control visual del trabajo e indica como fluye el trabajo a través de las diferentes etapas del proceso de desarrollo. El tablero consiste en una secuencia de columnas, cada columna del tablero es una etapa del progreso, mientras que en cada fila se colocan las diferentes actividades para llevar a cabo el proyecto, cada tarea es una etiqueta del tablero [17].

A medida que el trabajo avanza a lo largo del desarrollo, las tareas pasan por varios estados desde que la primera etapa (primera columna) hasta la última etapa (última columna).

Las tarjetas se pueden colocar según su nivel de prioridad en las diferentes columnas, poniendo las de mayor prioridad en la cima de la columna, de manera que cuando se finalice una tarea la siguiente en ser desarrollada sea la de mayor prioridad.

Las tarjetas de tareas pueden incluir el nombre de la persona que está encargada de dicha actividad y una descripción de la tarea y su principal función es mostrar de manera más clara el trabajo a realizar.

Kanban prioriza el trabajo que se está realizando, es decir busca que primero se acaben las tareas en desarrollo antes de empezar con otra tarea, por ello para iniciar un proyecto se debe realizar un listado de todas las tareas y definir las prioridades de las mismas, de esta manera se busca tener una mejor eficiencia en el momento del desarrollo [17].

1.3.8 RELACIÓN CON TRABAJOS SIMILARES

En [18] se desarrolló un sistema prototipo para gestión y seguimiento de los Trabajos de Titulación en la carrera de Ingeniería Electrónica y Redes de información. Este sistema fue desarrollado como una aplicación web, haciendo uso de ASP.NET MVC[18] y potenciado sus principales funcionalidades con Bootstrap[19], JQuery[20] y AJAX[21]. La aplicación dispone de módulos para el registro de nuevos trabajos, el cumplimiento de requisitos y recordatorios sobre las tareas, así como la generación de gráficos con información del proceso.

[18] ASP.NET MVC: es un *framework* de aplicaciones web que utiliza el patrón MVC (*Modelo-Vista-Controlador*) para la creación de sitios web dinámicos de manera más simple.

[19] Bootstrap: es un *framework* utilizado para la creación de interfaces web.

[20] JQuery: es una librería de código abierto de JavaScript que simplifica la tarea de programación, en especial para el desarrollo de sitios web.

[21] AJAX (*Asynchronous JavaScript and XML*): es un conjunto de técnicas para el desarrollo web.

En [19] el objetivo era el desarrollo e implantación de un sistema que mejore el proceso de gestión de proyectos que se desarrollan en la Escuela Politécnica Nacional contando con módulo donde se realice un seguimiento de las actividades propuestas por los diferentes tutores. Para este sistema se utiliza la metodología RUP[22], que es una metodología de desarrollo que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo.

La principal diferencia con los trabajaos mencionados anteriormente es que ellos se enfocan en la gestión y seguimiento de proyectos de titulación de una Facultad en específica de la Escuela Politécnica Nacional, mientras que en este Trabajo de Titulación se planifica la gestión y seguimientos de compromisos enfocados a las características y necesidades del área de análisis de operaciones del CENACE.

Otra diferencia radica en la tecnología que se empleó en cada uno de ellos; el primero es una aplicación web basada en MVC y potenciada con Bootstrap, JQuery y AJAX, y el segundo es una aplicación web que usa JSF[23] como Framework, mientras que para este Trabajo de Titulación se plantea realizar un servicio WCF e implementar una aplicación de escritorio en Windows que consuma dicho servicio.

[22] RUP (*Rational Unified Process*): es una metodología que asigna de forma disciplinada tareas y responsabilidades en una empresa de desarrollo.

[23] JSF (*JavaServer Faces*): es un *framework* de interfaz de usuario del lado de servidor para aplicaciones web basadas en Java.

2. METODOLOGÍA

En este capítulo se detallará un resumen del diseño e implementación del prototipo, para lo cual se empezará por definir su arquitectura y se determinarán las funciones de las distintas capas que tendrá, es decir la capa de presentación, lógica de negocio, acceso a datos y base de datos, que permitan la gestión y seguimiento de compromisos de CENACE. Para el desarrollo del prototipo se empleará la metodología de desarrollo ágil Kanban.

Como parte de la etapa de diseño, se indicará la definición de los requerimientos, la creación de las historias de usuario, la elaboración del tablero Kanban, la generación del diagrama relacional de la base datos, el diagrama de caso de uso, de actividades, de secuencia, de clases y los *sketches* de las vistas.

Como parte de la etapa de implementación, se instalarán las herramientas para la elaboración del prototipo, se creará un sistema de control de versiones, se realizará la codificación de cada capa del prototipo y el alojamiento del servicio WCF en un servidor IIS.

2.1 DISEÑO

2.1.1 DEFINICIÓN DE REQUERIMINETOS

En esta sección se presentará la situación actual para la gestión de compromisos de CENACE. Así también se detallarán los requisitos funcionales y no funcionales del prototipo, los mismos que se definirán con base a entrevistas que serán realizadas a miembros de la subgerencia nacional de análisis de operaciones de CENACE.

2.1.1.1 Situación actual de la gestión de compromisos de CENACE

El Operador Nacional de Electricidad CENACE, es una entidad encargada del manejo técnico y económico de la energía en bloque, por lo cual es un órgano técnico estratégico adscrito al Ministerio de Energía y Recursos Naturales No Renovables del Ecuador. Entre sus principales tareas están: operar el Sistema Nacional Interconectado, administrar las transacciones de bloques energéticos, y ser responsable del abastecimiento continúo de energía eléctrica al mínimo costo posible [20].

La entidad cuenta con una dirección ejecutiva y cuatro gerencias. Las gerencias son: la gerencia nacional de planeamiento operativo, la gerencia nacional de transacciones

comerciales, la gerencia nacional de desarrollo técnico y la gerencia nacional de operaciones. En esta última se encuentra la subgerencia nacional de análisis de operaciones, la cual se encarga de la gestión de los compromisos de esta área.

La subgerencia nacional de análisis de operaciones gestiona la información referente a los compromisos mediante un archivo en formato XLS[24]. Al tener la información en un archivo este puede contribuir a generar errores al realizar consultas sobre las principales métricas almacenadas.

Con la finalidad de mejorar la gestión de información de los compromisos de CENACE, se plantea el diseño e implementación de un prototipo de sistema distribuido que permita realizar el seguimiento de compromisos y realizar búsquedas de los mismos.

2.1.1.2 Entrevistas

Para obtener los requerimientos se aplicó una entrevista a cinco miembros de la subgerencia nacional de análisis de operaciones de CENACE, quienes son los encargados de la gestión de los compromisos de CENACE. La entrevista realizada está conformada de once preguntas y el modelo se presenta en el ANEXO A

La pregunta número 1 permite conocer si en la entidad se cuenta con un software que permita gestionar la información de compromisos de CENACE. En la Figura 2.1 se muestran los resultados de dicha pregunta, en donde se puede observar que no poseen un software que realice dichas tareas.

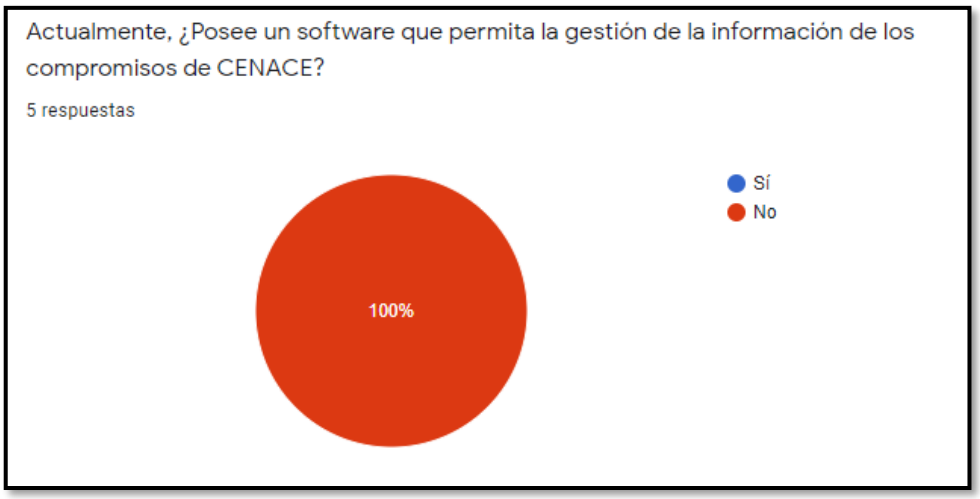

Figura 2.1 Resultados primera pregunta

[24] XLS: es la extensión de los archivos de Excel de Microsoft.

La pregunta número 2 busca conocer si dentro de la búsqueda de la información de los compromisos se han tenido problemas. En la Figura 2.2 se muestra que el 80% de los entrevistados han tenido algún problema al realizar búsquedas de la información.

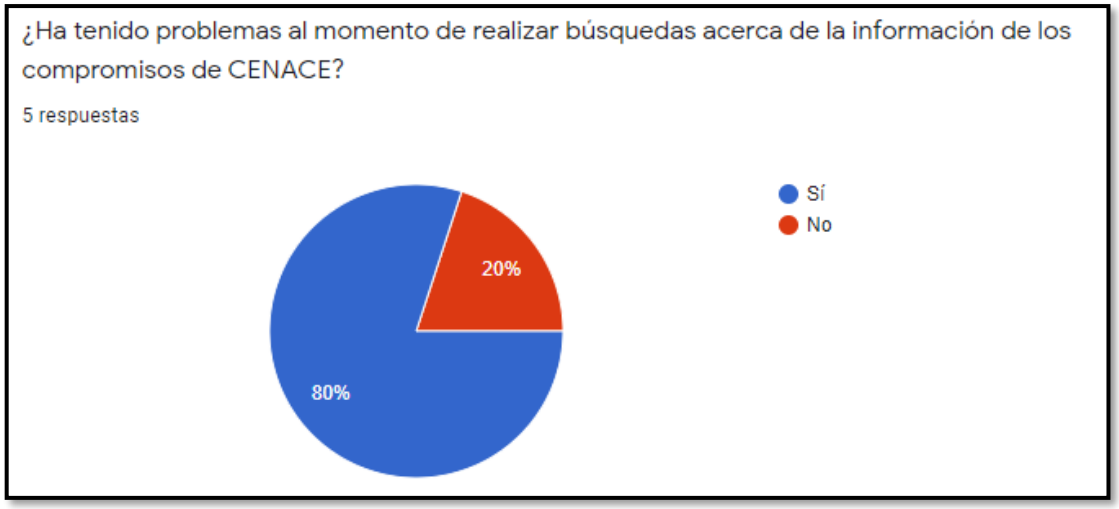

Figura 2.2 Resultados segunda pregunta

La pregunta número 3 fue contestado solo por las personas que tuvieron problemas en la gestión de compromisos y su objetivo fue conocer los principales problemas que han tenido al realizar las búsquedas de información. En la Figura 2.3 se muestra que el 100% de los entrevistados han tenido problemas con datos incompletos o que no tienen coherencia, el 50% indica que no encuentra la información referente a un compromiso, mientras que el 25% indica que su principal problema es la existencia de datos duplicados.

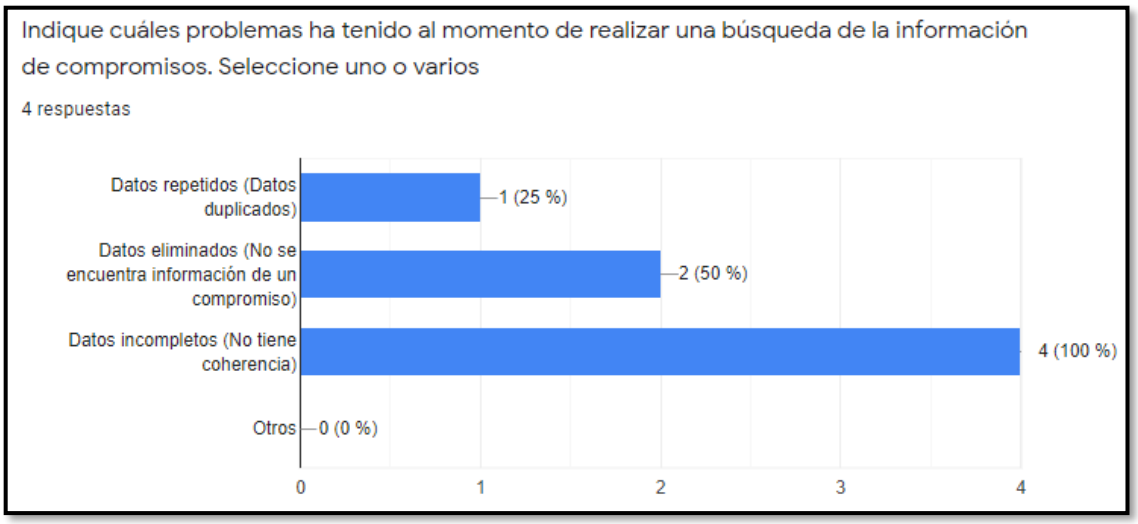

Figura 2.3 Resultados tercera pregunta

La pregunta número 4 busca conocer si es necesario el desarrollo de un software que permita la gestión de la información sobre los compromisos de CENACE. Como se observa en la Figura 2.4 el 100% de los entrevistados están de acuerdo en el desarrollo de un software.

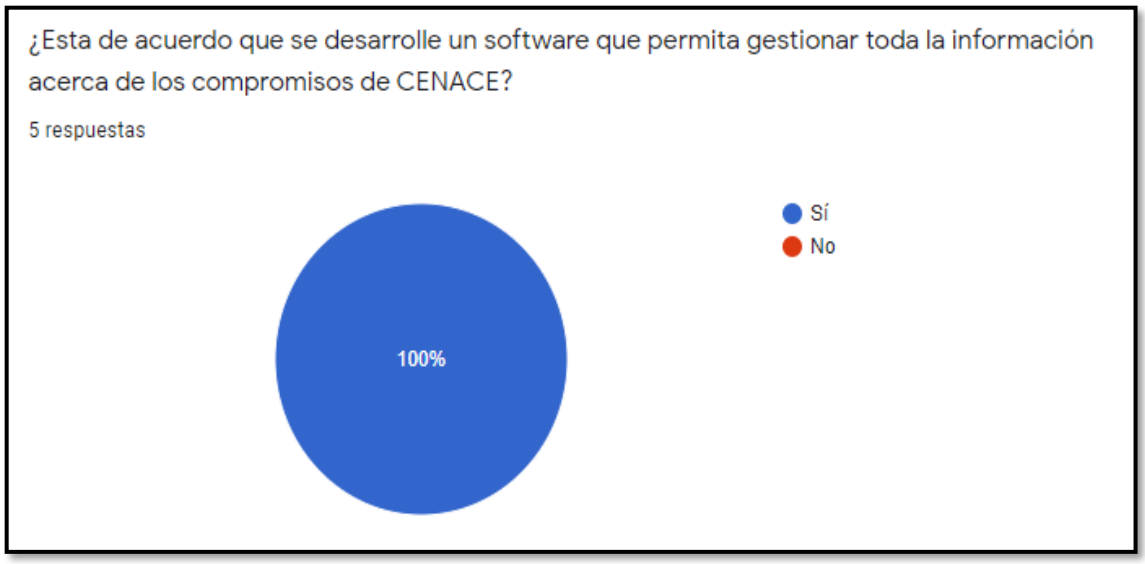

Figura 2.4 Resultados cuarta pregunta

En la pregunta número 5 se busca conocer qué tipo de software desean para la gestión de la información. Los resultados se muestran en la Figura 2.5 donde el 80% desean una aplicación de escritorio y un 20% desea una aplicación web. No hay interés en una aplicación para dispositivos móviles.

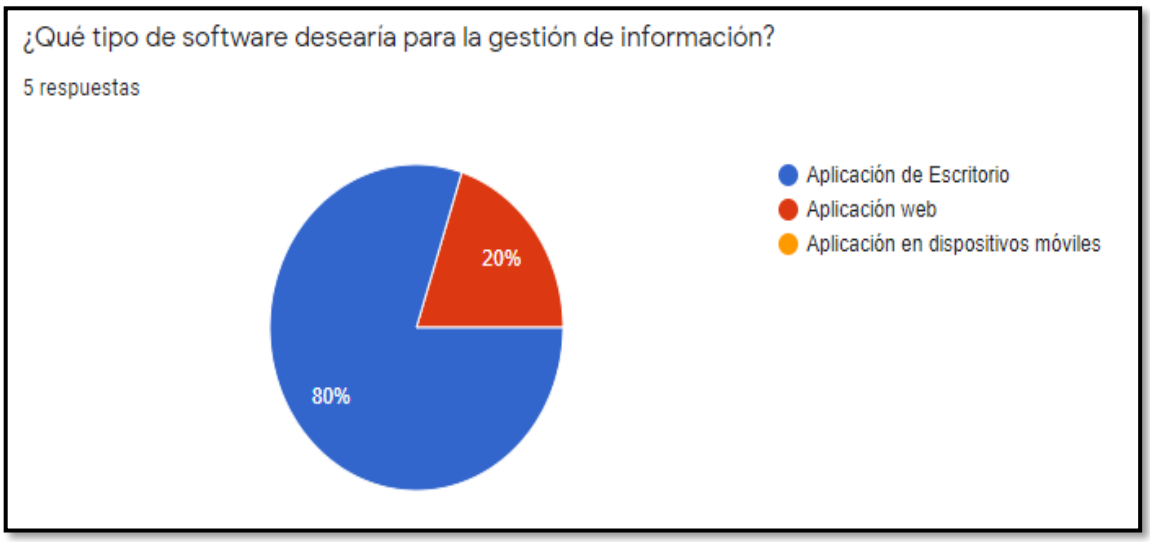

Figura 2.5 Resultados quinta pregunta

Con la pregunta número 6 se busca saber cuáles serán los criterios para el filtro de búsqueda de los compromisos de CENACE. En la Figura 2.6 se muestra que el 100% de los entrevistados desean un filtro de búsqueda basado en fechas el 80% basado en un identificador de reunión, el 60% basado en la fecha plazo del compromiso y el 20% basado en la empresa responsable.

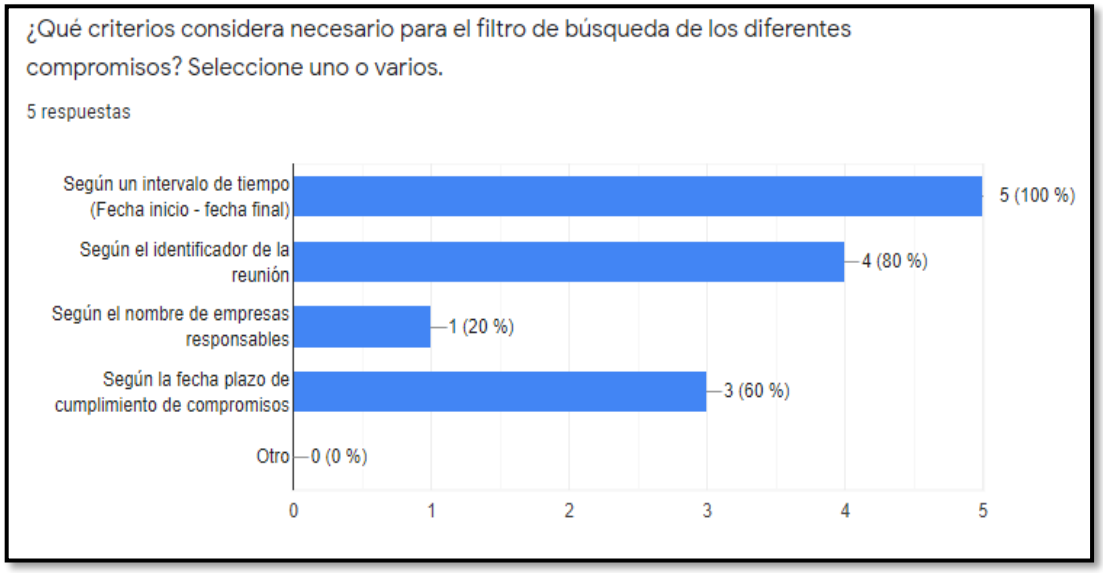

Figura 2.6 Resultados sexta pregunta

La pregunta número 7 permite percibir si se desea que se pueda modificar y eliminar los datos almacenados de eventos pasados. Los resultados de dicha cuestión se muestran en la Figura 2.7, donde el 80% indican que están de acuerdo en poder modificar o eliminar eventos pasados, mientras que el 20% opina que no es necesario.

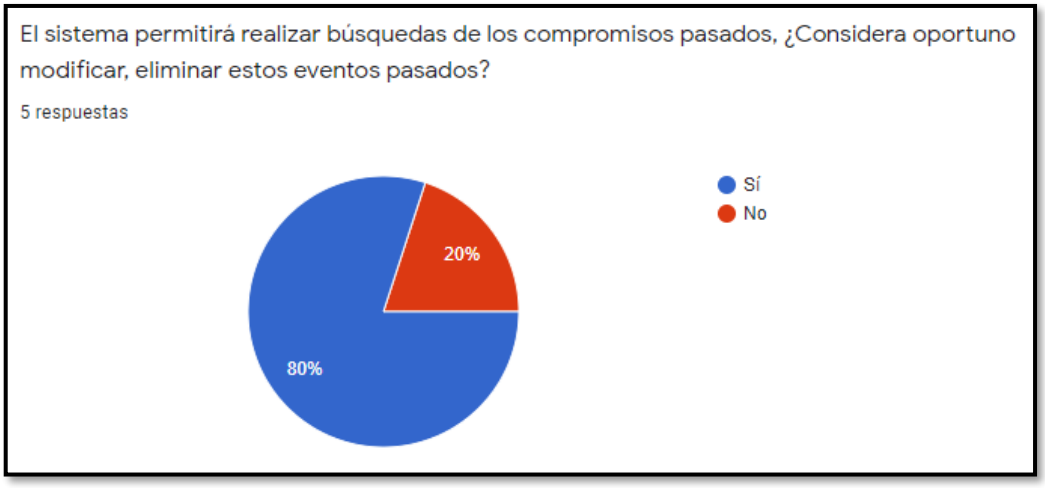

Figura 2.7 Resultados séptima pregunta

La pregunta número 8 busca establecer si se desea que el software notifique a los responsables de los compromisos próximos a cumplirse o vencidos mediante un correo electrónico, para lo cual como se muestra en la Figura 2.8 el 80% está de acuerdo.

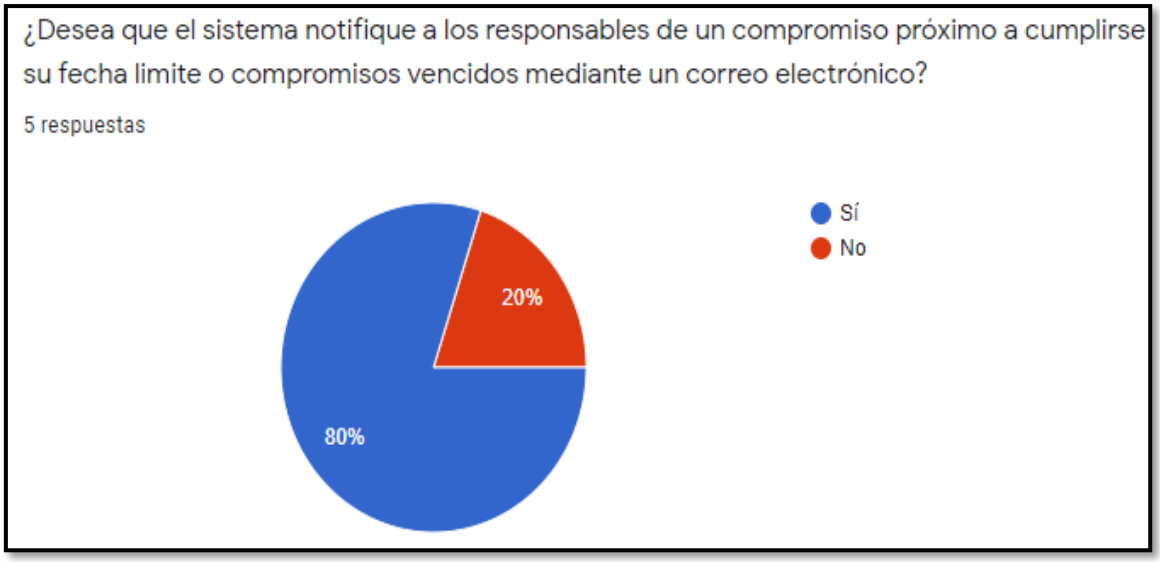

Figura 2.8 Resultados octava pregunta

La pregunta número 9 trata de resolver si se desea un módulo dentro del software que permita la configuración del envío de los correos electrónicos. Los resultados de esta pregunta se plasman en la Figura 2.9, el 80% está de acuerdo en disponer del módulo de configuración.

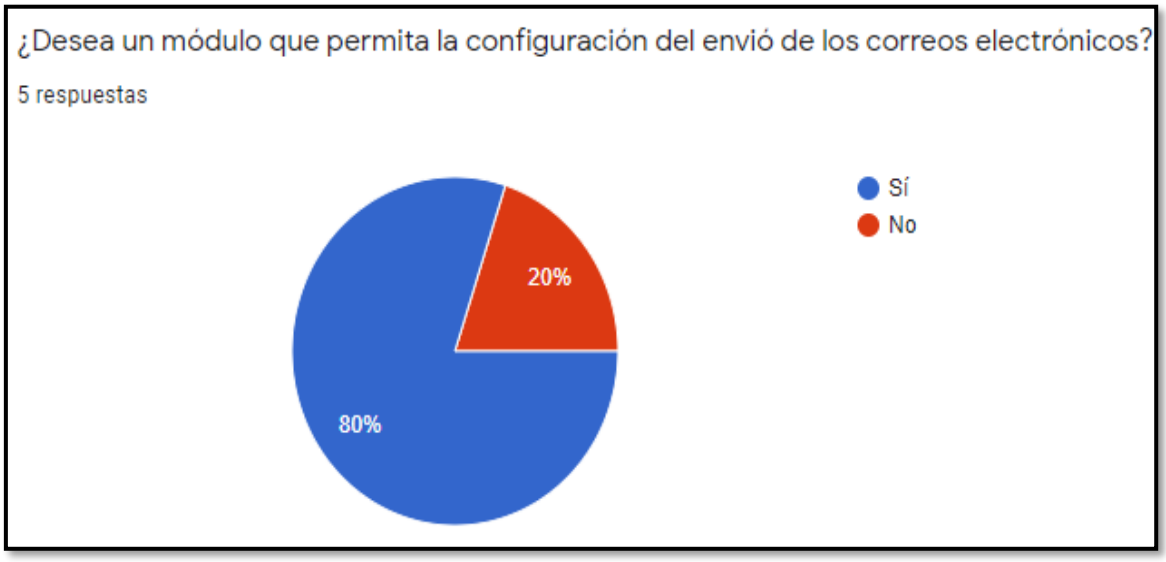

Figura 2.9 Resultados novena pregunta

La pregunta número 10 busca conocer si requieren un módulo que permita visualizar los compromisos que estén próximos a expirar y los compromisos vencidos. En la Figura 2.10 se indica que el 60% de los entrevistados están de acuerdo con este módulo.

Figura 2.10 Resultados décima pregunta

La pregunta número 11 permite conocer si se requiere iniciar sesión en la aplicación con un usuario y contraseña para poder acceder a todas las funcionalidades del aplicativo. Los resultados se muestran en la Figura 2.11, y se puede observar que el 80% de los entrevistados piensan que no es necesario.

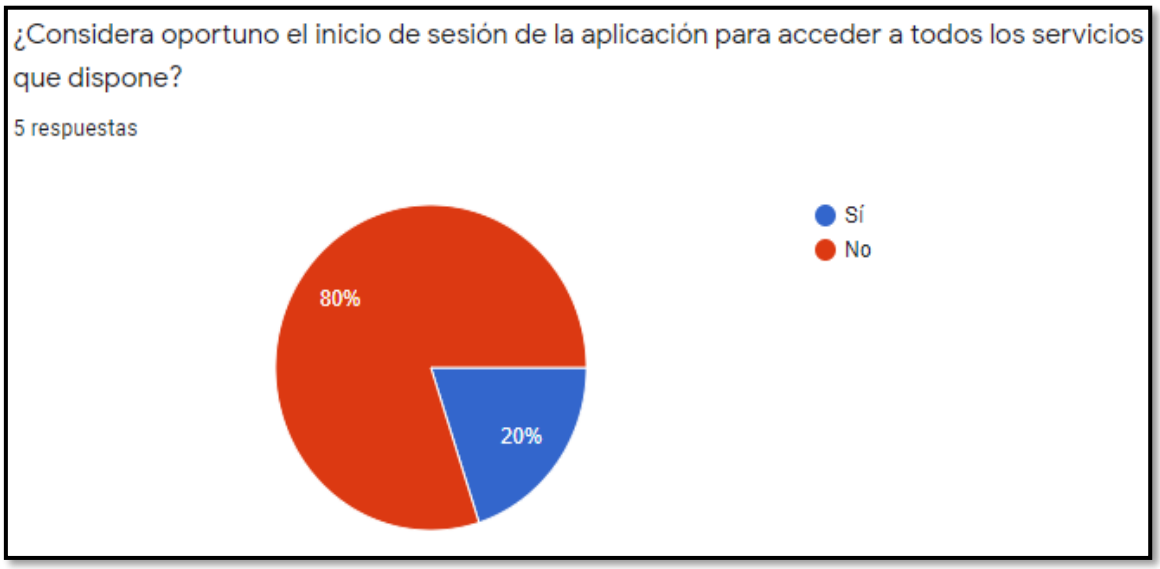

Figura 2.11 Resultados décima primer pregunta

2.1.1.3 Análisis de requerimientos

Para determinar los requerimientos del prototipo se emplearon los resultados de las encuestas, así como la información obtenida mediante reuniones con los encargados del área de operaciones nacionales de CENACE.

Requerimientos funcionales

- Se manejarán todas las funcionalidades de la aplicación por medio de un usuario master.
- El usuario master podrá realizar la creación, modificación y eliminación de las reuniones.
- El usuario master podrá insertar, eliminar o modificar los compromisos de una reunión específica y asociarlos a uno o varios responsables.
- El usuario master podrá modificar e ingresar la información correspondiente a empresas con las que tenga convenios.
- El usuario master podrá ingresar y modificar la información correspondiente a los diferentes responsables, además de asociarlos con una empresa específica.
- El usuario master podrá visualizar las reuniones ingresadas a la base de datos, además de tener un filtro de estas reuniones por código de reunión y por intervalo de tiempo.
- El usuario master podrá visualizar la información de empresas y responsables por medio de un filtro de búsqueda del nombre de la empresa y del número de cédula del responsable respectivamente.
- El usuario master podrá realizar un seguimiento de los compromisos, por medio de una búsqueda detallada a través de un periodo de tiempo o el identificador de la reunión; en la cual podrá ingresar el porcentaje de cumplimiento del compromiso.
- El usuario master podrá dar por completado un compromiso específico, además de agregar observaciones del mismo.
- El usuario master realizará la configuración del envío de notificaciones por medio de correos electrónicos a los diferentes responsables. Para ello podrá modificar el número de días antes de la caducidad de un compromiso para que se envíe el correo y el número de días después de la caducidad para que se envíe otro correo de alerta.

- El usuario master podrá modificar la información acerca del correo de salida, como por ejemplo el usuario y contraseña del mismo.
- El usuario master tendrá la posibilidad de ver una lista de compromisos próximos a cumplir su plazo límite.

Requerimientos no funcionales

- El prototipo dispondrá de una interfaz para el sistema operativo Windows.
- El prototipo empleará la base de datos de Microsoft SQL.
- El prototipo contará con un servicio web, el cual será consumido por la interfaz desarrollada para el sistema operativo Windows.

2.1.1.4 Historias de Usuario

Las historias de usuario son una herramienta utilizada para la descripción de una función del software desde el punto de vista del usuario final [21].

Formato de las historias de usuario

Para la elaboración de las historias de usuario se utilizará el formato que se muestra en la Tabla 2.1.

Tabla 2.1 Formato de historia de usuario

HISTORIA DE USUARIO					
ID:		**Rol:**		**Prioridad:**	
Nombre de Historia:					
Descripción:					

Los campos que se muestran en la Tabla 2.1 son los siguientes:

- ID: Este campo permite asociar una historia de usuario por medio de un identificador único.
- Rol: Indica el usuario al cual está dirigida la funcionalidad de la historia de usuario.
- Prioridad: Este campo denota el nivel de importancia de la historia de usuario en una escala de 1 al 5. Siendo 1, una baja prioridad y 5 una alta prioridad.

- Nombre de Historia: Es un título descriptivo que permita identificar el requerimiento del usuario.
- Descripción: En este campo se detalle el quién, qué y para qué del requerimiento del usuario.

Detalle de las historias de Usuario

Debido a que se desarrollaron varias historias de usuario, solo se presentan a manera de ejemplo dos. El resto de historias de usuario se encuentran en el ANEXO B.

En la Tabla 2.2 se describe un requerimiento de prioridad alta para que el usuario administrador pueda agregar todo tipo de reuniones al sistema y la información relacionada a este, como los compromisos y responsables.

Tabla 2.2 Historia de Usuario "Agregar reuniones al sistema"

HISTORIA DE USUARIO					
ID:	HU01	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Agregar reuniones al sistema					
Descripción: El usuario administrador desea agregar reuniones al sistema y toda la información relacionada con la reunión como lo son: tipo de reunión, fecha de reunión, tema de convocatoria y lista de compromisos.					

En la Tabla 2.3 se describe un requerimiento de prioridad medio bajo en que el usuario administrador desea que el identificador de reunión se genera de manera automática y dependiente del tipo de reunión que se seleccione.

Tabla 2.3 Historia de Usuario "Automatizar el identificador de la reunión"

HISTORIA DE USUARIO					
ID:	HU02	**Rol:**	Administrador	**Prioridad:**	2
Nombre de Historia: Automatizar el identificador de la reunión.					
Descripción: El usuario administrador desea automatizar el identificador de la reunión según el tipo de reunión (Comité de Análisis de Falla o Reunión Técnica), el número de reunión y el año de la reunión.					

2.1.2 TABLERO DE ACTIVIDADES KANBAN

Para realizar el tablero de actividades Kanban se utilizó como base los requerimientos definidos en la sección 2.1.1. La Tabla 2.4 presenta el tablero Kanban con las actividades requeridas para el prototipo.

Tabla 2.4 Actividades de Kanban

Lista de actividades
Implementar un mecanismo para agregar reuniones al sistema
Implementar un mecanismo para agregar compromisos a reuniones
Implementar una funcionalidad para agregar responsables
Implementar un mecanismo para buscar responsables
Implementar un mecanismo para agregar empresas
Implementar una funcionalidad para visualizar las diferentes reuniones almacenadas en el sistema
Implementar un método para editar una reunión y toda su información relacionada (Compromisos - Responsables)
Implementar una funcionalidad para eliminar una reunión del sistema.
Implementar un mecanismo para dar editar y dar seguimiento a un compromiso
Implementar un mecanismo para visualizar los compromisos vencidos y/o próximos a caducar
Implementar una funcionalidad para visualizar los responsables almacenados en el sistema
Implementar un método para editar un responsable
Implementar un método para visualizar las empresas almacenadas en el sistema
Implementar una funcionalidad para editar la empresa

2.1.3 ARQUITECTURA DEL PROTOTIPO

Para el prototipo propuesto para este Trabajo de Titulación se empleará la arquitectura cliente-servidor con 4 capas.

La capa de datos gestiona la información necesaria para el funcionamiento correcto del prototipo. Se accede a esta únicamente por la capa de acceso a datos.

La capa de acceso a datos permite la comunicación de la capa de datos con las capas superiores al otorgar un método de acceso a los datos, es decir permite realizar las operaciones de CRUD sobre la capa de datos.

La capa lógica de negocio, es una capa intermedia entre la capa de presentación y la de acceso a datos, se encarga de implementar la funcionalidad principal del sistema y encapsula la lógica necesaria para hacer cumplir las reglas de negocio del sistema.

Finalmente, la capa de presentación es la responsable de la interacción del usuario con sistema a través de una interfaz de usuario.

En la Figura 2.12 se presenta la arquitectura del prototipo y la interacción entre sus capas. Como se puede visualizar cada capa se comunica con sus adyacentes, es decir la capa superior consume una funcionalidad de la capa inferior.

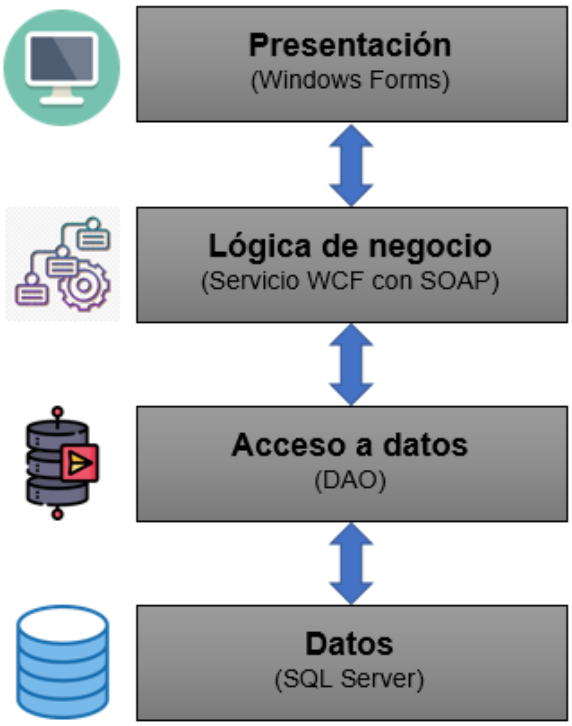

Figura 2.12 Arquitectura del prototipo

2.1.4 DIAGRAMA RELACIONAL

El prototipo propuesto trabajará con una base de datos implementada en Microsoft SQL Server, la cual permitirá almacenar toda la información. En la Figura 2.13 se presenta el diagrama relacional de la base de datos, en el cual se observan nueve tablas, cada una con un identificador único que será auto generado por el motor de la base de datos y que permitirá identificar a cada uno de los elementos de una tabla.

A continuación, se presentan a manera de ejemplo el detalle de dos tablas. El código de todas las tablas se encuentra en el ANEXO C.

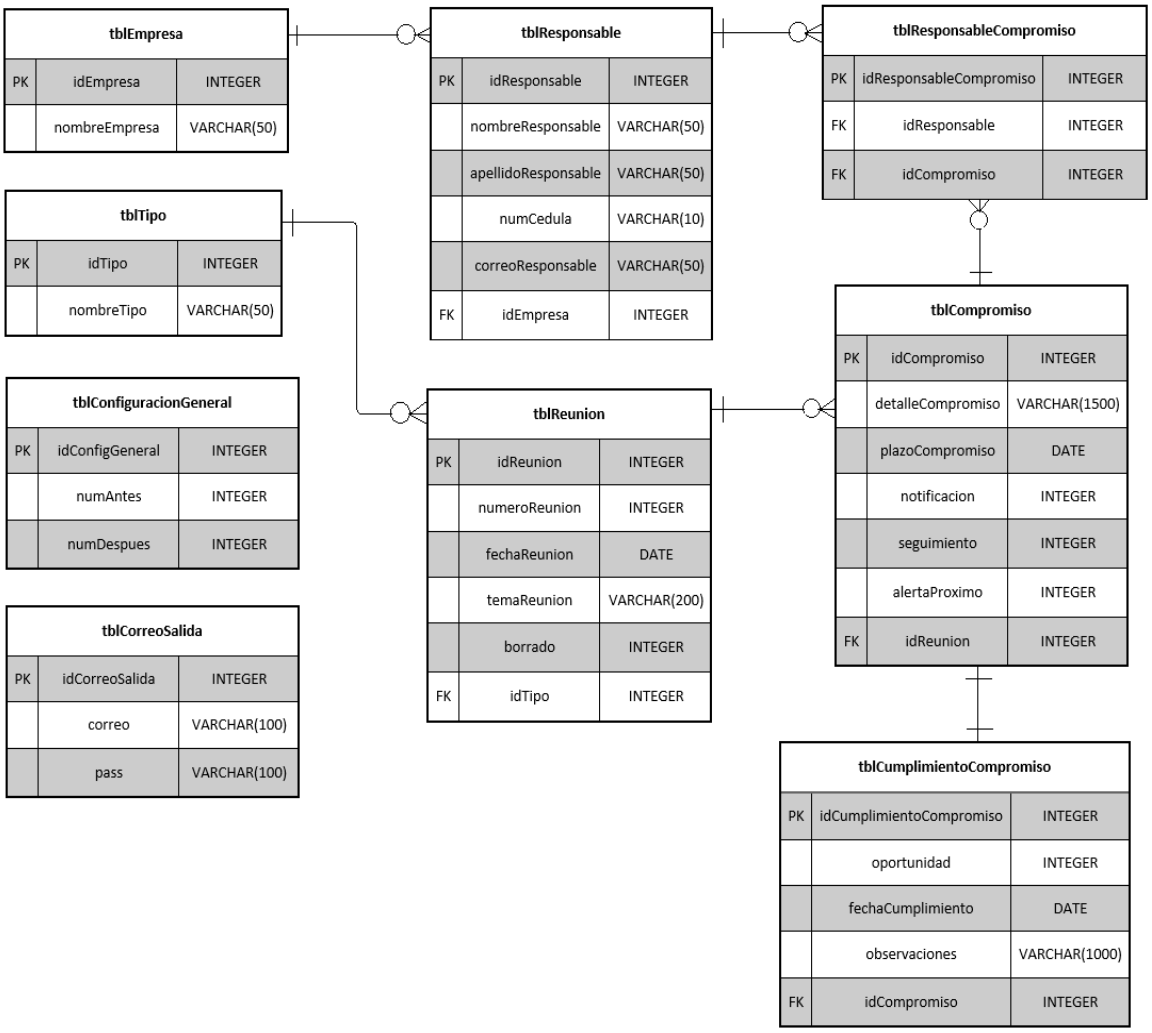

Figura 2.13 Diagrama relacional de la base de datos

La Tabla 2.5 indica el detalle de los atributos de la tabla `tblEmpresa` necesarios para identificar una empresa.

Tabla 2.5 Detalle de la tabla `tblEmpresa`

tblEmpresa		
Atributo	**Tipo de dato**	**Descripción**
`idEmpresa`	`int`	Identificador único y auto numerado que permite distinguir a cada empresa.
`nombreEmpresa`	`varchar(50)`	Nombre de la empresa.

La Tabla 2.6 indica el detalle de la tabla `tblResponsable` el cual almacenará información referente para identificar a los responsables.

Tabla 2.6 Detalle de la tabla `tblResponsable`

tblResponsable		
Atributo	**Tipo de dato**	**Descripción**
idResponsable	Int	Identificador único y auto numerado que permite distinguir a cada responsable.
nombreResponsable	varchar(50)	Nombre del responsable.
apellidoResponsable	varchar(50)	Apellido del responsable.
numCedula	varchar(10)	Número de cédula del responsable.
correoReponsable	varchar(50)	Correo electrónico del responsable.
idEmpresa	Int	Identificador único de la tabla `tblEmpresa` que sirve para relacionar un responsable con una empresa.

En el diagrama relacional de la Figura 2.13 se puede observar que una empresa puede tener uno o varios responsables, por lo cual la clave primaria (PK) de la tabla `tblEmpresa` pasa a la tabla `tblResponsable` como clave foránea (FK) y eso sirve para poder relacionar estas dos tablas.

2.1.5 DIAGRAMA DE ACTIVIDADES

En esta sección se presenta el diagrama de actividades para cada uno de los módulos del prototipo propuesto. Los diagramas de actividad son una herramienta que describen los diferentes procesos del prototipo.

Debido a la extensión de los diagramas de actividades, solo se presentan a manera de ejemplo dos. Todos los diagramas de actividades se incluyen en el ANEXO D.

2.1.5.1 Agregar Reunión

El prototipo permite al usuario ingresar la información necesaria para agregar una reunión a la base de datos. Figura 2.14 describe el proceso de agregar una reunión.

2.1.5.2 Agregar Compromisos

Como se observa en la Figura 2.14, el usuario puede agregar compromisos de una reunión, para ello la Figura 2.15 describe el proceso de agregar un compromiso.

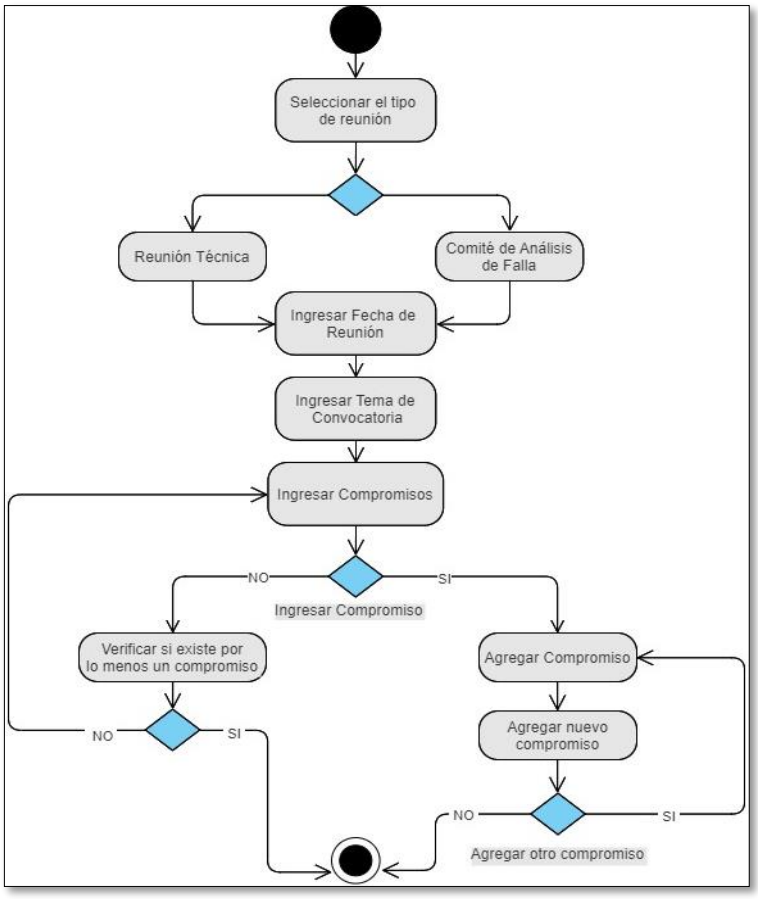

Figura 2.14 Diagrama de actividades para agregar una reunión

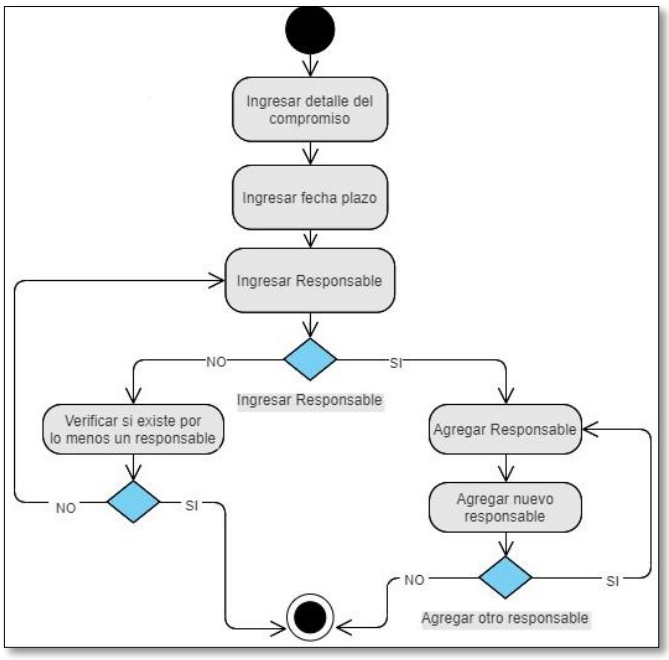

Figura 2.15 Diagrama de actividades para agregar un compromiso

2.1.6 DIAGRAMA DE CASOS DE USO

Con base en los requerimientos obtenidos en la sección 2.1.1 se generó el diagrama de caso de uso del prototipo, el cual posee un actor: el usuario.

En la Figura 2.16 se muestra el diagrama de caso de uso para todas las funcionalidades del prototipo desde el punto de vista de los diferentes actores.

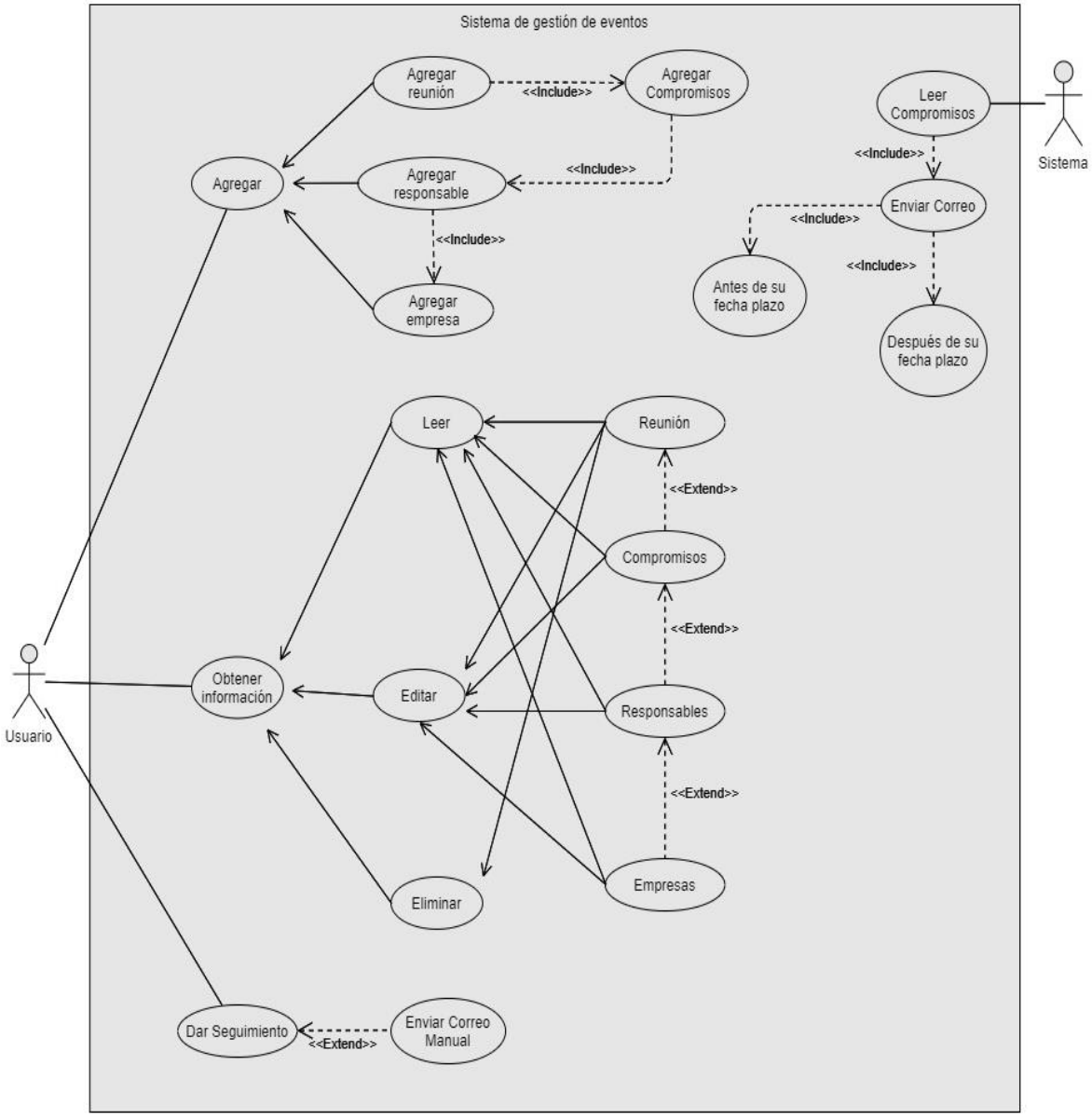

Figura 2.16 Diagrama de casos de uso del prototipo

2.1.7 DIAGRAMA DE SECUENCIA

El diagrama de secuencia describe detalladamente de manera cronológicamente cómo un grupo de objetos y componentes interactúan entre sí para completar un proceso [22].

Se presentan dos diagramas de secuencia de manera de ejemplo. Todos los diagramas de secuencia generados se incluyen en el ANEXO E

2.1.7.1 Agregar Reunión

El prototipo permite al usuario agregar una reunión y toda la información asociada a este. La Figura 2.17 describe el proceso de agregar una reunión, la secuencia inicia con el usuario seleccionando el tipo de reunión y con ello el servidor obtiene con base en la petición, el número de reunión según la selección del usuario dando como resultado el número de la reunión utilizado para generar el código de reunión de manera automática.

Una vez obtenida el código de reunión el usuario completará toda la información correspondiente de la reunión, de los compromisos y los responsables asociados al mismo, al finalizar se envía toda la información al servidor para que esta se almacene en la base de datos, y retornado como respuesta un mensaje indicando el registro exitoso o fallido.

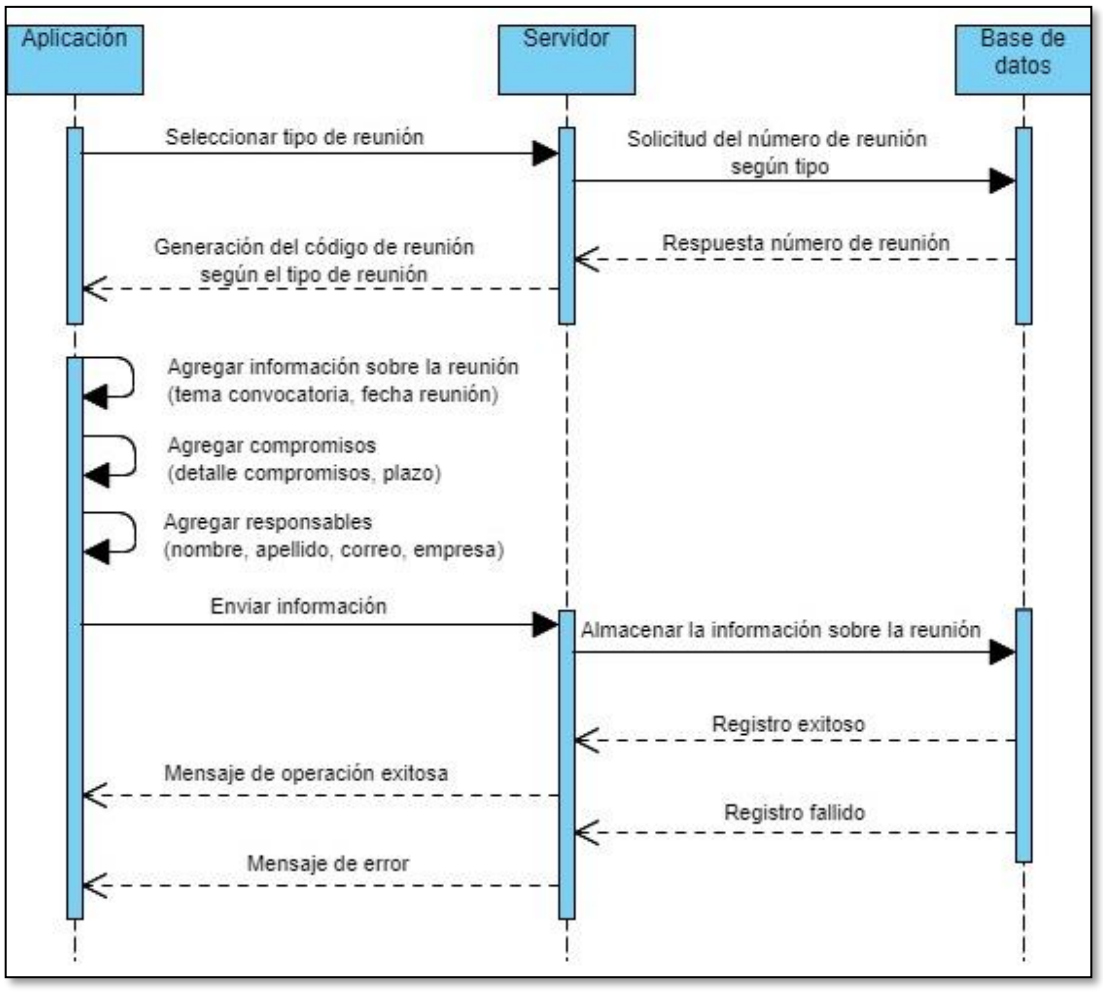

Figura 2.17 Diagrama de secuencias para agregar reunión

2.1.7.2 Agregar Responsable

La Figura 2.18 indica el proceso para agregar un responsable de un compromiso. Al inicio de la secuencia, el usuario envía toda la información perteneciente del nuevo responsable al servidor, consecuentemente el servidor realiza una petición a la base de datos para verificar si el número de cédula del nuevo responsable se encuentra en uso; si no existe otro responsable con el número de cédula indicado se da una validación correcta y se almacena en la base de datos, si el número de cédula se encuentra en uso se envía un mensaje de error al usuario indicando que el número de cédula indicado ya se encuentra registrado.

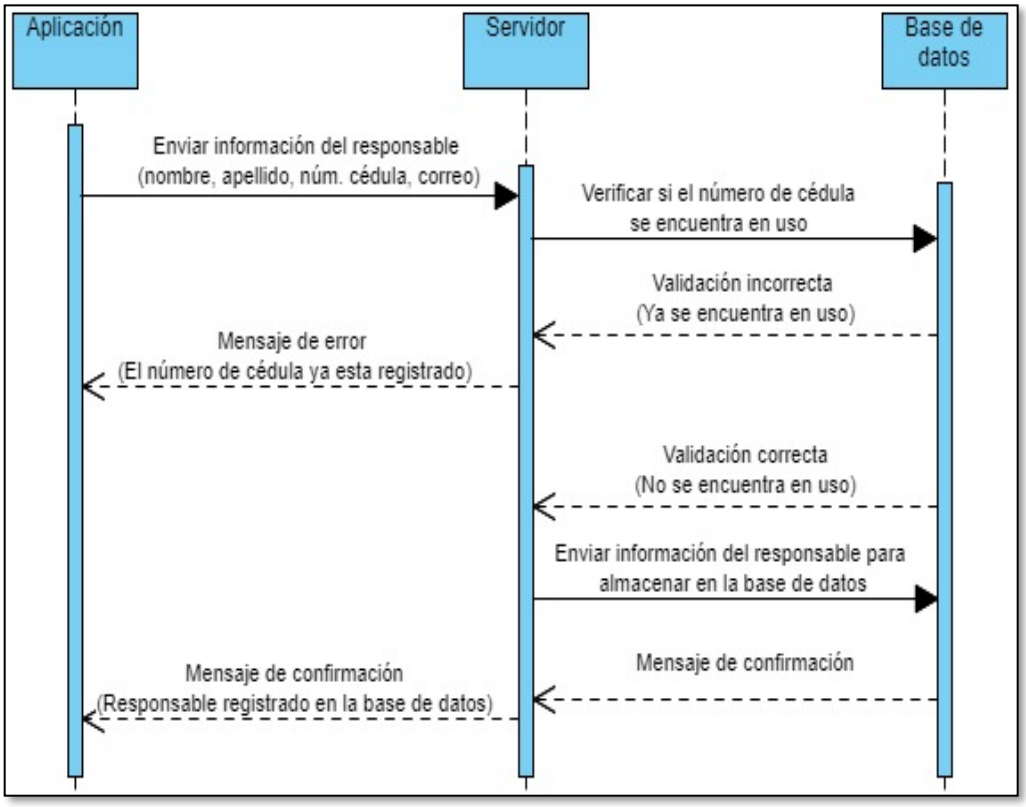

Figura 2.18 Diagrama de secuencia para agregar responsable

2.1.8 DIAGRAMA DE CLASES

El diagrama de clases permite representar de mejor manera los elementos y componentes de un sistema. En el diagrama de clases intervienen las clases de objetos y sus asociaciones, es decir la estructura y el comportamiento de cada objeto del sistema y sus relaciones con los demás objetos del sistema. Cada clase constará de un nombre de clase para identificarla, atributos propios de la clase y los métodos o funciones que la clase ofrece [23].

En la Figura 2.19 se presenta el diagrama de clases de la capa de datos, el cual asocia cada una de las tablas presentadas en el diagrama relacional de la Figura 2.13.

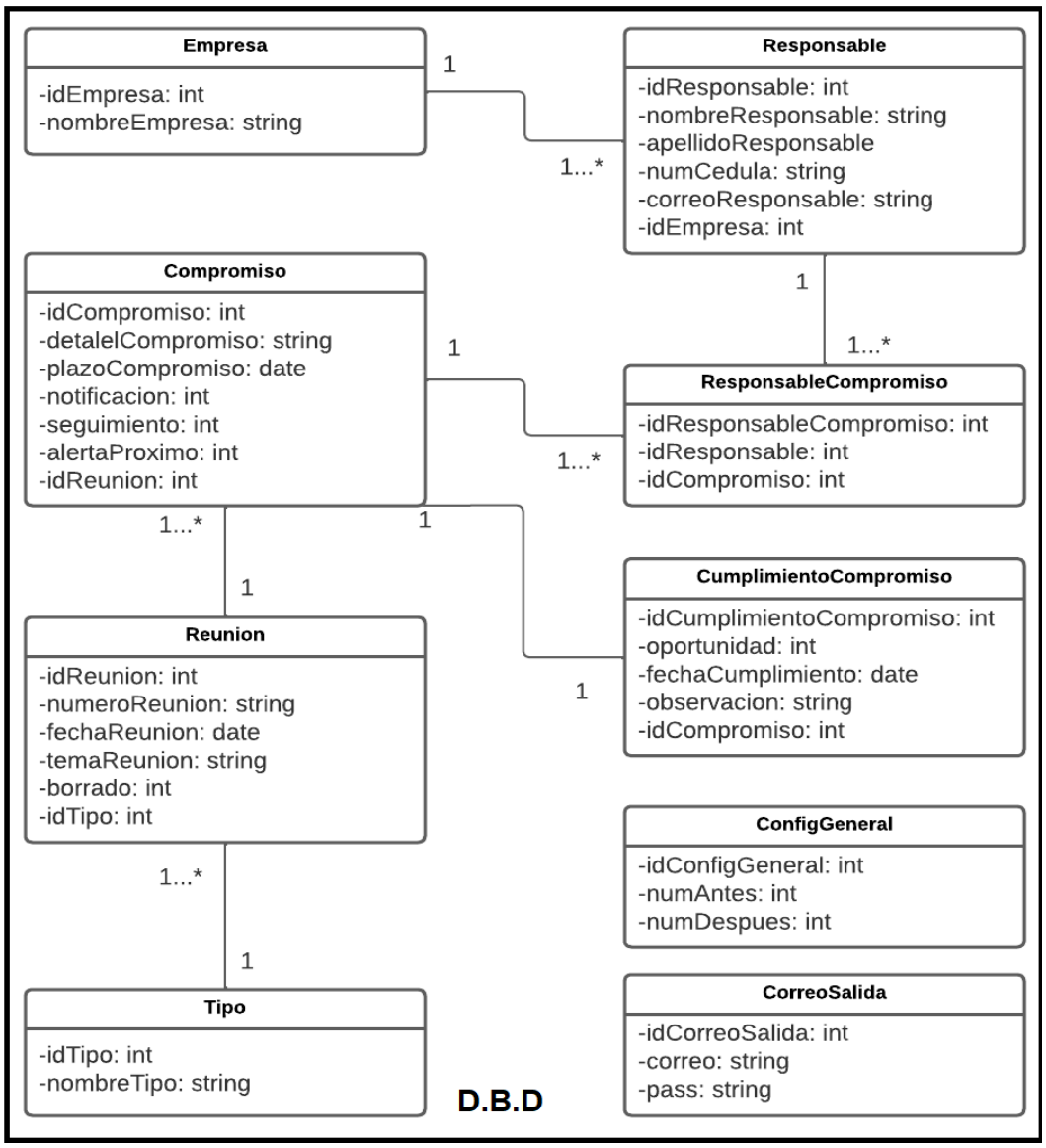

Figura 2.19 Diagrama de clases de la base de datos (D.B.D)

La Figura 2.20 muestra el diagrama de clases de la capa de acceso a datos, el cual consta de la clase `BaseDatos` que gestiona las clases de la capa de la base de datos (Figura 2.19) y utiliza el método de conexión a la base de datos de la interfaz `DataConetxt`.

La Figura 2.21 muestra el diagrama de clases de la lógica de negocio, el cual consta de la clase `Servicio` que a través de las operaciones de la interfaz `IGestionCompromisos` gestiona la información almacenada en la base de datos por medio de las clases establecidas en la capa de acceso a datos (D.A.D).

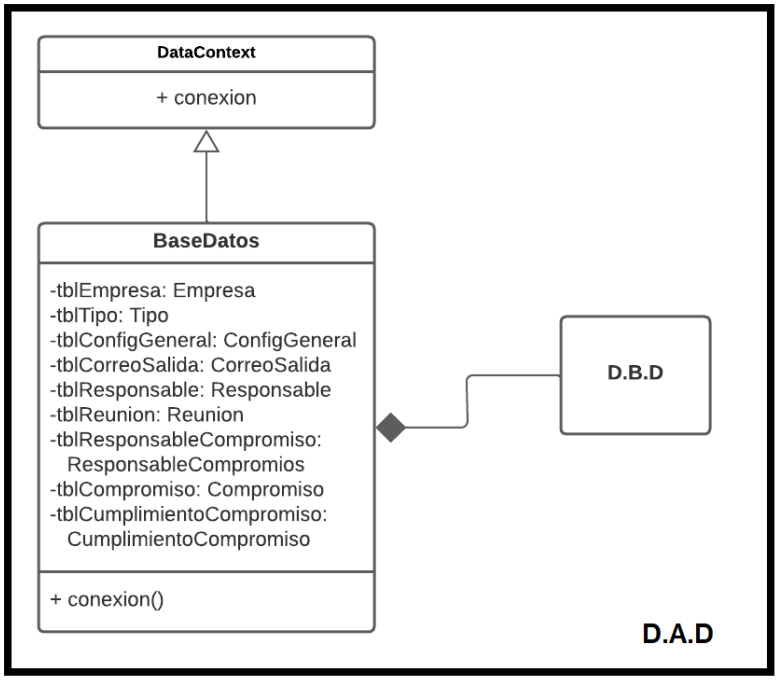

Figura 2.20 Diagrama de clases de la capa de acceso a datos (D.A.D)

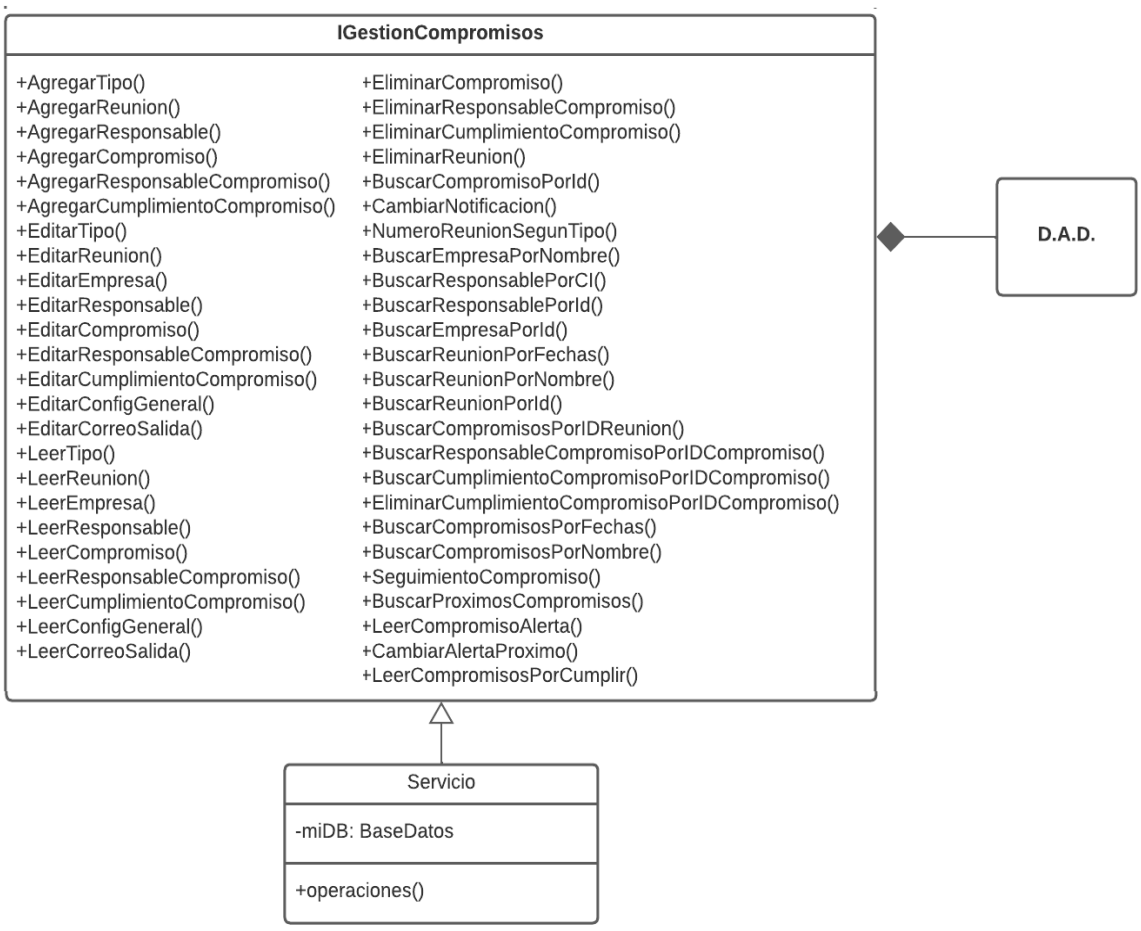

Figura 2.21 Diagrama de clases de la lógica de negocio

2.1.9 SKETCHES

Los bocetos o *sketches* son una herramienta muy importante para el diseño de la capa de presentación, que es conocida como la interfaz gráfica encargada de la interacción con el usuario final. El principal objetivo de los *sketches* es detallar la ubicación y la intervención de los componentes de cada vista de la interfaz gráfica.

En esta sección se presentan los *sketches* de los principales módulos del prototipo.

La Figura 2.22 muestra el *sketch* de la página principal de la aplicación, en donde se tiene una barra de título en la parte superior en la que se encuentra el título de la actividad, los botones de minimizar, maximizar y cerrar, respectivamente. En la derecha cuenta con un menú de opciones, mediante el cual el usuario podrá seleccionar la tarea que desea realizar como Agregar, Buscar, Realizar el seguimiento de Compromisos y Configurar el prototipo. Finalmente se posee un panel general en el cual se encuentra situado el logo de la CENACE; por otro lado, en este panel se presentarán las interfaces de cada tarea del prototipo.

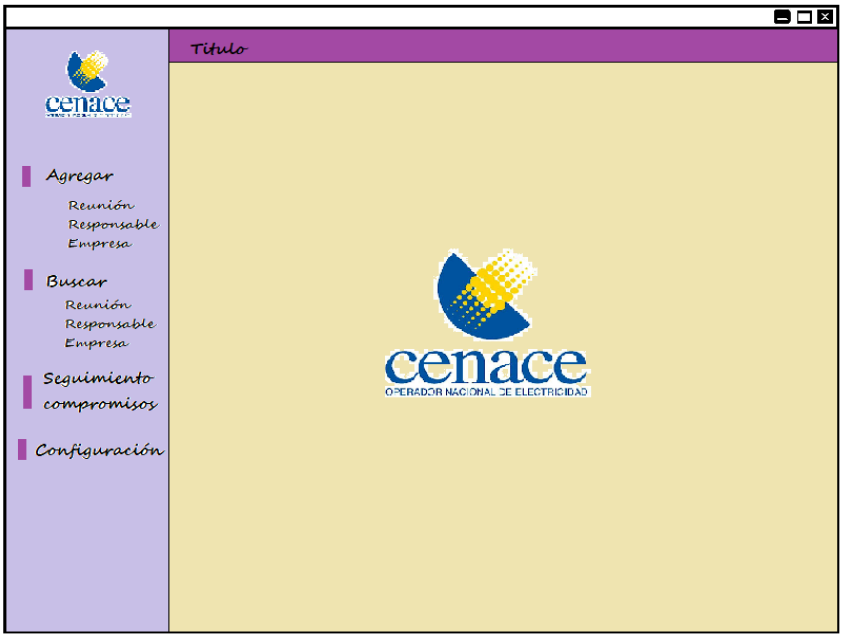

Figura 2.22 *Sketch* de la pantalla principal

La Figura 2.23 muestra el *sketch* de la interfaz que permitirá agregar o editar una reunión. En esta vista se presentarán los componentes o se cargará la información de una reunión respectivamente. El *sketch* cuenta con componentes que permiten seleccionar el tipo de reunión, la fecha de reunión, el tema de convocatoria, además de botones que permiten agregar, editar o eliminar compromisos de la reunión y dos listas que presentarán los

compromisos y los responsables de los mismos. Finalmente cuenta con un botón para Agregar la reunión o para guardar los cambios realizados en la reunión.

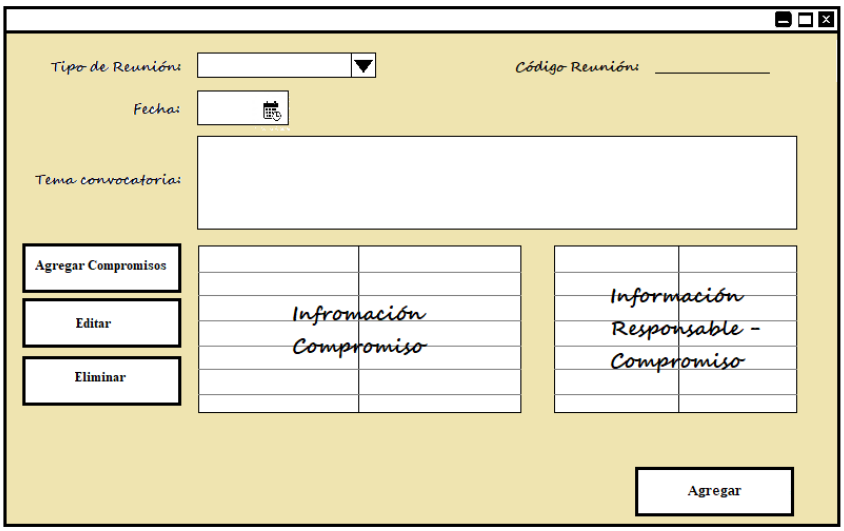

Figura 2.23 *Sketch* para agregar o editar una reunión

La Figura 2.24 muestra el *sketch* de la interfaz para agregar o editar un compromiso de una reunión, en esta vista se presentarán los componentes o se cargará la información del compromiso seleccionado previamente. Esta vista cuenta con componentes que permiten ingresar el detalle del compromiso, plazo, botones que permiten agregar, editar o eliminar responsables del compromiso y una lista en donde se puede visualizar a los responsables del compromiso. Finalmente, cuenta con un botón para finalizar la agregar o guardar la información del compromiso.

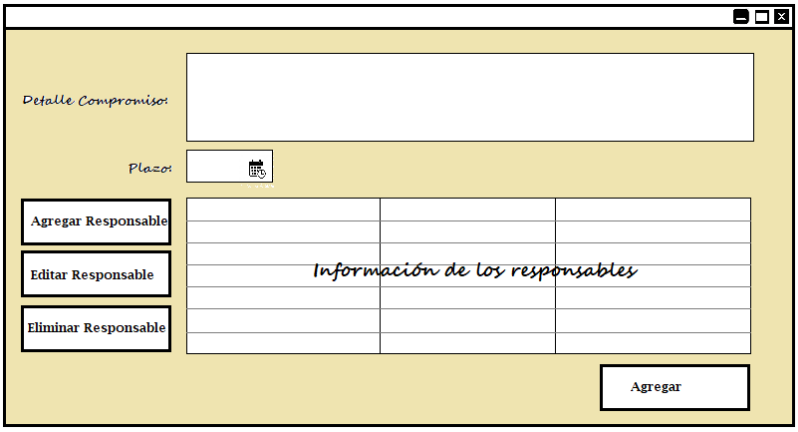

Figura 2.24 *Sketch* para agregar o editar un compromiso

Todos los *sketches* se presentan en el ANEXO F

2.2 IMPLEMENTACIÓN

2.2.1 INSTALACIÓN DE LAS HERRAMIENTAS NECESARIAS

En esta sección se presenta un resumen del proceso de la instalación de las herramientas necesarias para la implementación del prototipo.

2.2.1.1 Microsoft Visual Studio

Para instalar Microsoft Visual Studio, hay que ir a su página oficial [24] y descargar el instalador según el sistema operativo instalado. Una vez descargado, se debe ejecutar el archivo de instalación y seguir los pasos del mismo para completarlo. Una vez finalizado, se mostrará una ventana de bienvenida en la cual se puede iniciar sesión y activar la licencia. Una vez activada la licencia, el software está listo para ser utilizado, como se observa en la Figura 2.25, A través de Visual Studio se puede clonar un repositorio, abrir un proyecto o solución y crear un proyecto o solución.

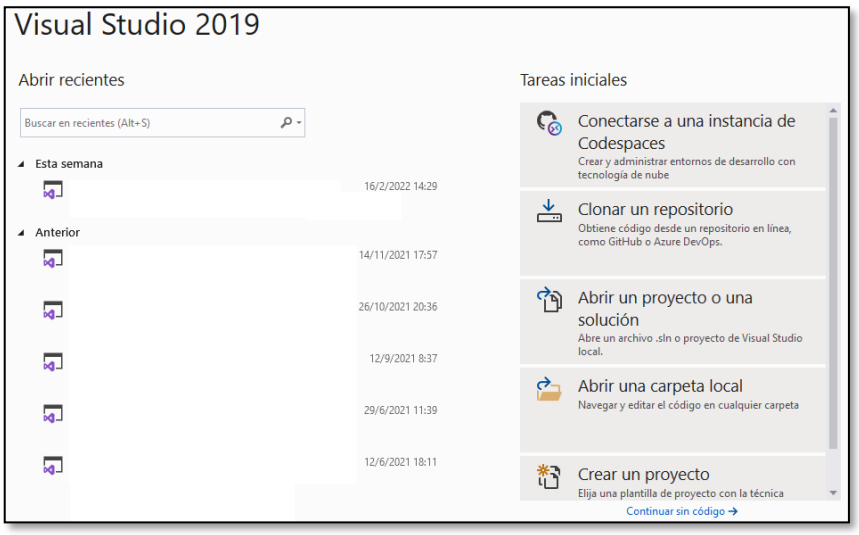

Figura 2.25 Ventana de inicio de Microsoft Visual Studio 2019

Para el funcionamiento del prototipo es necesario instalar algunos paquetes NuGet[25], estos paquetes son herramientas que facilitan tareas al momento de la implementación de una aplicación como la presentación de íconos y diseños, envíos de correos electrónicos, etc. Para instalar los paquetes NuGet, se necesita crear una solución y una vez creada, en el explorador de soluciones se presiona clic derecho para desplegará una lista de opciones

[25] NuGet es una extensión de Visual Studio que permite agregar, eliminar o modificar referencias a librerías o proyectos de Visual Studio.

en donde se encuentra “Administrar paquetes NuGet para la solución”, como se indica en la Figura 2.26.

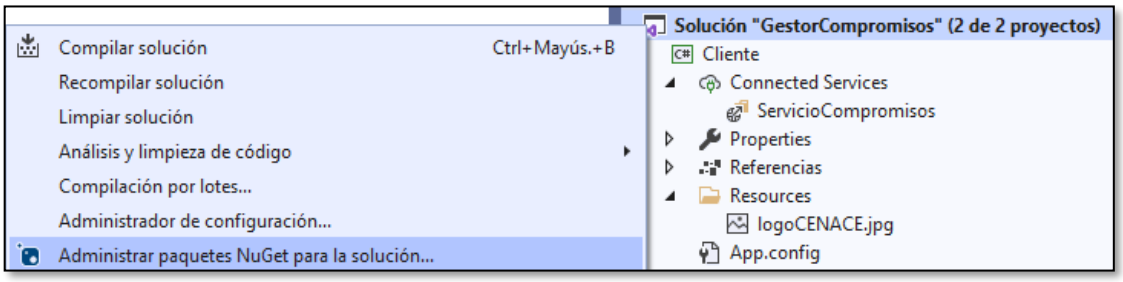

Figura 2.26 Administrar paquetes NuGet

En la Tabla 2.7 se muestran los paquetes necesarios para el funcionamiento del prototipo.

Tabla 2.7 Paquetes NuGet instalados

Nombre	Funcionalidad	Versión
`FontAwesome.Sharp`	Sustituye imágenes de íconos comunes por gráficos vectoriales convertidos en fuentes	v5.15.3
`System.Net.Mail`	Envía correo electrónicos	-

2.2.1.2 Microsoft SQL Server Management

Para instalar Microsoft SQL Server Management se debe descargar el instalador para el sistema operativo correspondiente, desde su página oficial [25]. Una vez descargado, se debe ejecutar el archivo de instalación y seguir los pasos del mismo para completarlo. Al finalizar, se mostrará una ventana de bienvenida, en la cual se puede iniciar sesión y realizar consultas a las diferentes bases de datos, como se muestra en la Figura 2.27.

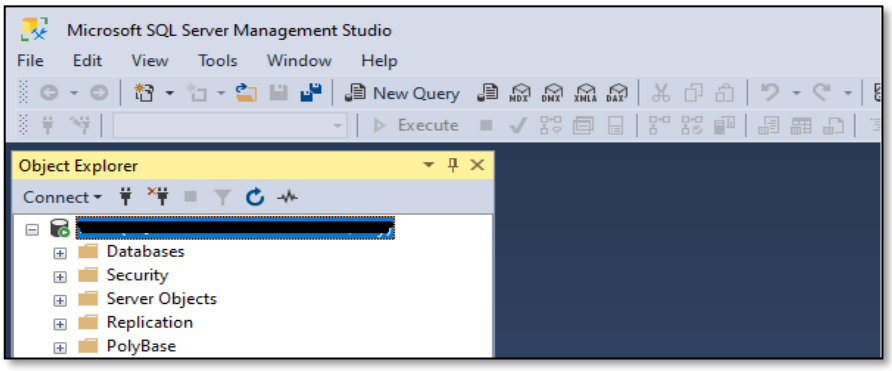

Figura 2.27 Página de inicio de Microsoft SQL Server Management Studio

2.2.1.3 INTERNET INFORMATION SERVICES (IIS)

IIS es un complemento de Windows, por lo cual no se requiere o instalarlo, solo es necesario habilitarlo. Para habilitar IIS en Windows mediante la opción de “Programas” del “Panel de control”, se accede a la opción “Activar o desactivar Programas y características” [26]. En la pantalla que se presente hay que seleccionar la característica IIS y aceptar la habilitación del mismo, como se observa en la Figura 2.28.

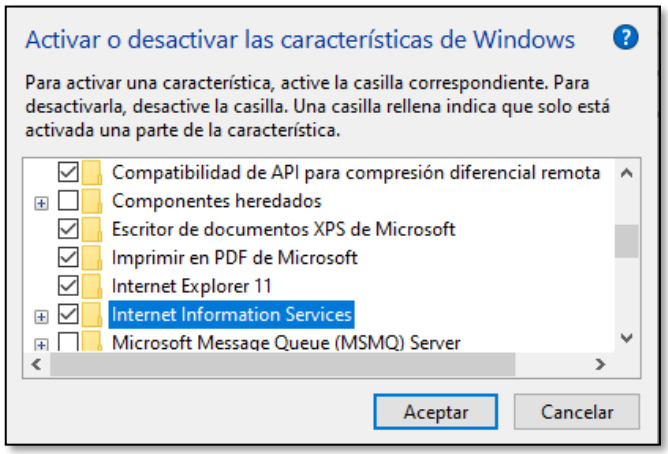

Figura 2.28 Habilitar IIS en Windows

2.2.2 CONTROL DE VERSIONES

Para la realización de un control de versiones por medio de un repositorio de GitHub[26], en el proyecto de Visual Studio, en el explorador de soluciones se presiona clic derecho sobre el nombre de la solución para desplegar una lista de opciones, en la cual se encuentra “Crear repositorio GIT”, como se indica en la Figura 2.29.

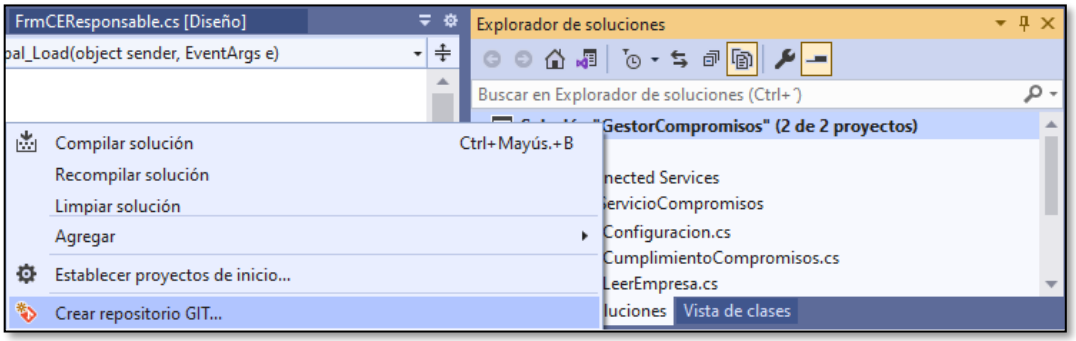

Figura 2.29 Crear repositorio de GitHub (Parte 1)

[26] GitHub es un servicio basado en la nube en el cual desarrolladores alojan sus proyectos utilizando un control de versiones del sistema.

Una vez que se accede a esta opción se despliega una ventana en la que se ingresan las credenciales de GitHub, el nombre del repositorio y el nivel de privacidad del mismo, como se observa en la Figura 2.30.

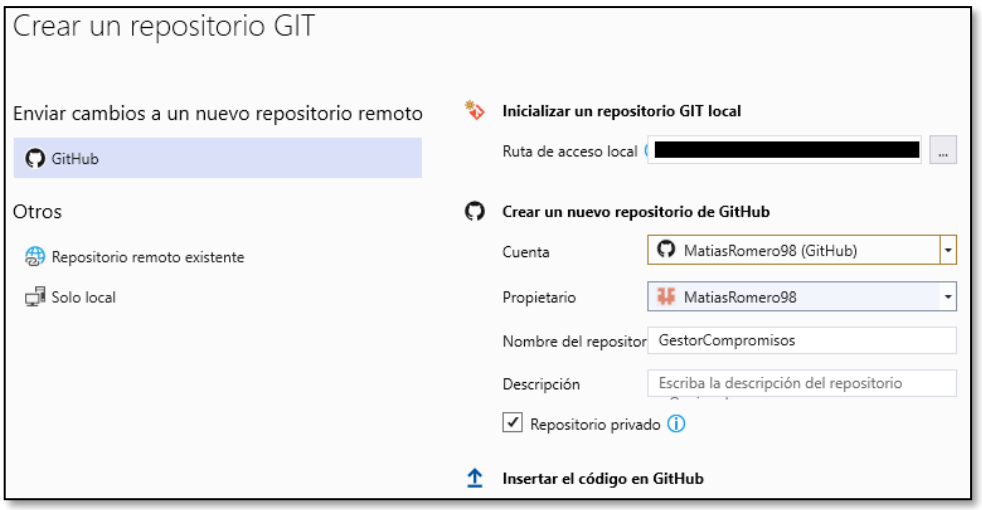

Figura 2.30 Crear repositorio de GitHub (Parte 2)

2.2.3 IMPLEMENTACIÓN DE LA BASE DE DATOS

En esta sección se presenta la codificación de la base de datos en SQL Server, la misma que permitirá el almacenamiento de la información del prototipo de gestión y seguimiento de compromisos de CENACE.

La creación de la base de datos y de sus tablas se hará por medio de las sentencias de SQL, y con base en el diagrama relacional de la Figura 2.13 (sección 2.1.4). En el Código 2.1, se puede observar el código para crear la base de datos, en la línea 1 se muestra la sentencia para crear la base de datos llamada `dBCenaceAO`, y en la línea 3 se muestra la sentencia para usar dicha base de datos.

```
1  create database dBCenaceAO
2  go
3  use dBCenaceAO
4  go
```

Código 2.1 Creación y uso de la base de datos

En el Código 2.2 se muestra la creación de la tabla `tblCompromiso`. En la línea 37 se observa la sentencia para crear la tabla `tblCompromiso` y en las líneas de la 38 a la 45

se definen sus atributos. La línea 38 define la clave primaria y auto numerada de la tabla, mientras que las líneas 39 a 40 hacen referencia a la clave foránea de la tabla `tblReunion`.

```
37  create table tblCompromiso(
38  idCompromiso int primary key identity (1,1),
39  idReunion int foreign key references tblReunion (idReunion)
40  on update no action on delete no action,
41  detalleCompromiso varchar (1500),
42  plazoCompromiso DateTime,
43  notificacion int,
44  seguimiento int,
45  alertaProximo int
46  )
47  go
```

Código 2.2 Creación de la tabla `tblCompromiso`

El código completo del prototipo incluido el código de la base de datos `dBCenaceAO` se encuentra en el ANEXO G

2.2.4 IMPLEMENTACIÓN DE LA CAPA DE ACCESO A DATOS

Dentro de la solución de Visual Studio se crea un proyecto del tipo “Aplicación de servicios WCF” que servirá para la codificación de la capa de acceso a datos y la capa lógica de negocio. Para la capa de acceso a datos se utilizará el método DAO, para lo cual dentro del proyecto se crea la clase llamada `BaseDatos`.

Además, en el proyecto se añade una referencia a la librería `system.data.linq` que ayudará a crear la conexión de con Server y referenciar cada tabla de la base de datos para poder trabajar con ella (ver Figura 2.31).

Figura 2.31 Agregar referencia

Una vez seleccionada esta opción se desplegará la ventana del Administrador de Referencias, mediante la cual se puede realizas búsqueda de las distintas librerías que se pueden agregar al proyecto, en particular si se ingresa la palabra clave "`linq`" se presentará un listado de librerías relacionadas, y del listado se deberá seleccionar las dos referencias del servicio de `linq` y se presiona en el botón aceptar, como se muestra en la Figura 2.32.

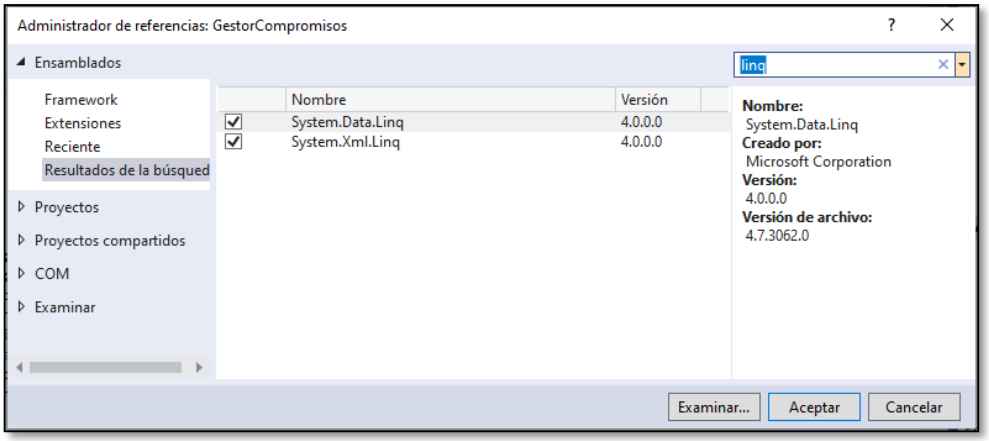

Figura 2.32 Administrador de referencias

Finalmente, para evitar emplear el nombre completamente calificado de las clases que tiene la librería, se agrega el código de las líneas 5 y 6 del Código 2.3, en la sección de espacios de nombres.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Data.Linq;
6  using System.Data.Linq.Mapping;
```

Código 2.3 Espacio de nombres de la clase `BaseDatos`

Dentro de la clase `BaseDatos`, se requiere un objeto para realizar la conexión con la base de datos, el mismo que requiere de un `string` que contenga los detalles de la conexión.

Visual Studio dispone de una herramienta para ayudar en el proceso de conectarse con la base de datos, para emplear la herramienta se usa la opción del menú "Herramientas" y se escoge "Conectar con la base de datos", como se muestra en la Figura 2.33.

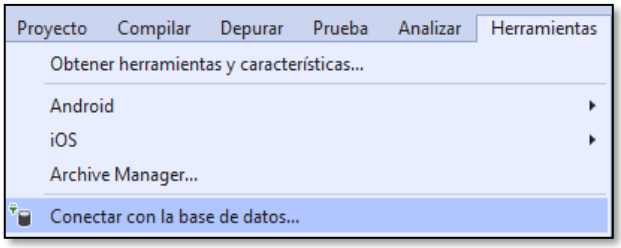

Figura 2.33 Herramienta de Visual Studio para conectar la solución con la base de datos

Una vez seleccionada la opción “Conectar con la base de datos” se desplegará una ventana que permite indicar el origen de datos, el nombre del servidor, el tipo de autenticación con el servidor y la base de datos del servidor a la que se desea realizar la conexión, como se muestra en la Figura 2.34.

Finalmente, para obtener el `string` de la conexión, mediante el explorador de servidores, se escoge la conexión realizada, y en la propiedad “Cadena de conexión” se pude observar la información requerida (ver Figura 2.35).

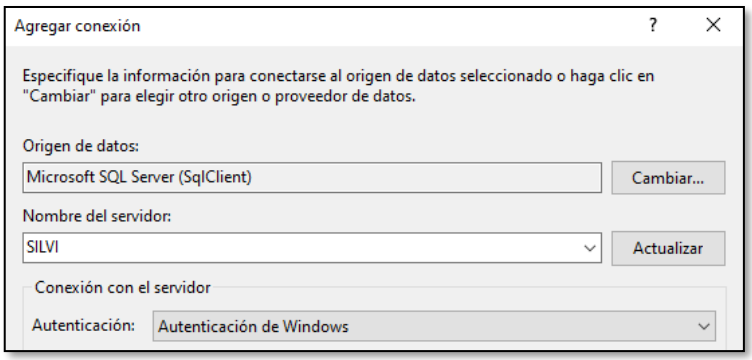

Figura 2.34 Agregar conexión con el servidor de base de datos

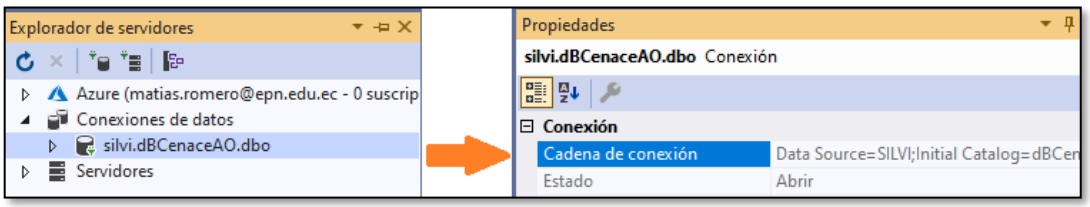

Figura 2.35 Cadena de conexión de la base de datos

La cadena de conexión será empleada en la clase `BaseDatos`. El código de la clase `BaseDatos` se visualiza en el Código 2.4; la línea 12 muestra la cadena de conexión

empleada en el constructor de la clase y en las líneas 13 al 21 se especifican los atributos de la clase que servirán para asociar cada una de las tablas de la base de datos con las clases generadas en la capa de acceso a datos.

```
10      public class BaseDatos : DataContext
11      {
            1 referencia | 0 cambios | 0 autores, 0 cambios
12          public BaseDatos() : base(@"Data Source=SILVI;Initial Catalog=dBCenaceA
13          public Table<Tipo> tblTipo;
14          public Table<Reunion> tblReunion;
15          public Table<Empresa> tblEmpresa;
16          public Table<Responsable> tblResponsable;
17          public Table<Compromiso> tblCompromiso;
18          public Table<CumplimientoCompromiso> tblCumplimientoCompromiso;
19          public Table<ResponsableCompromiso> tblResponsableCompromiso;
20          public Table<ConfigGeneral> tblConfigGeneral;
21          public Table<CorreoSalida> tblCorreoSalida;
22      }
```

Código 2.4 Clase `BaseDatos`

Para realizar la asociación de las tablas de la base de datos con las clases de la capa de acceso a datos, se debe considerar que cada clase debe tener los mismos atributos que las tablas. En el Código 2.5 se muestra la clase `Compromiso` que servirá para realizar la asociación con la tabla `tblCompromiso`.

```
96      [Table(Name = "tblCompromiso")]
        -referencias | 0 cambios | 0 autores, 0 cambios
97      public class Compromiso
98      {
99          [Column(IsPrimaryKey = true, IsDbGenerated = true, AutoSync = AutoSync.OnInsert)]
100         public int idCompromiso;
101         [Column]
102         public int idReunion;
103         [Column]
104         public string detalleCompromiso;
105         [Column]
106         public DateTime plazoCompromiso;
107         [Column]
108         public int notificacion;
109         [Column]
110         public int seguimiento;
111         [Column]
112         public int alertaProximo;
            -referencias | 0 cambios | 0 autores, 0 cambios
113         public override string ToString()
114         {
115             return idCompromiso + idReunion + detalleCompromiso + plazoCompromiso.ToString();
116         }
117     }
```

Código 2.5 Asociación de la clase `Compromiso` con la tabla `tblCompromiso`

En el Código 2.5, la línea 96 indica la tabla que está referenciada con la clase `Compromiso`, es decir la tabla `tblCompromiso`. Las líneas 99 a 112 indican los atributos que posee la clase y que están asociados con columnas de la tabla. Finalmente, entre las líneas 113 a 116 se define un método para presentar mediante un `string` ciertos atributos de la clase.

El código completo del prototipo incluido el código de la capa de acceso a datos se encuentra en el ANEXO G.

2.2.5 IMPLEMENTACIÓN DE LA CAPA LÓGICA DE NEGOCIO

Para la codificación de la capa lógica de negocio, en la solución de Visual Studio se agrega un nuevo proyecto del tipo “Aplicación del servicios WCF”, como se observa en la Figura 2.36.

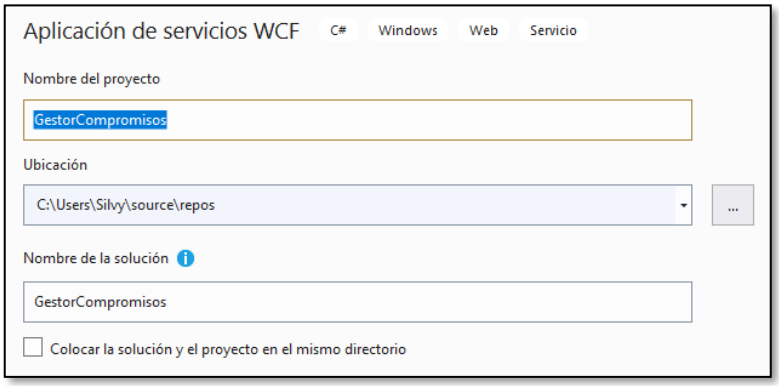

Figura 2.36 Creación del proyecto del tipo Aplicación de servicios WCF

En los archivos generados por Visual Studio, se requiere modificar dos de ellos. El archivo `IGestionCompromiso.cs` que define el contrato de servicio y el archivo `Service1.svc` que contiene la implementación del contrato en el archivo `GestionCompromiso.cs`, como se observa en la Figura 2.37.

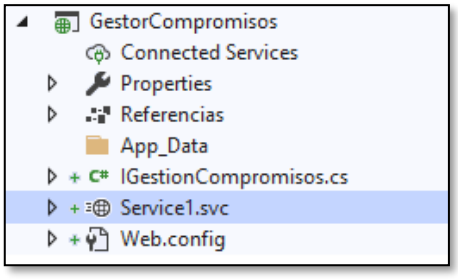

Figura 2.37 Archivos del proyecto Aplicaciones de servicio WCF

2.2.5.1 Implementación del contrato de servicio

El contrato de servicio (`IGestionCompromiso.cs`) es una interfaz que debe ser implementada por un servicio (`Service1.svc`). En la interfaz se deben definir los contratos de operación, lo cual incluye la definición de los parámetros que requiere cada operación así como el tipo de valor que cada operación retornará.

En el Código 2.6 se describe un par de operaciones para agregar datos a la base de datos. La línea 12 indica que la interfaz `IGestionCompromiso` es un contrato de servicio, la línea 17 y 20 indican que el método definido en la siguiente línea (línea 18 y 21 respectivamente) es un contrato de operación.

En la línea 18 se define el método denominado `AgregarTipo` que no devuelve un valor (`void`) y que acepta como parámetros de entrada un `string`. Este método permitirá agregar los diferentes tipos de reunión que CENACE necesite.

```
12      [ServiceContract]
        1 referencia | 0 cambios | 0 autores, 0 cambios
13      public interface IGestionCompromisos
14      {
15
16          //Agregar
17          [OperationContract]
            1 referencia | 0 cambios | 0 autores, 0 cambios
18          void AgregarTipo(string nombreTipo);
19
20          [OperationContract]
            1 referencia | 0 cambios | 0 autores, 0 cambios
21          void AgregarReunion(int idTipoReunion, int numeroReunion,
22              DateTime fechaReunion, string temaReunion);
23
```

Código 2.6 Operaciones de contrato de la interfaz `IGestionCompromiso`

En el Código 2.7 se aprecia un método denominado `EditarEmpresa` que no devuelve un valor (`void`) y que acepta como parámetros de entrada el identificador de la empresa a editar y el nombre que se modificará, este método permite modificar el nombre de una empresa.

```
47          [OperationContract]
            1 referencia | 0 cambios | 0 autores, 0 cambios
48          void EditarEmpresa(int idEmpresa, string nombreEmpresa);
```

Código 2.7 Operaciones de contrato de la interfaz `IGestionCompromiso` para edición

En la línea 83 del Código 2.8 se define el método `LeerCompromiso` que devuelve una lista de objetos de tipo `Compromisos`. Este método permite recuperar todos los compromisos almacenados en la base de datos.

```
82      [OperationContract]
        1 referencia | 0 cambios | 0 autores, 0 cambios
83      List<Compromiso> LeerCompromiso();
```

Código 2.8 Operaciones de contrato de la interfaz `IGestionCompromiso` para recuperar información

En el Código 2.9 se presenta una operación para eliminar elementos de la base de datos. La línea 100 indica dicha operación denominada `EliminarCompromiso` que no devuelve un valor (*void*) y como parámetros de entrada es un dato del tipo *int* del id del compromiso a eliminar, esta operación permite eliminar un compromiso específico por medio de su identificador.

```
 99     [OperationContract]
        1 referencia | 0 cambios | 0 autores, 0 cambios
100     void EliminarCompromiso(int idCompromiso);
```

Código 2.9 Operaciones de contrato de eliminar

La línea 134 del Código 2.10 indica una operación denominada `BuscarReunionPorFechas` que devuelve una lista de la clase `Reunion` y como parámetros de entrada son un dato del tipo *dateTime* con la fecha de inicio y otro dato del tipo *dateTime* con la fecha de fin, esta operación permite obtener todas las reuniones dadas en un intervalo de tiempo.

```
133     [OperationContract]
        1 referencia | 0 cambios | 0 autores, 0 cambios
134     List<Reunion> BuscarReunionPorFechas(DateTime fechaInicio, DateTime fechaFin);
```

Código 2.10 Operaciones de contrato extras

2.2.5.2 Implementación del servicio

Dentro de la clase `Service1.svc` se va a implementar la interfaz del contrato de servicio. En el Código 2.11 se presentan los métodos definidos en el Código 2.6.

La línea 13 indica que la clase `Service1` implementa la interfaz `IGestionCompromisos`, en línea 15 se crea un objeto de la clase `BaseDatos` para hacer uso de la capa de acceso de datos. Entre las líneas 18 a 25 se agregan los diferentes tipos de reunión de CENACE en la base de datos, en particular en la línea 20 se crea una instancia de la clase `Tipo`, en la línea 21 se inicializa el objeto de la clase `Tipo` con los parámetros de entrada del método, en la línea 22 se inserta la información en el objeto que representa a la base de datos por medio del método `InsertOnSubmit`, mientras que la línea 23 actualiza los cambios realizados en la base de datos.

Las líneas 25 a 37 permiten agregar la información de una reunión a la base de datos, entre las líneas 29 a 34 se crea una instancia de la clase `Reunion`, y se inicializa la misma con los parámetros de entrada del método, en la línea 35 se inserta la información en el objeto que representa a la base de datos, mientras que la línea 36 actualiza los cambios realizados en la base de datos.

```
13      public class Service1 : IGestionCompromisos
14      {
15          BaseDatos miDB = new BaseDatos();
16
17          //Agregar
            1 referencia | 0 cambios | 0 autores, 0 cambios
18          public void AgregarTipo(string nombreTipo)
19          {
20              Tipo aux = new Tipo();
21              aux.nombreTipo = nombreTipo;
22              miDB.tblTipo.InsertOnSubmit(aux);
23              miDB.SubmitChanges();
24          }
25
            1 referencia | 0 cambios | 0 autores, 0 cambios
26          public void AgregarReunion(int idTipoReunion, int numeroReuni
27              DateTime fechaReunion, string temaReunion)
28          {
29              Reunion aux = new Reunion();
30              aux.idTipo = idTipoReunion;
31              aux.numeroReunion = numeroReunion;
32              aux.fechaReunion = fechaReunion;
33              aux.temaReunion = temaReunion;
34              aux.borrado = 0; //0 significa que no esta borrado
35              miDB.tblReunion.InsertOnSubmit(aux);
36              miDB.SubmitChanges();
37          }
```

Código 2.11 Método para agregar información

En el Código 2.12 se implementa la operación definida en el Código 2.7, la cual permite editar la información de una empresa. Entre las líneas 129 a 131 se busca un registro del tipo `tblEmpresa` almacenado en la base de datos cuyo identificador sea similar al del parámetro de entrada del método, en la línea 133 se modifica los atributos de dicha empresa con el resto de parámetros de entrada del método, mientras que la línea 134 actualiza los cambios realizados en la base de datos.

```
127     public void EditarEmpresa(int idEmpresa, string nombreEmpresa)
128     {
129         var iempresa = (from iter in miDB.tblEmpresa
130                         where iter.idEmpresa == idEmpresa
131                         select iter).SingleOrDefault();
132
133         iempresa.nombreEmpresa = nombreEmpresa;
134         miDB.SubmitChanges();
135     }
```

Código 2.12 Método para editar información

En el Código 2.13 se define la operación descrita en el Código 2.8, la cual permite leer la información de un compromiso. La línea 278 crea una lista de objetos de la clase `Compromiso`, las líneas 279 y 280 realizan una consulta a la base de datos acerca de todos los registros almacenados en `tblCompromiso`. Entre las líneas 282 a 285 se agrega cada resultado de la consulta previa a la lista creada en la línea 278, mientras que en la línea 286 se retorna la lista de la clase `Compromiso`.

```
276     public List<Compromiso> LeerCompromiso()
277     {
278         List<Compromiso> compromisos = new List<Compromiso>();
279         var listCompro = from aux in miDB.tblCompromiso
280                          select aux;
281
282         foreach (var iter in listCompro)
283         {
284             compromisos.Add(iter);
285         }
286         return compromisos;
287     }
```

Código 2.13 Método para leer información

En el Código 2.14 se presenta la operación definida en el Código 2.9, la cual permite eliminar la información de un compromiso .Entre las líneas 336 a 338 se busca un registro

de la tabla `tblCompromiso` cuyo identificador sea similar al del parámetro de entrada del método. La línea 339 elimina el registro de la base de datos por medio del comando `DeleteOnSubmit`, mientras que la línea 340 actualiza los cambios realizados en la base de datos.

```
334         public void EliminarCompromiso(int idCompromiso)
335         {
336             var icompromiso = (from iter in miDB.tblCompromiso
337                                where iter.idCompromiso == idCompromiso
338                                select iter).SingleOrDefault();
339             miDB.tblCompromiso.DeleteOnSubmit(icompromiso);
340             miDB.SubmitChanges();
341         }
```

Código 2.14 Método para eliminar información

En el Código 2.15 se puede ver la operación extra definida en el Código 2.10, el cual permite buscar todas reuniones realizadas entre un intervalo de fechas. La línea 468 crea una lista de objetos del tipo `Reunion`. Entre las líneas 470 a 473 se busca todos los registros de la tabla `tblReunion` cuya `fechaReunion` este entre la fecha de inicio y la fecha de fin dadas en los parámetros de entrada del método. Entre las líneas 474 a 477 se agrega cada resultado de la consulta previa a la lista creada en la línea 468, mientras que la línea 479 retorna la lista de la clase `Reunion`.

El código completo del prototipo incluido la codificación de la capa lógica de negocio se presenta en el ANEXO G.

```
466     public List<Reunion> BuscarReunionPorFechas(DateTime fechaInicio, DateTime fechaFin)
467     {
468         List<Reunion> reuniones = new List<Reunion>();
469
470         var listReunion = from aux in miDB.tblReunion
471                           where aux.fechaReunion >= fechaInicio && aux.fechaReunion <= fechaFin
472                           select aux;
473
474         foreach (var iter in listReunion)
475         {
476             reuniones.Add(iter);
477         }
478
479         return reuniones;
480     }
```

Código 2.15 Métodos extras

2.2.6 IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

Para la capa de presentación se incluyó en la solución de Visual Studio un proyecto nuevo. El proyecto es del tipo “Aplicación de Windows Forms”, como se observa en la Figura 2.38 y se denomina `Cliente`.

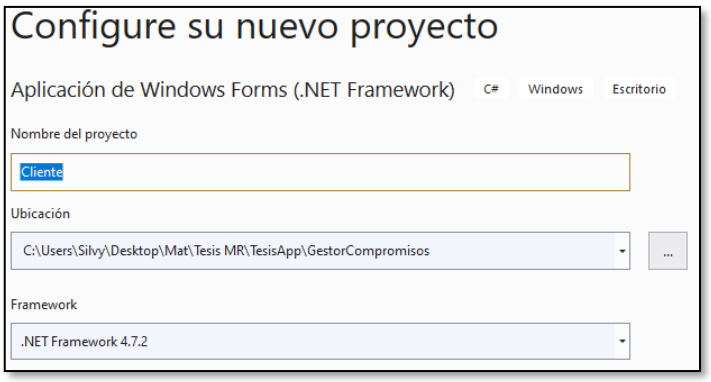

Figura 2.38 Creación del proyecto del tipo Aplicación de Windows Forms

Una vez creado el proyecto, Visual Studio desplegará un formulario como el que se observa en la Figura 2.39. Para diseñar la interfaz de usuario se puede arrastrar un control o componente visual específico desde el cuadro de herramientas o *toolbox*[27] (parte izquierda de la Figura 2.39).

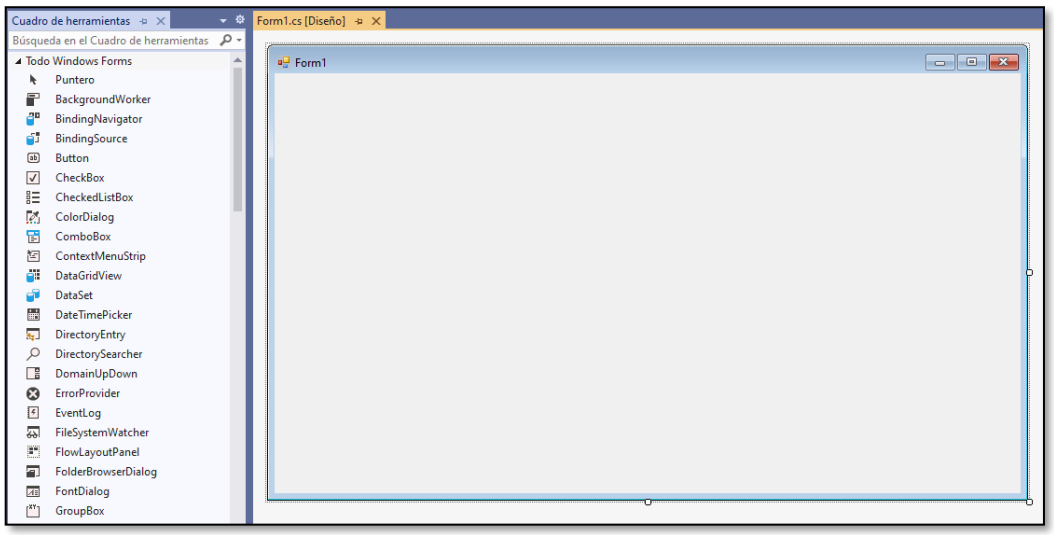

Figura 2.39 Windows Forms y *Toolbox*

[27] *Toolbox:* es una herramienta de Visual Studio que ofrece al desarrollador un conjunto de componentes visuales para su uso en formularios Windows.

2.2.6.1 Agregar referencia a servicio

Para poder acceder al servicio creado en la sección 2.2.5, se agrega la referencia al servicio en el proyecto del cliente, para ello mediante la opción del cliente denominada “Servicios Conectados”, presionando clic derecho sobre este, se desplegará una lista de opciones en donde se encuentra “Agregar referencia de servicio…”, como se observa en la Figura 2.40.

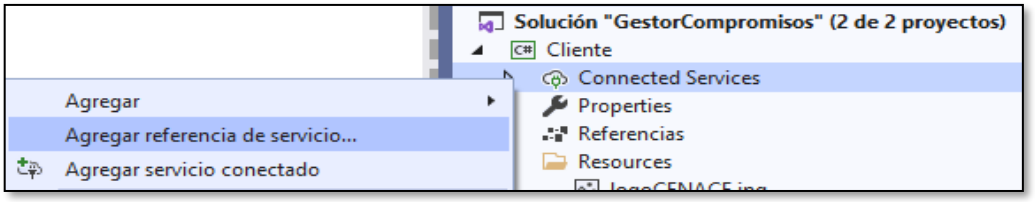

Figura 2.40 Agregar referencia de servicio (Parte 1)

Para agregar la referencia de servicio se requiere conocer la dirección del servicio, como en este ejemplo el servicio se encuentra dentro de la misma solución al presionar en el botón Detectar, Visual Studio detectará e ingresará automáticamente la URL del servicio. Adicionalmente se puede establecer el nombre que tendrá el espacio de nombres luego de lo cual se debe presionar en el botón Aceptar. En la Figura 2.41 se presenta la ventana que permite agregar la referencia de servicio.

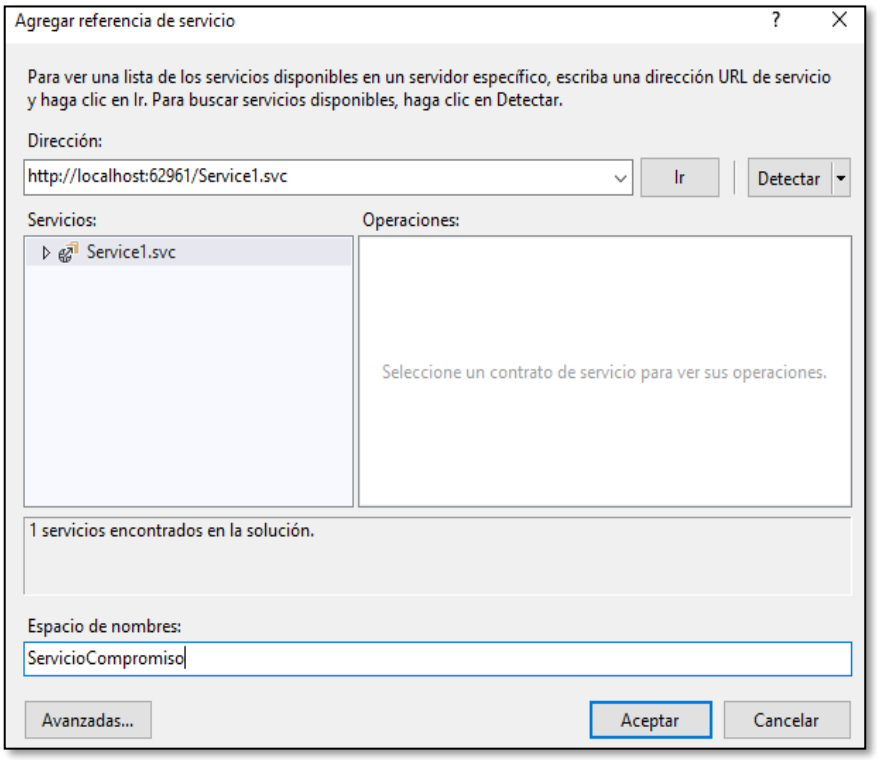

Figura 2.41 Agregar referencia de servicio (Parte 2)

2.2.6.2 Codificación de las interfaces de usuario

En el proyecto `Cliente` se crearon doce formularios, como se observa en la Figura 2.42.

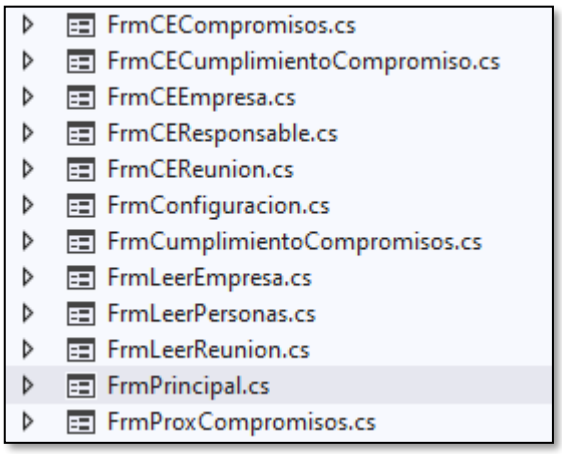

Figura 2.42 Formularios del proyecto `Cliente`

Los formularios `FrmCECompromisos` y `FrmCECumplimientoCompromiso` permiten al usuario poder agregar y editar la información de los compromisos de una reunión y el cumplimiento de un compromiso, respectivamente. Los formularios `FrmCEEmpresa`, `FrmCEResponsable` y `FrmCEReunion` permiten agregar y modificar la información de una empresa, responsable y reunión respectivamente.

El formulario `FrmCumplimientoCompromisos` permite ver los compromisos de una reunión, así como registrar su cumplimiento mediante el formulario `FrmCECumplimientoCompromiso`; mientras que el formulario `FrmPrxoCompromisos` permite observar los compromisos que están próximos a cumplirse.

Los formularios `FrmLeerEmpresa`, `FrmLeerPersona` y `FrmLeerReunion` permiten al usuario poder realizar búsquedas de los registros de empresas, responsables y reuniones respectivamente.

Finalmente, el formulario `FrmPrincipal` permite al usuario ingresar a todos los módulos de gestión de la aplicación y sirve como formulario padre de los formularios presentados.

La Figura 2.43 muestra el formulario `FrmCEReunion` en donde se emplearon controles de tipo `Label`, `TextBox`, `ComboBox`, `DateTimePicker` y `Button` para mostrar texto, ingresar texto, seleccionar una opción de una lista, seleccionar una fecha y realizar una acción, respectivamente. Para presentar los resultados de una búsqueda se emplea el control `ListView`. El resto de formularios emplean los mismos controles.

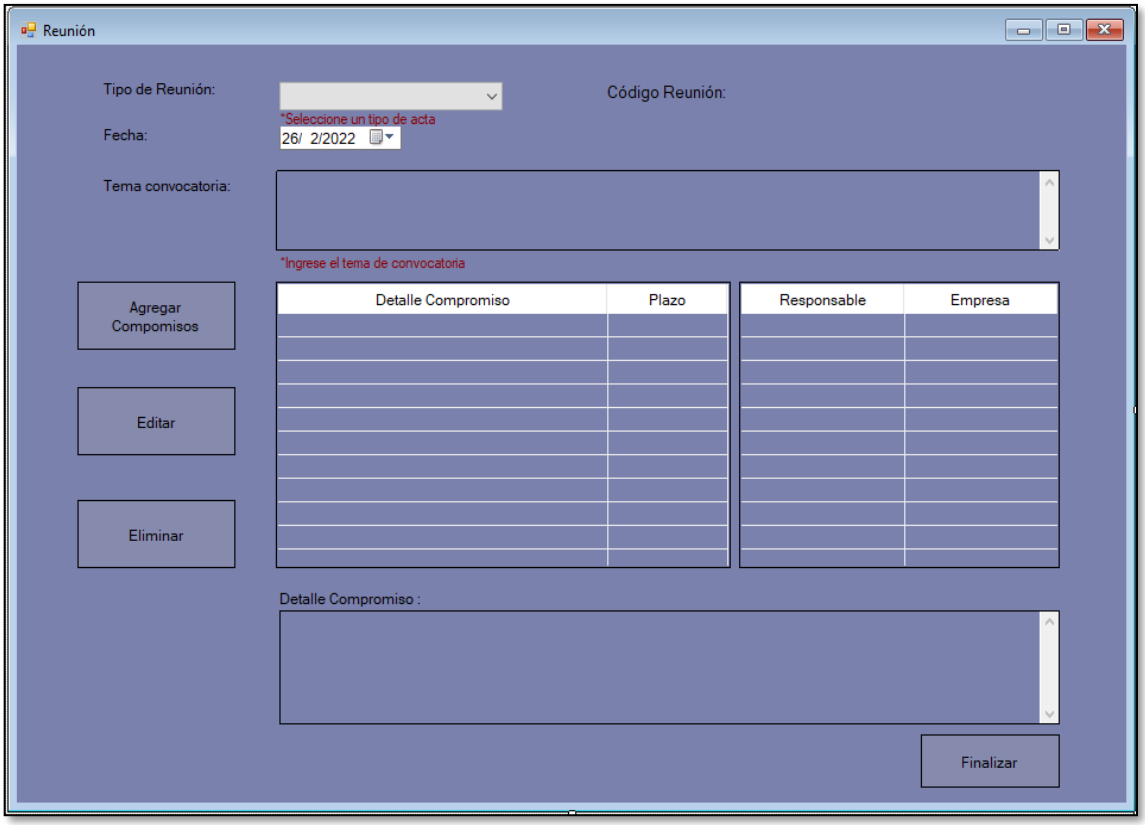

Figura 2.43 Formulario `FrmCEReunion`

El formulario `FrmCEReunion` se utiliza para agregar y editar la información de una reunión.

Para poder emplear el servicio agregado en la sección 2.2.6.1, se debe crear un objeto proxy que permita la comunicación con el servicio, como se observa en el Código 2.16. En la línea 15 se muestra la creación de la instancia del objeto proxy que realizará la comunicación con el servicio previamente agregado. Las líneas del 17 al 20 inicializan el formulario con todos sus componentes.

```
11  namespace Cliente
12  {
        7 referencias | 0 cambios | 0 autores, 0 cambios
13      public partial class FrmCEReunion : Form
14      {
15          ServicioCompromisos.GestionCompromisosClient cliente;
16
            0 referencias | 0 cambios | 0 autores, 0 cambios
17          public FrmCEReunion()
18          {
19              InitializeComponent();
20          }
```

Código 2.16 Instancia de la referencia del servicio

Debido a que este formulario permite agregar y modificar la información de una reunión, se definen estados de forma que el formulario pueda realizar ciertas actividades cuando el usuario desee agregar una nueva reunión o cuando desee modificar la reunión. Los estados se gestionan en el constructor, como se aprecia en el Código 2.17, en particular en las líneas 35 a 40 se define el constructor del formulario el cual acepta dos parámetros de entrada, el estado, que puede ser para agregar reunión o para editar reunión y el identificador de la reunión, que en caso de edición será necesario.

```
35          public FrmCEReunion(string iniestado, int idReunion)
36          {
37              this.estado = iniestado;
38              this.idReunionEditar = idReunion;
39              InitializeComponent();
40          }
```

Código 2.17 Constructor del formulario `FrmCEReunion`

Mediante el manipulador del evento `Load` del formulario (`FrmCEReunion_Load`), el cual se muestra en el Código 2.18, se realiza lo siguiente: en la línea 45 se inicializa el objeto proxy, en las líneas 45 a 46 se ocultan los controles que presentarán las alertas, los cuales aparecerán en caso de que no se llene de manera adecuada la información de una reunión.

Finalmente, las líneas 47 a 54, con base en el estado, determinará el método que se ejecutará para agregar una reunión o para editar una reunión.

```
42          private void FrmCEReunion_Load(object sender, EventArgs e)
43          {
44              cliente = new ServicioCompromisos.GestionCompromisosClient();
45              lblAlertaTemaConvocatoria.Hide();
46              lblAlertaTipoActa.Hide();
47              if(estado == "agregarReunion")
48              {
49                  EstadoAgregarReunion();
50              }
51              else if (estado == "editarReunion")
52              {
53                  EstadoEditarReunion();
54              }
55          }
```

Código 2.18 Manipulador del evento `Load` del formulario `FrmCEReunion`

Si se desea agregar una reunión se habilitarán y mostrarán todos los controles del formulario, como se muestra en el Código 2.19, adicionalmente en la línea 74 se llama al

método `CargarTipoActa`, el cual, mediante el objeto proxy se comunica con el servicio WCF para realizar una consulta a la base de datos.

```
57      public void EstadoAgregarReunion()
58      {
59          //Todo activado
60          //
61          txtTemaConvo.BackColor = Color.FromArgb(123,129,173);
62          //
63          //Informacion general habilitado
64          cbxTipoActa.Enabled = true;
65          dtpFechaReunion.Enabled = true;
66          txtTemaConvo.Enabled = true;
67          btnAgregarCompro.Enabled = true;
68          lvxCompromisos.Show();
69          lvxResponsables.Show();
70          btnEditarCompromiso.Show();
71          btnEliminarCompromiso.Show();
72          lblDetalle.Show();
73          txtDetalleCompromiso.Show();
74          CargarTipoActa();
75      }
```

Código 2.19 Método `EstadoAgregarReunion`

En el Código 2.20 se muestra el método `CargarTipoActa`; en la línea 140 se limpia toda la información que se encuentre en el control de tipo `comboBox`, en la línea 141 se utiliza el método `LeerTipo` del objeto proxy para acceder al servicio WCF y recuperar los registros de la tabla `tblTipo`, mientras que en la línea 143 se carga el control de tipo `comboBox` con los nombres del tipo de reunión obtenido mediante el objeto proxy.

```
138     public void CargarTipoActa()
139     {
140         cbxTipoActa.Items.Clear();
141         foreach (var iter in cliente.LeerTipo())
142         {
143             cbxTipoActa.Items.Add(iter.nombreTipo);
144         }
145     }
```

Código 2.20 Método `CargarTipoActa`

Para agregar una reunión, el usuario podrá seleccionar el tipo de reunión definidas, esto es “Reunión Técnica” o “Comité de Análisis de falla”, indicar la fecha de la reunión, ingresar

el tema de convocatoria y agregar, editar o eliminar un compromiso. Al momento de seleccionar un tipo de reunión, se debe generar de manera automática el código de reunión, para ello se usa el manipulador del evento `SelectedIndexChanged` del control del tipo `comboBox` (`cbxTipoActa_SelectedIndexChanged`) como se indica el Código 2.21.

En la línea 149 se oculta el control que gestiona la alerta relacionada con no haber seleccionado una reunión. En la línea 150 se comprueba si el estado del formulario es para agregar una reunión, en caso de que cumpla dicha condición se cumplirán las instrucciones descritas en las líneas 150 a 164.

En las líneas 152 y 158, se usa el método `ToString` del control tipo `comboBox` para conocer el nombre del tipo de reunión que ha seleccionado el usuario y conocer si esta fue "Reunión Técnica" o "Comité de Análisis de Falla", respectivamente.

Si el usuario seleccionó la opción "Reunión Técnica" en el control del tipo `comboBox`, se realiza lo siguiente: en la línea 154 se inicializará el atributo `idTipo` el cual es el identificador de la clase `Tipo` para el registro perteneciente a "Reunión Técnica", en la línea 155 se utiliza el método `NumeroReunionSegunTipo` del objeto proxy para acceder al servicio WCF y obtener el número de reunión correspondiente a "Reunión Técnica", en la línea 156 por medio del método `Text` del control tipo `textBox` se escribe el código de la reunión correspondiente.

Si el usuario seleccionó la opción "Comité de Análisis de Falla" en el control del tipo `comboBox`, se realiza lo siguiente: en la línea 160 se inicializará el atributo `idTipo` el cual es el identificador de la clase `Tipo` para el registro perteneciente a "Comité de Análisis de Falla", en la línea 161 se utiliza el método `NumeroReunionSegunTipo` del objeto proxy para acceder al servicio WCF y obtener el número de reunión correspondiente a "Comité de Análisis de Falla", en la línea 162 por medio del método `Text` del control tipo `textBox` se escribe el código de la reunión correspondiente.

Para agregar un compromiso el usuario dará *click* sobre el botón "Agregar Compromiso" del formulario `FrmCEReunion`, lo cual llamará al manipulador del evento `Click` del botón (`btnAgregarCompro_Click`), el cual se muestra en el Código 2.22.

En la línea 197 se comprueba si el estado del formulario es para agregar una reunión, en caso de que cumpla dicha condición se cumplirán las instrucciones descritas en las líneas 198 a 206. En la línea 199 se crea un objeto del tipo `FrmCECompromisos` que acepta un parámetro de entrada el cual representa un estado, ya que el formulario

`FrmCECompromisos` se usa tanto para agregar y editar los compromisos, por lo cual es necesario especificar a acción que se va a realizar.

En la línea 200 se utiliza el método `ShowDialog` para mostrar en pantalla el formulario creado en la línea 199, mientras que las líneas 201 a 205 se añade el registro creado en el formulario `FrmCECompromisos` a una lista de compromisos.

```
147         private void cbxTipoActa_SelectedIndexChanged(object sender, EventArgs e)
148         {
149             lblAlertaTipoActa.Hide();
150             if (estado == "agregarReunion")
151             {
152                 if (cbxTipoActa.SelectedItem.ToString() == "Reunión Técnica")
153                 {
154                     idTipo = 1;
155                     numeroReunion = cliente.NumeroReunionSegunTipo(idTipo, DateTime.Now);
156                     txtNumActa.Text = "RT-" + numeroReunion + "-" + dtpFechaReunion.Value.Year.ToString();
157                 }
158                 else if (cbxTipoActa.SelectedItem.ToString() == "Comité de Análisis de Falla")
159                 {
160                     idTipo = 2;
161                     numeroReunion = cliente.NumeroReunionSegunTipo(idTipo, DateTime.Now);
162                     txtNumActa.Text = "CAF-" + numeroReunion + "-" + dtpFechaReunion.Value.Year.ToString();
163                 }
164             }
```

Código 2.21 Manipulador del evento `SelectedIndexChanged` del control `cbxTipoActa`

```
194         private void btnAgregarCompro_Click(object sender, EventArgs e)
195         {
196             
197             if (estado == "agregarReunion")
198             {
199                 FrmCECompromisos frmCECompromisos = new FrmCECompromisos("agregar");
200                 frmCECompromisos.ShowDialog();
201                 if(frmCECompromisos.Aux != null)
202                 {
203                     listaCompromiso.Add(frmCECompromisos.Aux);
204                     ActualizarAcuerdos();
205                 }
206             }
```

Código 2.22 Manipulador del evento *Click* del control `btnAgregarCompro`

Para editar un compromiso el usuario dará *click* sobre el botón “Editar Compromiso” del formulario `FrmCEReunion`, lo cual llamará al manipulador del evento `Click` del botón (`btnEditarCompromiso_Click`), el cual se muestra en el Código 2.23.

En la línea 260 se comprueba mediante el método `SelectedItems.Count` del control `listView` si el usuario ha seleccionado un compromiso de la lista para editarlo, en caso de que cumpla dicha condición se cumplirán las instrucciones descritas en las líneas 262 a 266, caso contrario se cumplirá la instrucción de la línea 271 la cual mostrará un mensaje de error.

En las líneas 262 a 263 se obtiene el número de registro del compromiso seleccionado del control `listView` por medio del método `SelectedItems[0]`, en la línea 264 se crea un objeto del tipo `FrmCECompromisos` que acepta tres parámetros de entrada: un estado, la lista de compromisos y el identificador del compromiso a editar.

En la línea 265 se utiliza el método `ShowDialog` para mostrar en pantalla el formulario creado en la línea 264, adicionalmente en la línea 266 se llama al método `ActualizarAcuerdos`, el cual actualiza los registros del control `listView`.

```
258     private void btnEditarCompromiso_Click(object sender, EventArgs e)
259     {
260         if (lvxCompromisos.SelectedItems.Count > 0)
261         {
262             ListViewItem item = lvxCompromisos.SelectedItems[0];
263             int idAux = Convert.ToInt32(item.Text);
264             FrmCECompromisos frmCECompromisos = new FrmCECompromisos("editar", listaCompromiso, idAux);
265             frmCECompromisos.ShowDialog();
266             ActualizarAcuerdos();
267 
268         }
269         else
270         {
271             MessageBox.Show("Seleccione un compromiso para editar");
272         }
273     }
```

Código 2.23 Manipulador del evento *Click* del control `btnEditarCompromiso`

Para eliminar un compromiso el usuario dará *click* sobre el botón “Eliminar Compromiso” del formulario `FrmCEReunion`, lo cual llamará al manipulador del evento `Click` del botón (`btnEliminarCompromiso_Click`), el cual se muestra en el Código 2.24.

En la línea 277 se comprueba mediante el método `SelectedItems.Count` del control `listView` si el usuario ha seleccionado un compromiso de la lista para eliminarlo, en caso de que cumpla dicha condición se cumplirán las instrucciones descritas en las líneas 280 a 292, caso contrario se cumplirá la instrucción de la línea 296 la cual mostrará un mensaje de error.

En las líneas 280 a 281 se obtiene el número de registro del compromiso seleccionado del control `listView` por medio del método `SelectedItems [0]`, en la línea 282 se crea un objeto del tipo `Compromiso`, entre las líneas 283 a 290 se recorre la lista de registros del control `listView`, en la línea 285 se comprueba si el identificador de cada registro del control `listView` sea igual al número de registro que se obtuvo en la línea 281. Si es igual, se instancia el objeto de la línea 282 con la información del registro seleccionado.

Una vez finalizada la lista de registros del control `listView` o encontrado el registro el cual el usuario selecciono para eliminar, en la línea 291 se elimina el compromiso seleccionado de la lista de compromisos de la reunión por medio del método `Remove`, en la línea 292 se llama al método `ActualizarAcuerdos` el cual actualiza los registros del control `listView`.

Tanto para los eventos de agregar, editar y eliminar compromisos se llama a un método `ActualizarAcuerdos` que se muestra en el Código 2.25. En las líneas 221 a 223 se limpian todas las listas mediante el método `Clear`, mientras que en las líneas 224 a 230 se llena el componente `ListView` por cada compromiso que se encuentre en la lista, mediante los métodos `Items.Add` y `SubItems.Add`.

```
275     private void btnEliminarCompromiso_Click(object sender, EventArgs e)
276     {
277         if (lvxCompromisos.SelectedItems.Count > 0)
278         {
279
280             ListViewItem item = lvxCompromisos.SelectedItems[0]; ;
281             int idAux = Convert.ToInt32(item.Text); //El ID del item
282             Compromiso auxCompromiso = new Compromiso();
283             foreach (var iter in listaCompromiso)
284             {
285                 if (iter.IdCompromiso == idAux)
286                 {
287                     auxCompromiso = iter;
288                     break;
289                 }
290             }
291             listaCompromiso.Remove(auxCompromiso);
292             ActualizarAcuerdos();
293         }
294         else
295         {
296             MessageBox.Show("Seleccione un compromiso para eliminar");
297         }
298     }
```

Código 2.24 Manipulador del evento *Click* del control `btnEliminarCompromiso`

```
219         public void ActualizarAcuerdos()
220         {
221             lvxCompromisos.Items.Clear();
222             lvxResponsables.Items.Clear();
223             txtDetalleCompromiso.Clear();
224             foreach (var iter in listaCompromiso)
225             {
226                 ListViewItem item = new ListViewItem();
227                 item = lvxCompromisos.Items.Add(Convert.ToString(iter.IdCompromiso));
228                 item.SubItems.Add(iter.Detalle);
229                 item.SubItems.Add(iter.Plazo.Date.ToString("dd/MM/yyyy"));
230             }
231         }
```

Código 2.25 Método `ActualizarAcuerdos`

Finalmente, para agregar la reunión, el usuario va a dar *click* sobre el botón “Finalizar” del formulario `FrmCEReunion`, lo cual llamará al manipulador del evento `Click` del botón (`btnFinalizar_Click`), el cual se muestra en el Código 2.26.

En la línea 302 se comprueba si el estado del formulario es para agregar una reunión, en caso de que cumpla dicha condición se cumplirán las instrucciones descritas en las líneas 304 a 326. La línea 304 verifica que se no falte ningún dato por medio del método `ComprobarDatos`.

En la línea 307 se utiliza el método `AgregarReunion` del objeto proxy para acceder al servicio WCF y agregar una nueva reunión a los registros de la tabla `tblReunion`, en la línea 309 se utiliza el método `LeerReunion` del objeto proxy para acceder al servicio WCF y recuperar los registros de la tabla `tblReunion` y con la ayuda del método `Last` se obtiene el último registro de la tabla `tblReunion`.

Entre las líneas 311 a 324 se recorre cada uno de los registros de la lista de objetos del tipo `Compromiso` para ser asociados a la última reunión ingresada y almacenada en la base de datos con ayuda del objeto proxy. En la línea 315 se utiliza el método `AgregarCompromiso` del objeto proxy para acceder al servicio WCF y agregar un nuevo compromiso a los registros de la tabla `tblCompromiso`, en la línea 316 se utiliza el método `LeerCompromiso` del objeto proxy para acceder al servicio WCF y recuperar los registros de la tabla `tblCompromiso` y con la ayuda del método `Last` se obtiene el último registro ingresado de la tabla `tblCompromiso`.

Entre las líneas 319 a 323 se recorre cada uno de los registros de la lista de objetos del tipo `ResponsableCompromiso` para ser asociados al último compromiso ingresado y

almacenado en la base de datos con ayuda del objeto proxy. En la línea 321 se utiliza el método `AgregarResponsableCompromiso` del objeto proxy para acceder al servicio WCF y agregar un nuevo responsable del compromiso a los registros de la tabla `tblResponsableCompromiso`.

Finalmente, en la línea 339 se cierra el formulario `FrmCEReunion` mediante el método `Close`.

```
300     private void btnFinalizar_Click(object sender, EventArgs e)
301     {
302         if (estado == "agregarReunion")
303         {
304             if (ComprobarDatos())
305             {
306                 //Se agrega la reunión en la db
307                 cliente.AgregarReunion(idTipo, numeroReunion, dtpFechaReunion.Value.Date,
308                     txtTemaConvo.Text);
309                 int idReunion = cliente.LeerReunion().Last().idReunion;
310                 //Se agregan los compromisos de la reunión
311
312                 foreach (var iter in listaCompromiso)
313                 {
314
315                     cliente.AgregarCompromiso(idReunion, iter.Detalle, iter.Plazo.Date);
316                     int idCompromiso = cliente.LeerCompromiso().Last().idCompromiso;
317
318                     //Se agrega la tabla responsable - compromiso
319                     foreach (var iter2 in iter.ResponsableCompromisos)
320                     {
321                         cliente.AgregarResponsableCompromiso(iter2.IdResponsable, idCompromiso);
322
323                     }
324                 }
325                 this.Close();
326             }
327         }
```

Código 2.26 Manipulador del evento *Click* del control `btnFinalizar`

El código completo del prototipo incluido el código de los diferentes formularios que conforman la capa de presentación se encuentra en el ANEXO G

2.2.7 ALOJAMIENTO DEL SERVICIO WCF

Una vez comprobado que el prototipo funciona de manera deseada, se alojó el servicio WCF en un servidor IIS para Windows 10.

Para publicar el servicio, en el Visual Studio se escoge el proyecto `GestorCompromiso` en el explorador de soluciones. Una vez seleccionado, se presiona clic derecho sobre el

nombre del proyecto para que se despliegue una lista de opciones, en la cual se debe escoger la opción “Publicar” (Ver Figura 2.44).

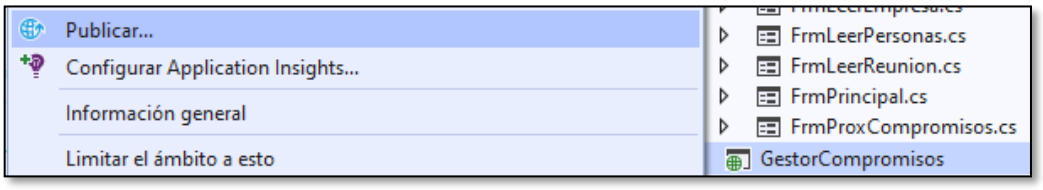

Figura 2.44 Publicación del servicio (Parte 1)

Una vez seleccionado la opción “Publicar”, se desplegará una ventana, como la que se muestra en la Figura 2.45. En esta ventana se debe escoger el destino en el que será publicado el servicio. En este caso se escogerá la opción “carpeta”, para publicar el servicio WCF en un recurso compartido de archivos del servidor IIS.

Una vez escogida la opción para publicar el servicio, se debe especificar la ubicación de la carpeta compartida, como se observa en la Figura 2.46. Es importante mencionar que para poder publicar el servicio de manera correcta se debe iniciar Visual Studio como administrador del sistema, ya que al publicar el servicio se requieren permisos de administrador.

Finalmente, Visual Studio desplegará una ventana con el resumen de las acciones realizadas, y que permite actualizar los detalles de ser necesario o publicar el servicio al presionar en el botón Publicar, situado en la parte superior izquierda (ver la Figura 2.47).

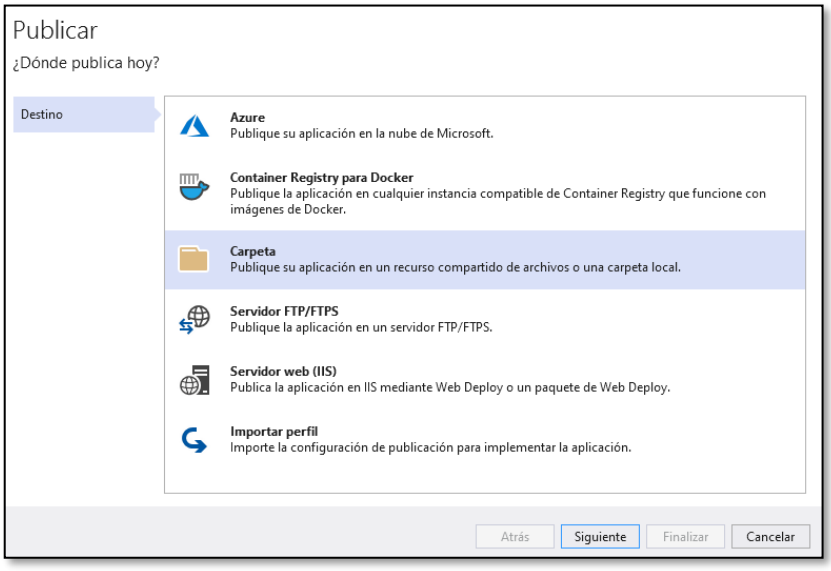

Figura 2.45 Publicación del servicio (Parte 2)

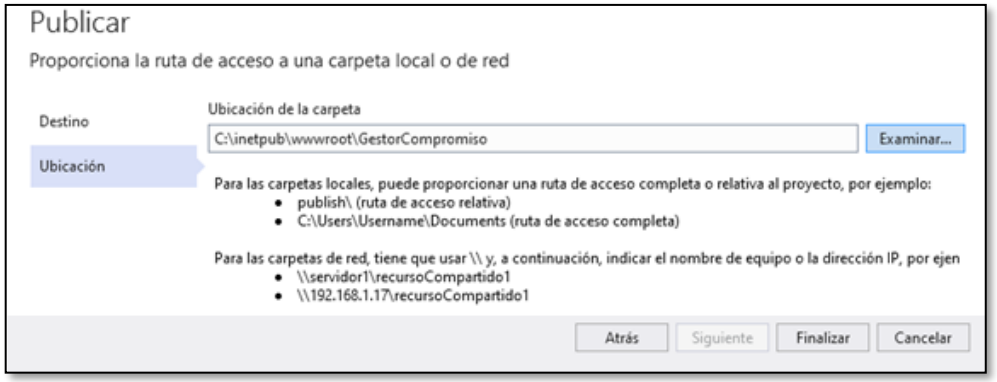

Figura 2.46 Publicación del servicio (Parte 3)

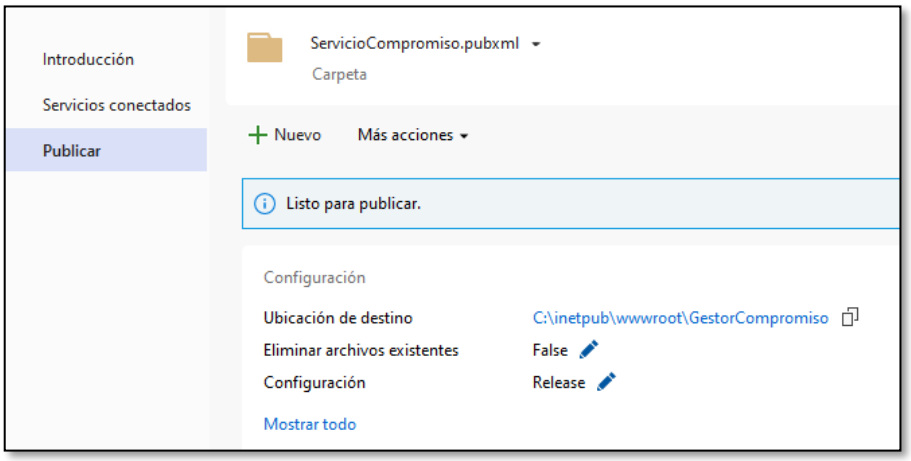

Figura 2.47 Publicación del servicio (Parte 4)

Para comprobar que el servicio se ha publicado de manera correcta, se puede revisar el Administrador de IIS. En el Administrador de IIS, se presentará la carpeta que contiene el código del proyecto. En dicha carpeta hay que presionar botón derecho y luego escoger la opción “Convertir en aplicación”, como se visualiza en la Figura 2.48.

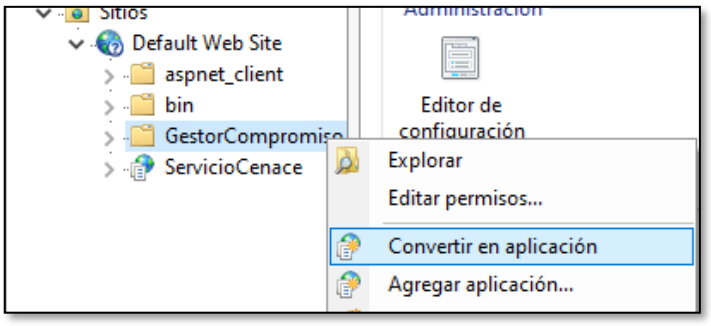

Figura 2.48 Administrador de IIS

Al seleccionar la opción “Convertir en aplicación” se despliega una ventana que permite modificar algunas variables para la aplicación, como indica la Figura 2.49.

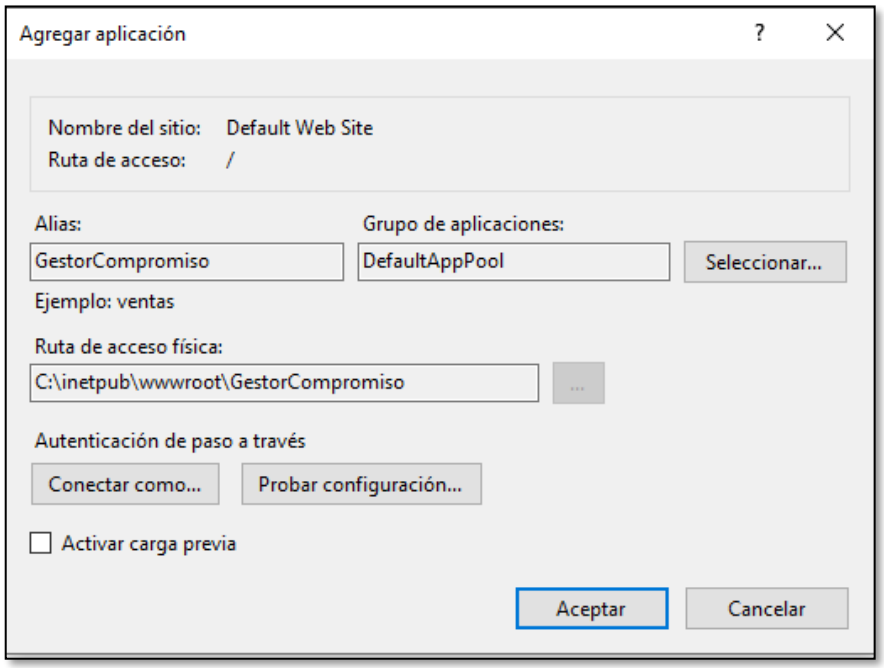

Figura 2.49 Opción de IIS para agregar la aplicación

Una vez convertido en aplicación, se puede examinar el servicio escogiendo en la parte izquierda la carpeta de la aplicación, luego de lo cual en la parte derecha, en las acciones se debe escoger “Examinar aplicación”, como se observa en la Figura 2.50.

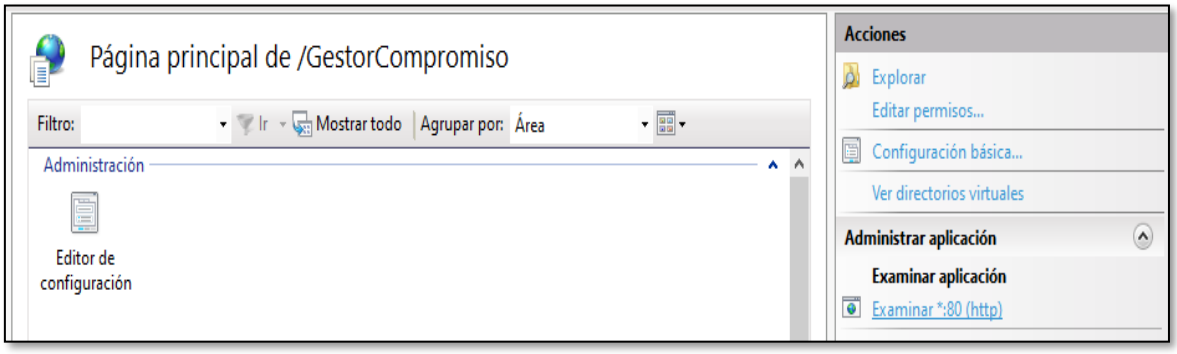

Figura 2.50 Examinar aplicación

Una vez presionado en la opción “Examinar aplicación” se desplegará información del servicio WCF en el explorador web predeterminado, como se muestra en la Figura 2.51.

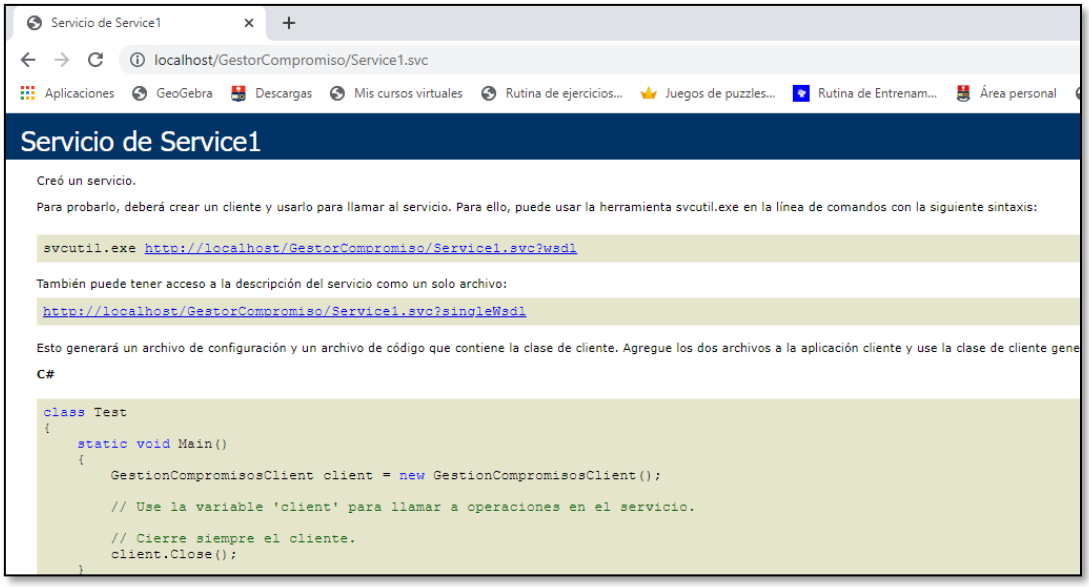

Figura 2.51 Comprobación de la publicación del servicio

Una vez comprobado la correcta publicación del servicio, es necesario actualizar la referencia de servicio del cliente, para que el cliente use el servicio WCF publicado en IIS. Para actualizar la referencia de servicio se deben realizar las acciones indicadas en la sección 2.2.6.1, pero especificando en la URL la dirección del servicio WCF que se presentó en el explorador web de la Figura 2.51, como se muestra en la Figura 2.52.

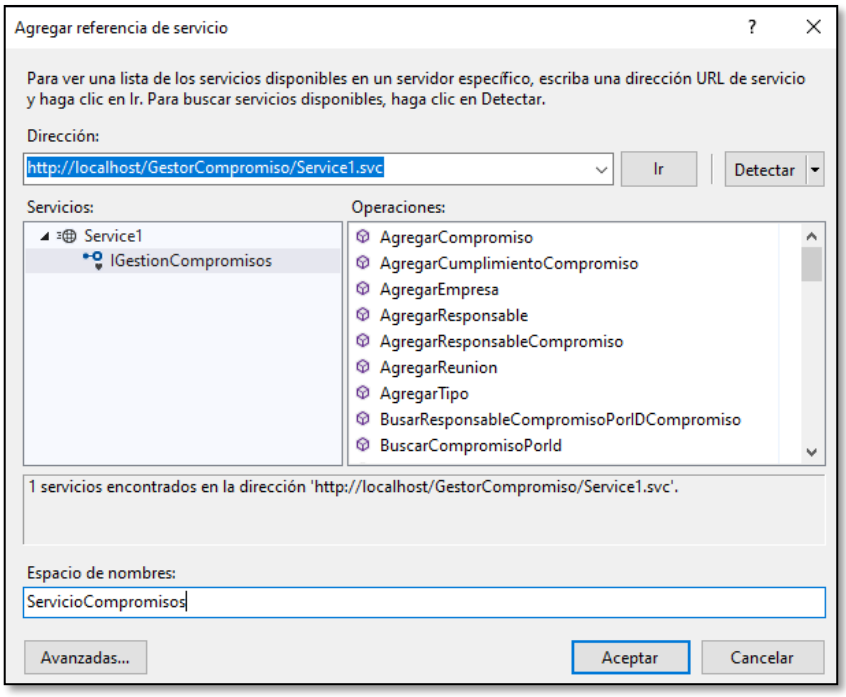

Figura 2.52 Agregar referencia al servicio publicado

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan los resultados más relevantes de las pruebas de funcionamiento del prototipo, las cuales fueron realizadas por parte de los miembros del área de análisis de operaciones de CENACE. Además de la encuesta de satisfacción realizada a cinco miembros de CENACE con los resultados obtenidos. Finalmente, se presentarán errores encontrados durante las pruebas y su respectiva corrección.

3.1 PRUEBAS DE FUNCIONAMIENTO

En esta sección se presentan las pruebas de funcionamiento del prototipo realizadas. Se efectuaron pruebas con la base de datos por medio de consultas realizadas desde la aplicación del cliente, hacia el servicio WCF al usar sus diferentes métodos y el funcionamiento de toda la lógica presente en la aplicación.

3.1.1 AGREGAR O MODIFICAR REUNIÓN

La Figura 3.1 muestra el formulario para agregar o modificar una reunión y toda la información relacionada con el mismo. Para agregar una reunión el usuario debe escoger un tipo de reunión y con ello se genera de manera automática el código de reunión, como se muestra en la Figura 3.2, con lo que se valida uno de los requerimientos descritos en la sección 2.1.1.

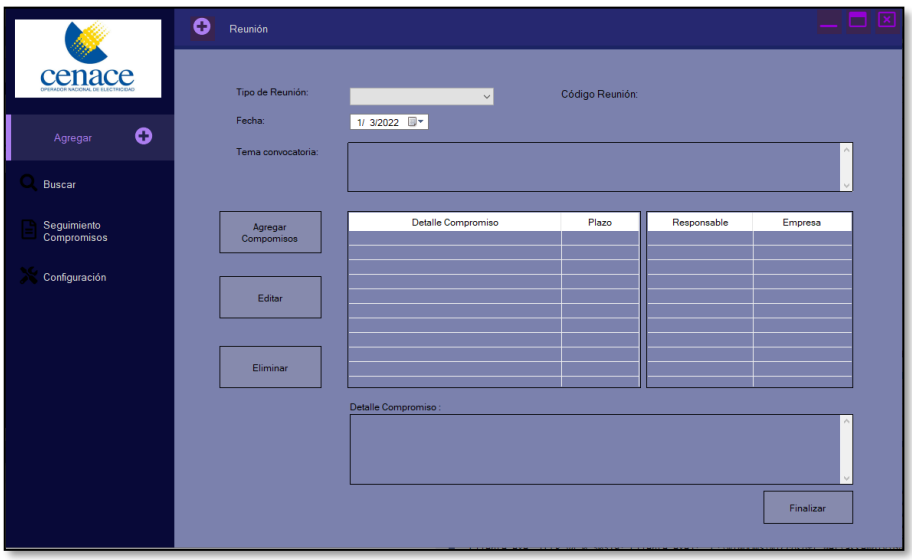

Figura 3.1 Interfaz gráfica para agregar reunión

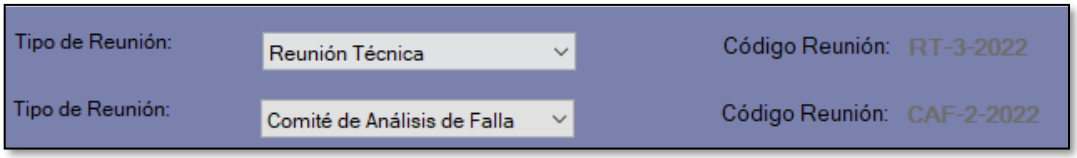

Figura 3.2 Automatización del código de reunión según el tipo de reunión

El usuario no podrá agregar una reunión si esta no posee un tipo de reunión, fecha, tema y por lo menos un compromiso, como se evidencia en la Figura 3.3. Si se completa toda la información necesaria de la reunión, el prototipo guarda la información en la base de datos al presionar clic en el botón finalizar.

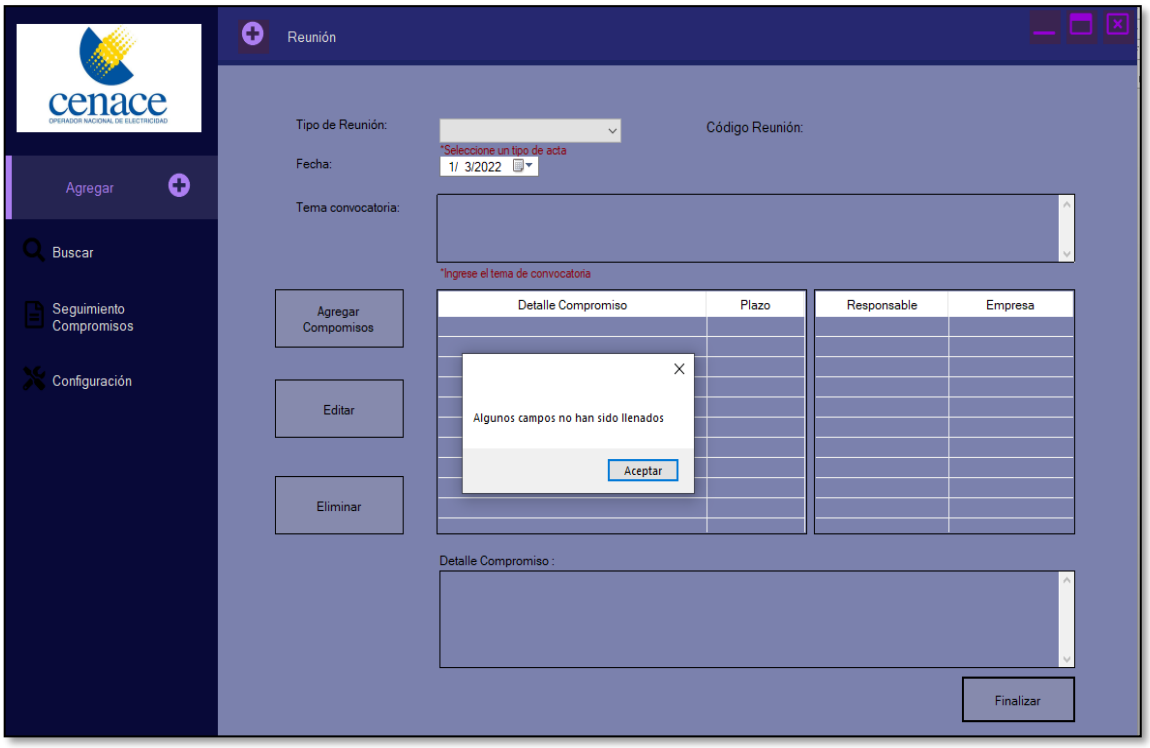

Figura 3.3 Mensaje de error al agregar una reunión

3.1.2 AGREGAR O MODIFICAR COMPROMISOS

Al momento de agregar o modificar una reunión se pueden agregar uno o varios compromisos acordados en esta reunión. La Figura 3.4 muestra el formulario para agregar un compromiso y toda la información relacionada con el mismo. Para agregar el compromiso el usuario debe llenar la información correspondiente al detalle, plazo y por lo menos un responsable del compromiso; caso contrario no se podrá agregar con éxito, como se muestra en la Figura 3.5.

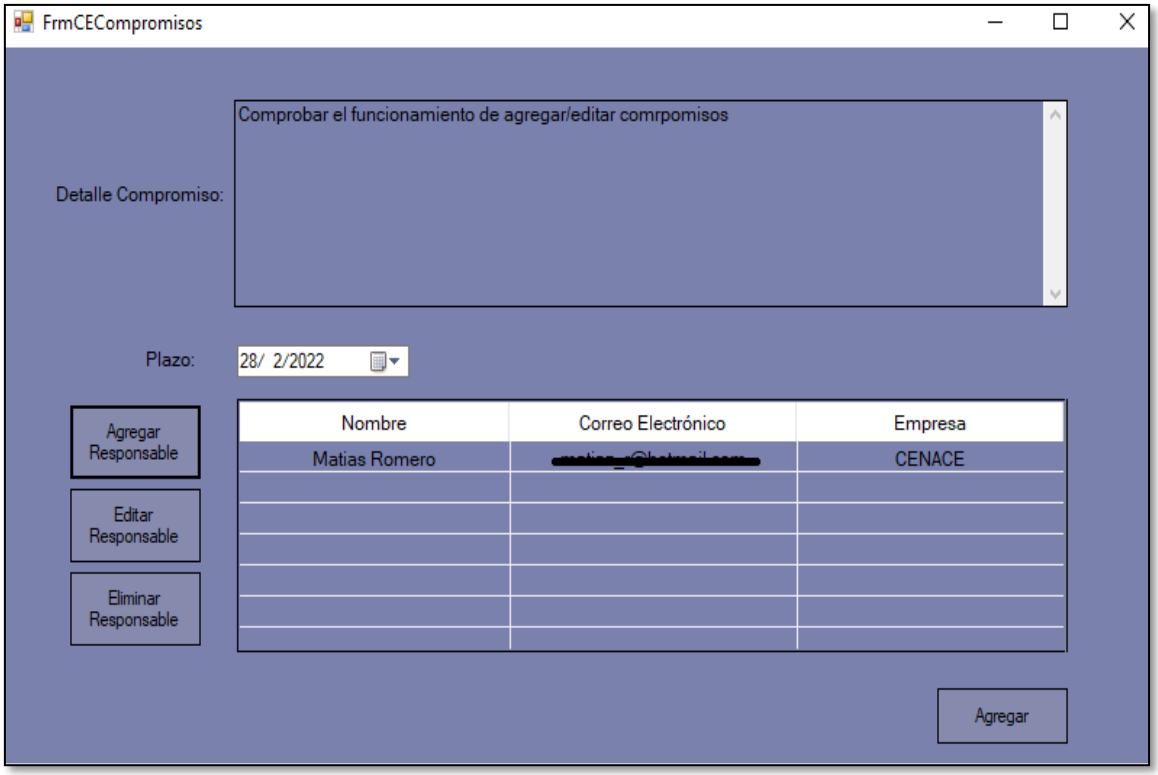

Figura 3.4 Interfaz gráfica para agregar compromisos

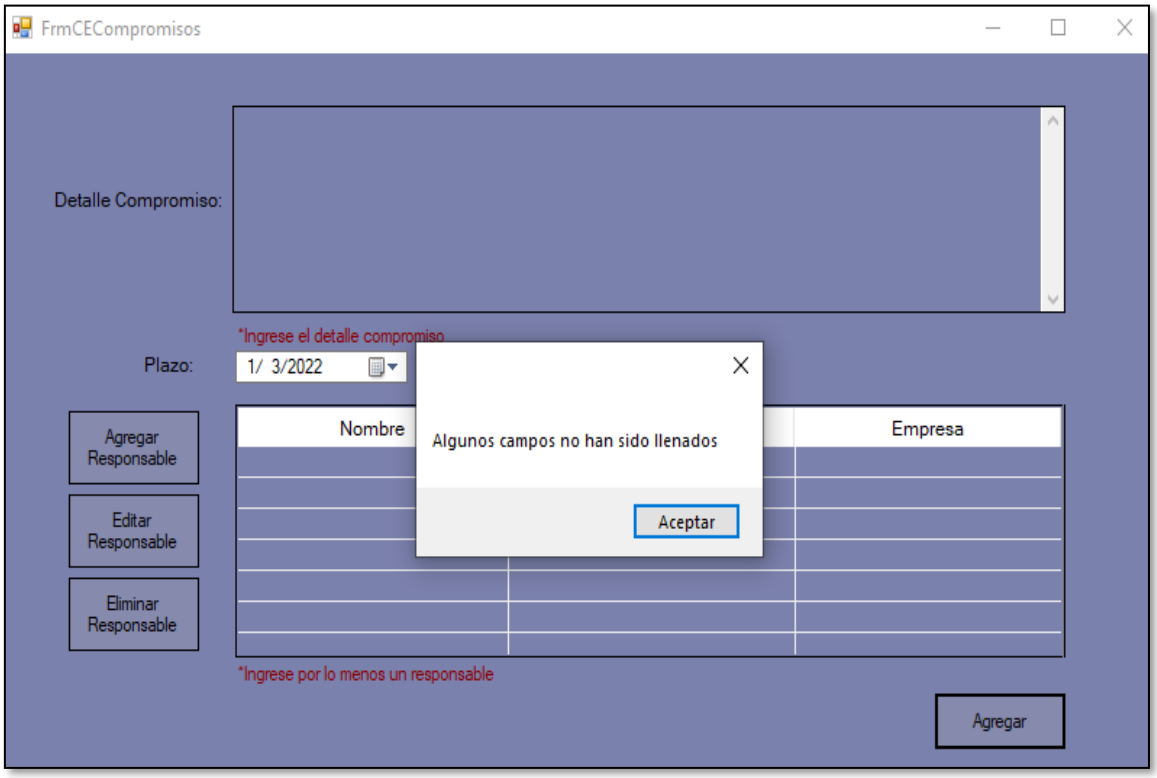

Figura 3.5 Mensaje de error al agregar compromisos

3.1.3 AGREGAR O MODIFICAR RESPONSABLES

Al momento de agregar o modificar un compromiso de una reunión se requiere agregar uno o varios responsables encargados del compromiso. La Figura 3.6 muestra el formulario para agregar un responsable, mediante el cual el usuario puede buscar un responsable y asociarlo con el compromiso, o puede agregar un nuevo responsable. Al agregar un responsable nuevo, el usuario debe llenar la información correspondiente al nombre, apellido, número de cédula, correo electrónico y o bien debe escoger una empresa crearla.

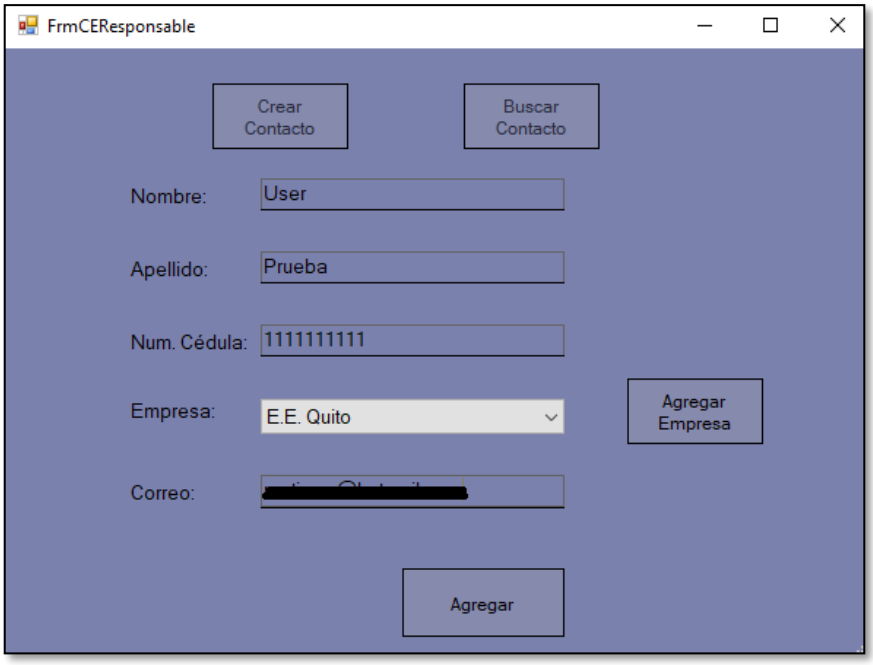

Figura 3.6 Interfaz gráfica para agregar nuevo responsable

En caso de que el usuario ingrese un número de cédula que ya haya sido ingresado, el prototipo presentará un mensaje de error como el que se indica en la Figura 3.7.

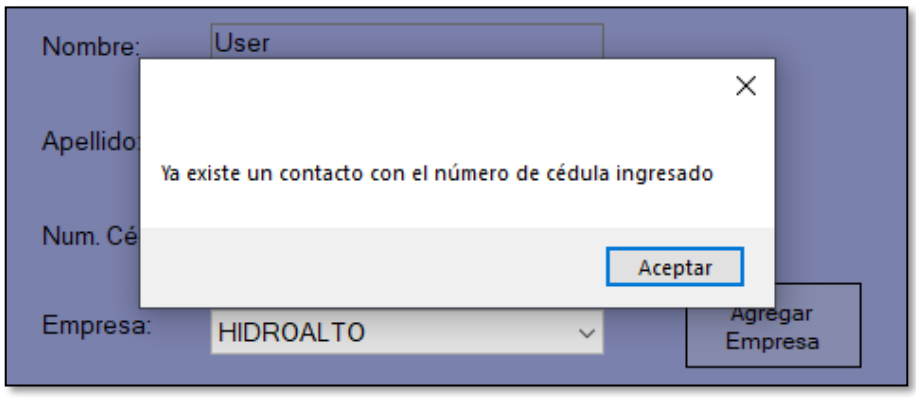

Figura 3.7 Mensaje de error al agregar un responsable

3.1.4 AGREGAR O MODIFICAR EMPRESA

Al momento de agregar o modificar un responsable se puede seleccionar o agregar una empresa, en caso de que esta no exista. En la Figura 3.8 se presenta el formulario para agregar una empresa.

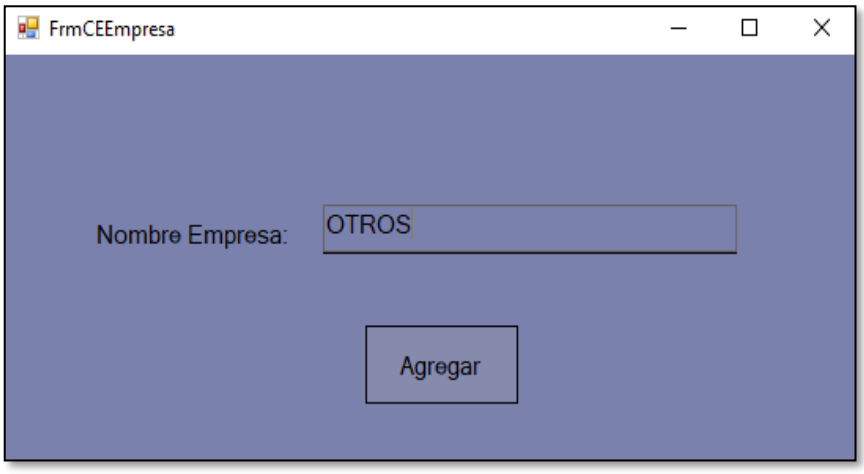

Figura 3.8 Interfaz gráfica para agregar empresa

En caso de que el usuario ingrese un nombre de empresa que ya se encuentre registrado, aparecerá un mensaje de error como el que se indica en la Figura 3.9.

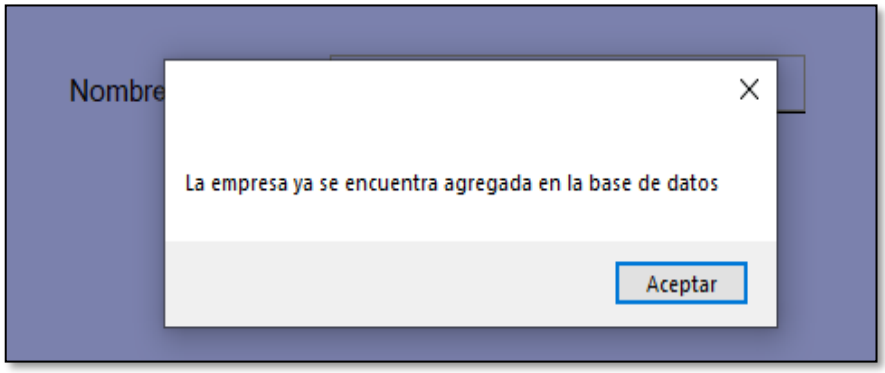

Figura 3.9 Mensaje de error al agregar una empresa

3.1.5 BUSCAR REUNIÓN, RESPONSABLE O EMPRESA

El usuario puede buscar una reunión, responsable o empresa. Para buscar una reunión, el usuario puede utilizar los siguientes filtros de búsqueda: código de reunión o intervalos de fechas; o se pueden presentar todas las reuniones. El formulario de búsqueda se indica en la Figura 3.10.

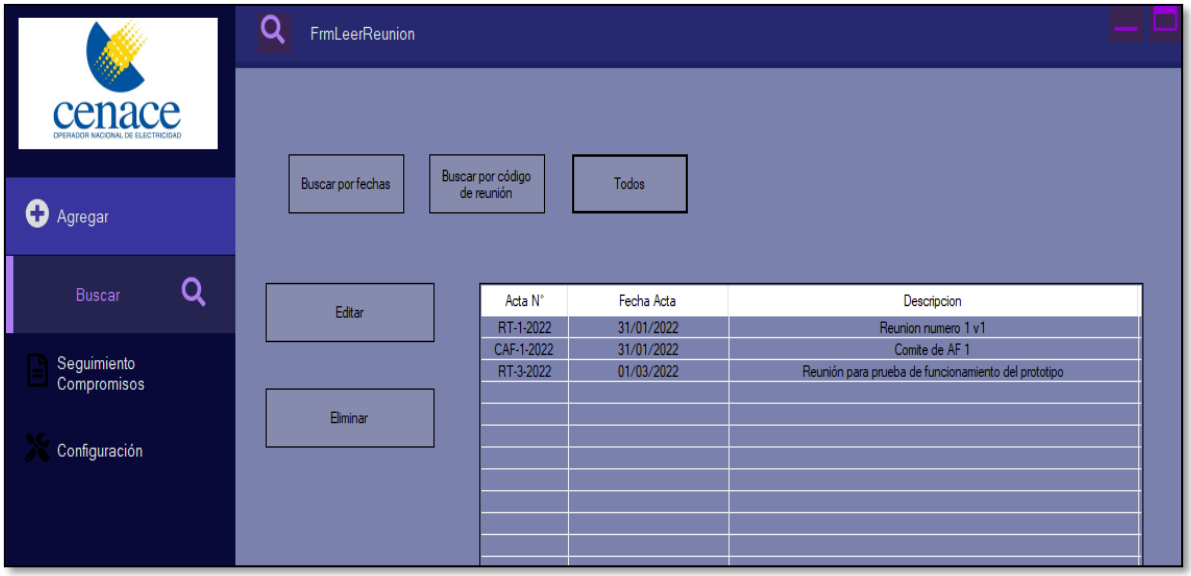

Figura 3.10 Interfaz gráfica para buscar una reunión

Para buscar un responsable, el usuario puede utilizar el número de cédula del responsable o puede listar todos los responsables, mediante el formulario de búsqueda de responsables que se visualiza en la Figura 3.11.

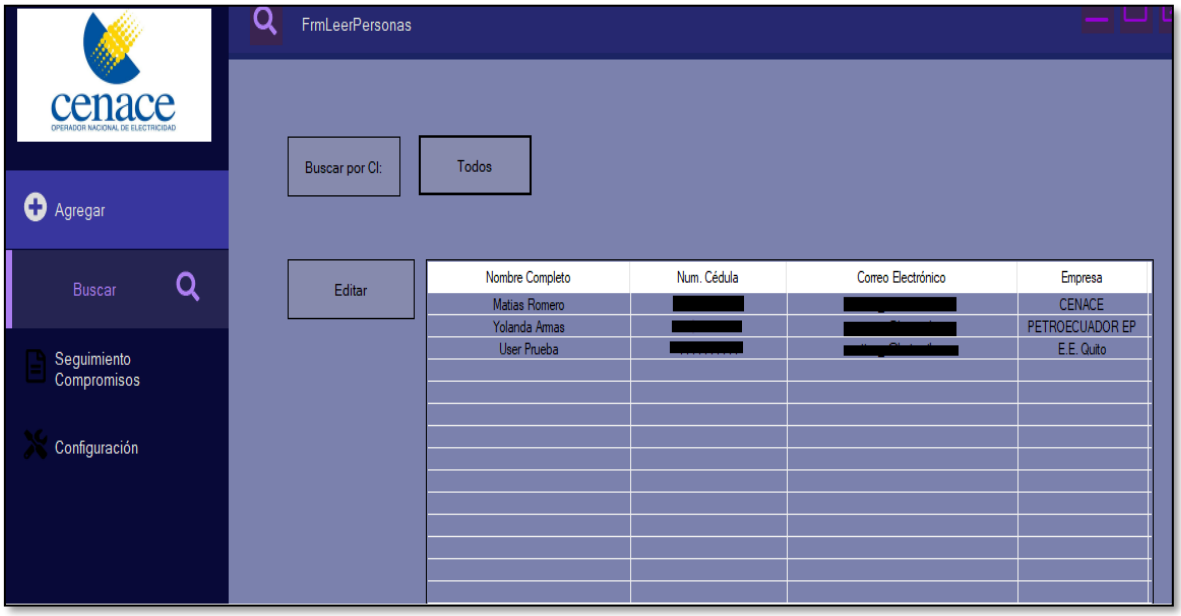

Figura 3.11 Interfaz gráfica para buscar un responsable

Para buscar una empresa, se emplea el formulario indicado en la Figura 3.12. El usuario puede buscar usando el nombre de la empresa o puede enlistar todas las empresas.

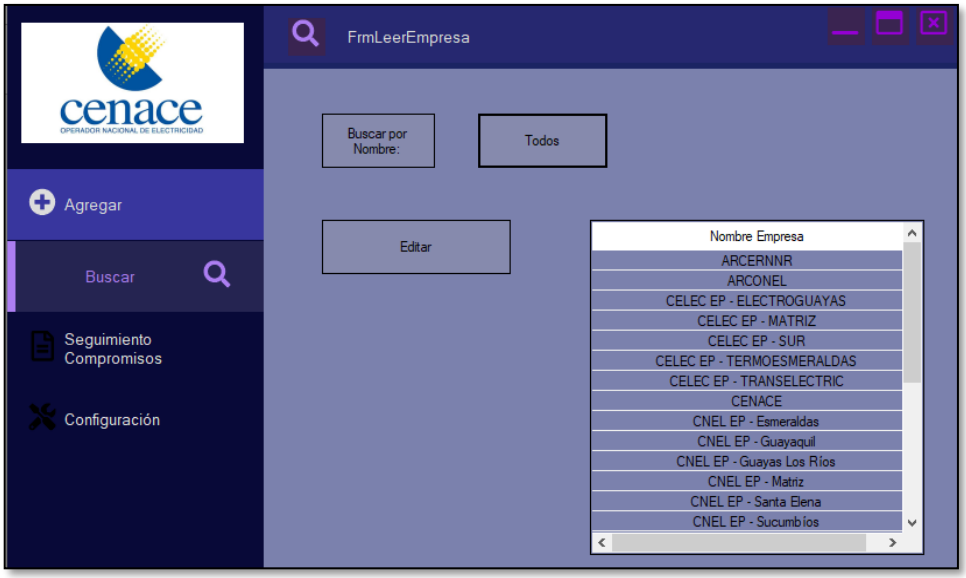

Figura 3.12 Interfaz gráfica para buscar una empresa

Los formularios para búsqueda disponen de botones que permite editar o eliminar cualquier registro seleccionado.

3.1.6 SEGUIMIENTO DE COMPROMISOS

Para dar seguimiento a los compromisos, el usuario dispone del formulario para buscar compromisos que se visualiza en la Figura 3.13. El usuario puede buscar los compromisos de una o varias reuniones, usando filtros de búsqueda como el código de reunión, intervalos de fechas o puede enlistar todas los compromisos de todas las reuniones.

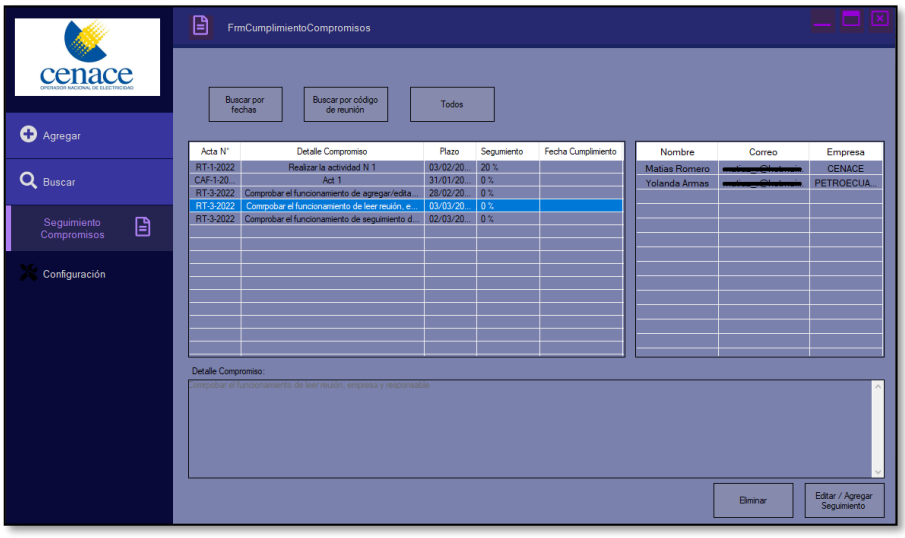

Figura 3.13 Interfaz gráfica para buscar compromisos

Una vez realizada la búsqueda, el usuario puede eliminar un compromiso o dar seguimiento al mismo. La Figura 3.14 muestra el formulario que permite dar seguimiento a un compromiso, mediante el cual o puede modificar la información del compromiso así como establecer el porcentaje de seguimiento del mismo.

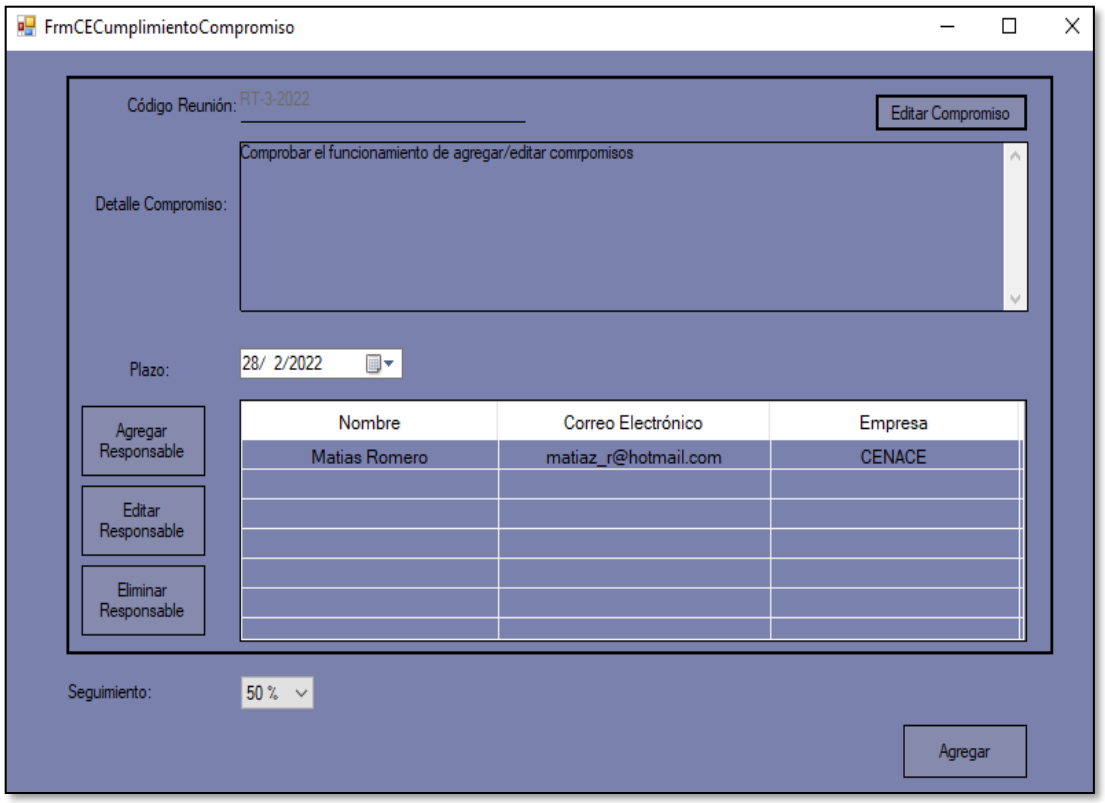

Figura 3.14 Interfaz gráfica para agregar el seguimiento de un compromiso

En caso de que el usuario coloca un valor del 100% en el seguimiento de compromiso, el formulario presenta controles para definir la fecha de cumplimiento así como para ingresar las posibles observaciones del mismo, como se indica en la Figura 3.15.

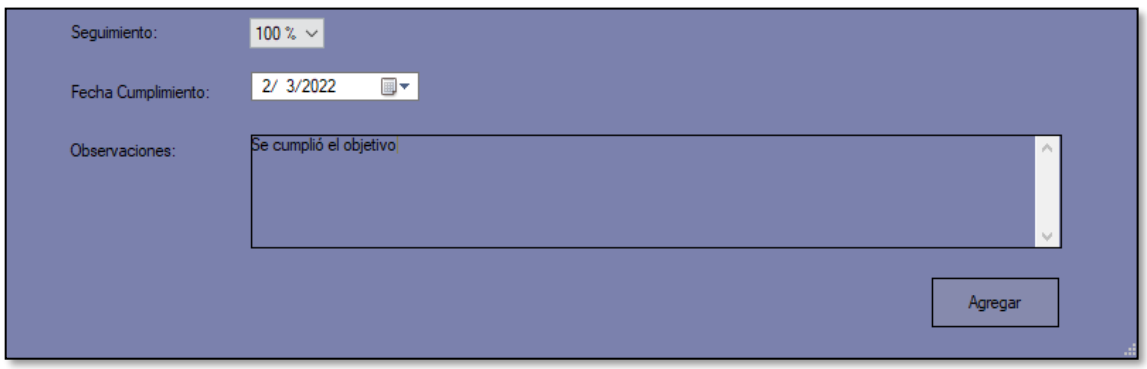

Figura 3.15 Interfaz gráfica para agregar cumplimiento de un compromiso

3.1.7 NOTIFICACIÓN DE PRÓXIMOS COMPROMISOS Y COMPROMISOS VENCIDOS

El prototipo tiene la funcionalidad de notificar al usuario acerca de compromisos cuya fecha de cumplimiento está próxima a vencer, así como también de compromisos cuya fecha de cumplimiento ya ha vencido. Mediante el formulario de alerta de compromisos, se presentará una lista de compromisos que cumplan con las condiciones indicadas en el párrafo previo, como se indica en la Figura 3.16.

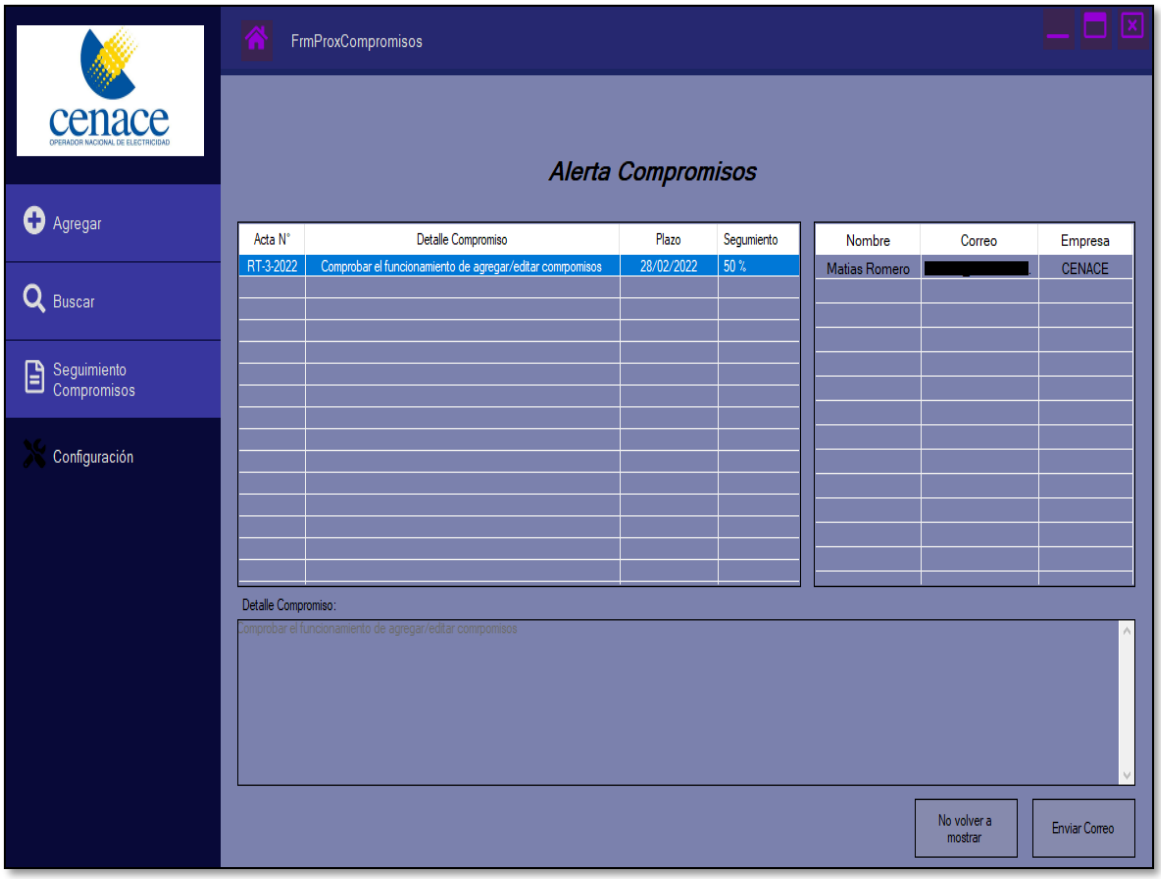

Figura 3.16 Interfaz gráfica para visualizar los compromisos vencidos

Mediante el formulario, el usuario puede informar a los responsables del compromiso mediante un correo electrónico que se genera desde el cliente, o indicar que no vuelva a presentarse la notificación del compromiso. Si el usuario decide enviar el correo electrónico a los responsables, dicho correo se enviará siempre y cuando, el cliente posea una conexión a Internet, Un ejemplo de correo electrónico enviado se visualiza en la Figura 3.17. Por otro lado, aquellos compromisos que estén a tres días de su fecha de cumplimiento y no se han cumplido, el prototipo notifica de manera automática mediante correo electrónico a los responsables de los mismos.

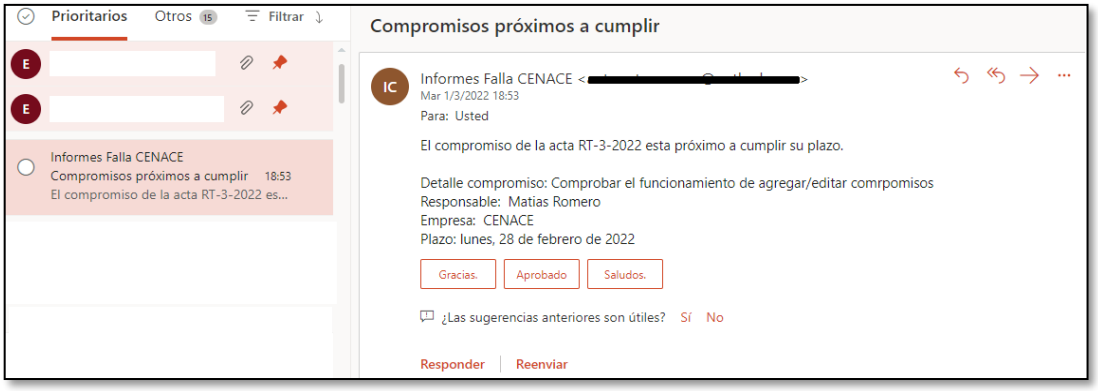

Figura 3.17 Notificación vía correo electrónico

3.2 PRUEBAS DE USUARIO

Una vez levantado el servicio WCF en IIS, se procedió a realizar las pruebas de usuario con los miembros del área de análisis de operaciones de CENACE.

Para que los usuarios sepan cómo emplear la aplicación se realizó un manual de usuario del prototipo, el cual se incluye en el ANEXO H. Una vez que los integrantes de CENACE usaron el prototipo, se solicitó que completen una encuesta para conocer si el prototipo desarrollado cumplió con los requisitos que deseaban. El modelo de la encuesta se encuentra en el ANEXO I.

En la Tabla 3.1 y Tabla 3.2, se resumen los resultados obtenidos de la encuesta aplicada a cinco miembros del área de análisis de operaciones de CENACE.

Tabla 3.1 Resultados de la encuesta de satisfacción (Parte 1)

No.	Pregunta	Respuesta	
		Sí	No
1	La aplicación, ¿le permitió agregar una nueva reunión al sistema?	100%	0%
2	La aplicación, ¿generó de manera automática el código de reunión al momento de seleccionar el tipo de reunión?	100%	0%
3	La aplicación, ¿le permitió agregar uno o varios compromisos a una reunión específica?	100%	0%
4	La aplicación, ¿le permitió agregar uno o varios responsables a un compromiso específico?	100%	0%

Tabla 3.2 Resultados de la encuesta de satisfacción (Parte 2)

No.	Pregunta	Respuesta	
		Sí	No
5	La aplicación, ¿le permitió agregar o seleccionar una empresa para asociarla con un responsable de un compromiso?	100%	0%
6	La aplicación, ¿le permitió buscar un responsable anteriormente registrado para asociarlo con un compromiso?	100%	0%
7	La aplicación, ¿le permitió editar una reunión y toda la información relacionada a ella (Lista de compromisos y lista de responsables de compromisos)?	100%	0%
8	La aplicación, ¿le permitió visualizar la lista de reuniones almacenadas en el sistema?	100%	0%
9	La aplicación, ¿le permitió visualizar la lista de empresas almacenadas en el sistema?	100%	0%
10	La aplicación, ¿le permitió visualizar la lista de responsables almacenados en el sistema?	100%	0%
11	La aplicación, ¿le permitió visualizar la lista de empresas almacenadas en el sistema?	100%	0%
12	La aplicación, ¿le permitió visualizar la lista de compromisos de una reunión almacenadas en el sistema?	100%	0%
13	La aplicación, ¿le permitió dar seguimiento a un compromiso?	100%	0%
14	La aplicación, ¿le permitió visualizar y modificar la información sobre la configuración general del sistema?	100%	0%
15	¿La aplicación envía una notificación por correo electrónico a los responsables de compromisos próximos a caducar?	100%	0%
16	¿La aplicación muestra una lista de compromisos que se encuentra próximos a caducar y caducados?	100%	0%
17	Del 1 al 5, siendo 1 poco amigable y 5 muy amigable ¿Qué tan intuitiva y amigable fue la interfaz de usuario?	4,6/5	
18	Del 1 al 5, siendo 1 muy mala y 5 muy buena ¿Qué tan conforme se encuentra con el prototipo presentado?	4,8/5	

Como se puede observar en la Tabla 3.1 y Tabla 3.2todos los requerimientos del usuario fueron cumplidos satisfactoriamente.

Durante las pruebas se reportaron algunos errores, los cuales se detallan en la Tabla 3.3, así también se describe la solución de cada uno de ellos.

Tabla 3.3 Errores encontrados en las pruebas de usuario

Error	Solución
Al momento de editar una reunión y modificar el tipo de reunión, y posteriormente regresar al tipo de reunión anterior, el código de reunión se modifica de igual manera y no vuelve al anterior.	Cuando el usuario selecciona la opción "Editar Reunión", en el manipulador del evento `Load` del formulario `FrmCEReunion` se crea una variable local que almacena el tipo de reunión posterior a la edición. Al finalizar la modificación de la reunión, se comprobará si se cambió el tipo de reunión con ayuda de la variable previamente creada. Si se modificó el tipo de reunión se modificará el código de reunión, si no se modificó el tipo de reunión se mantendrá el mismo código de reunión.
Al momento de editar una reunión, empresa, compromiso o responsable no se actualiza de información del mismo en las listas presentadas.	Cada vez que el usuario edite un registro (responsable, empresa, reunión o compromiso) y se vuelva a la pantalla en donde se enlista el registro se llama a un método que actualiza los registros con los cambios realizados.
Cuando el usuario desea buscar reuniones en base a un intervalo de fechas y coloca la fecha de inicio del intervalo mayor a la fecha de fin, genera un error en el objeto proxy al intentar acceder al servicio WCF y realizar la consulta en la base de datos.	En el formulario `FrmLeerReunion`, al momento de buscar reuniones en base a un intervalo de fechas se verificará que el intervalo de fechas sea el adecuado previo a la consulta a la base de datos a través del servicio WCF por medio del proxy. En caso de que el intervalo sea erróneo, se mostrará un mensaje de alerta al usuario indicando que el intervalo de fechas no es el adecuado.

Además, durante las pruebas se realizaron algunas sugerencias sobre el prototipo, los cuales se detallan en la Tabla 3.4.

Tabla 3.4 Sugerencias obtenidas en las pruebas de usuario

Sugerencia
Cambiar el aspecto del menú principal del formulario de inicio (`FrmPrincipal`) de botones a un menú desplegable.
Cambiar el texto de “Número de reunión” a “Código de reunión” en todos los formularios que trabajen con la información de las reuniones.
Al momento de enlistar los registros de los compromisos de una reunión, mostrar también una lista de los responsables de cada uno de los compromisos.
Cambiar el color de los mensajes de alerta a color rojo.
Utilizar la misma tipografía en todos los formularios del prototipo

4. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y recomendaciones producto del presente Trabajo de Integración Curricular.

4.1 CONCLUSIONES

- Al concluir este Trabajo de Titulación se dispone de un prototipo de sistema para la gestión y seguimiento de compromisos de CENACE. Este prototipo permite que los miembros del área de análisis de operaciones realicen la administración de los compromisos generados en las distintas reuniones del área, así como facilita el acceso a la información, y su búsqueda a través de filtros que permiten realizar consultas de los distintos informes almacenados en el prototipo.

- Para el Trabajo de Titulación se desarrolló en una arquitectura cliente-servidor, cuya funcionalidad fue estructurada en cuatro capas: capa de datos, capa de acceso a datos, capa de negocio y capa de presentación.

- Para el Trabajo de Titulación se utilizó SQL como lenguaje de desarrollo de la base de datos para realizar operaciones de acceso y manipulación de los datos, para la capa de acceso a datos se utilizó la tecnología DAO, en la capa lógica de negocio se realizó un servicio WCF con el uso del lenguaje orientado a objetos C# en conjunto con ASP.NET y la capa de presentación por medio de Windows Forms.

- Por medio de la realización de encuestas y reuniones con los miembros del personal del área de análisis de operaciones de CENACE se establecieron los requerimientos funcionales y no funcionales, lo que permitió construir un *backlog* que fue de utilidad para el desarrollo del prototipo.

- Como parte del diseño del prototipo se realizaron diagramas de la arquitectura del prototipo, diagrama de casos de uso, de actividades, de secuencia y clases que sirvieron de base para la implementación de cada una de las capas del prototipo.

- Con base en los requerimientos establecidos se desarrolló la aplicación cliente, la cual permite al usuario solicitar y obtener información desde el servicio WCF, el cual a su vez interactúa con la información presente en la base de datos. El usuario puede crear, modificar, eliminar y mostrar las reuniones almacenadas en el prototipo, además de gestionar toda la información relacionada con estas, como la

lista de compromisos y la lista de responsables de cada compromiso. También permite realizar consultas sobre los distintos compromisos considerando las reuniones en las que fueron acordados los mismos para que con ello se pueda modificar su información o agregar seguimiento de los compromisos. Por otra parte, el prototipo permite modificar su configuración, para gestionar las notificaciones y las claves de acceso de la cuenta de correo electrónico que permite enviar los correos con las notificaciones.

- La utilización de la metodología de desarrollo ágil Kanban permitió organizar las distintas tareas necesarias para el desarrollo de este Trabajo de Titulación.
- Se emplearon objetos proxys en la aplicación cliente que funcionan como una instancia local del servicio WCF hospedado en el servidor IIS y ofrece todas las operaciones necesarias para el correcto funcionamiento del prototipo.
- La validación del prototipo se realizó con miembros del área de análisis de operaciones de CENACE, quienes usaron el prototipo y pudieron identificar errores o plantearon sugerencias sobre el mismo. Esta validación permitió corroborar que los usuarios están satisfechos con el trabajo realizado y por tanto se puede afirmar que se cumplió los requerimientos iniciales propuestos.

4.2 RECOMENDACIONES

- Se recomienda, como futuras actualizaciones del prototipo, implementar un módulo tipo *dashboard* que permita analizar y monitorizar la información sobre los compromisos.
- Debido a que se desarrolló el prototipo con base en una arquitectura basada en capas, se recomienda incluir en la capa de presentación una aplicación móvil o aplicación web que consuma el servicio WCF de manera de expandir la funcionalidad del prototipo haciendo uso de otros tipos de clientes.
- Se recomienda la publicación del servicio WCF en la nube de Microsoft (Azure), ya que esta garantizan un mayor nivel de seguridad y alta disponibilidad a un menor costo que el tener un servidor propio.
- Se recomienda para futuras actualizaciones implementar roles de usuario para que cada uno de estos tengan actividades específicas en el prototipo.

- Se recomienda la expansión del campo de utilización del prototipo a otras áreas de CENACE que requieran una gestión de la información de los compromisos de toda la empresa.
- Se recomienda implementar un módulo tipo *login* para el acceso y manipulación de la información almacenada en la base de datos.
- Para futuras actualizaciones, se recomienda nuevos filtros de búsqueda de los registros de las reuniones, compromisos, responsables y empresas con la finalidad de tener un manejo más eficiente de la información almacenada, como por ejemplo buscar las reuniones dadas en un año en específico.
- Se recomienda la utilización de la metodología de desarrollo ágil Kanban para cualquier tipo de trabajo, ya que permite culminar un trabajo de manera ordenada estableciendo tareas y el flujo del mismo.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. T. Chambers, «Apertura de la conferencia CISCO,» de *Servicio al cliente de CISCO*, 2019.

[2] Operador Nacional de Electricidad - CENACE, «Gob.ec,» Gobierno Electrónico, 01 12 2021. [En línea]. Available: https://www.gob.ec/cenace. [Último acceso: 03 01 2022].

[3] Microsoft Documents, «Arquitecturas de aplicaciones web comunes,» 2021. [En línea]. Available: https://docs.microsoft.com/es-es/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures. [Último acceso: 21 10 2021].

[4] M. Muñoz, «Introducción al desarrollo de aplicaciones N-Capas con tecnologías Microsoft – Manual de estudiante,» de *Introducción al desarrollo de aplicaciones N-Capas con tecnologías Microsoft – Manual de estudiante*, México, TI-capacitación, 2018, pp. 6-8, 17-20.

[5] IBM Cloud Education, «What is three-tier architecture?,» 2020. [En línea]. Available: https://www.ibm.com/cloud/learn/three-tier-architecture. [Último acceso: 21 10 2021].

[6] J. Esterkin, «Layered Architecture,» OpenClassrooms, 29 06 2020. [En línea]. Available: https://openclassrooms.com/en/courses/6397806-design-your-software-architecture-using-industry-standard-patterns/6896176-layered-architecture. [Último acceso: 03 01 2022].

[7] Microsoft Documents, «Databases,» docs.microsoft, 25 05 2021. [En línea]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/databases/databases?view=sql-server-ver15. [Último acceso: 21 10 2021].

[8] C. Coronel y S. Morris, Database Systems: Design, Implementation, & Management, Cengage Learning, 2016.

[9] A. Silberschatz, H. Korth y S. Sudarshan, Fundamentos de bases de datos, McGraw-Hill Interamericana de España S.L., 2014.

[10] Microsoft Documents, «What is SQL Server Management Studio (SSMS)?,» docs.microsoft, 27 12 2021. [En línea]. Available: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15. [Último acceso: 03 01 2022].

[11] A. Garrido Tejero, «El patrón DAO (Data Access Object). Manteniendo la persistencia de datos,» Universitat Politècnica de València UPV, 04 10 2017. [En línea]. Available: https://www.youtube.com/watch?v=CEDKxPCgosY&list=LL. [Último acceso: 03 01 2022].

[12] O. Blancarte, «Data Access Object (DAO) Pattern,» oscarblancarteblog.com, 10 12 2018. [En línea]. Available: https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/. [Último acceso: 03 01 2022].

[13] Microsoft Documents, «What Is Windows Communication Foundation,» docs.microdosft, 16 12 2021. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf. [Último acceso: 03 01 2022].

[14] Microsoft Documents, «Fundamental Windows Communication Foundation Concepts,» docs.microsoft, 15 09 2021. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/fundamental-concepts. [Último acceso: 21 10 2021].

[15] Microsoft Documents, «Windows Communication Foundation Architecture,» docs.microsoft, 15 09 2021. [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/architecture. [Último acceso: 21 10 2021].

[16] Á. De León, «Servidor IIS,» Tutoriales de Hosting, 11 11 2019. [En línea]. Available: https://blog.infranetworking.com/servidor-iis/. [Último acceso: 21 10 2021].

[17] L. Castellano Lendínez, «Kanban. Metodología para aumentar la efciencia de los procesos,» 2019. [En línea]. Available: https://www.3ciencias.com/wp-content/uploads/2019/03/ART.-2-TECNO-Ed.-29_Vol.-8_n%C2%BA-1-1.pdf. [Último acceso: 21 10 2021].

[18] H. D. Andrade Unusungo, «Desarrollo de un sistema prototipo para gestión y seguimiento de los trabajos de titulación en la Carrera de Ingeniería Electrónica y Redes de Información,» bibdigital.epn.edu.ec, 15 03 2018. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/19295. [Último acceso: 21 10 2021].

[19] R. X. García Pazmiño, «Desarrollo e implantación del sistema de seguimiento de proyectos de investigación y vinculación para la escuela politécnica nacional,» bibdigital.epn.edu.ec, 09 05 2014. [En línea]. Available: http://bibdigital.epn.edu.ec/handle/15000/7376. [Último acceso: 21 10 2021].

[20] Operador Nacional de Electricidad - CENACE, «cenace.gob,» 01 12 2021. [En línea]. Available: http://www.cenace.gob.ec/quienes-somos/. [Último acceso: 06 01 2022].

[21] M. Rehkophf, «Atlassian Agile Coach,» Atlassian, [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories. [Último acceso: 09 02 2022].

[22] R. C. Martin, UML PARA PROGRAMADORES JAVA, PEARSON EDUCACION, 2004.

[23] Diagramas UML, «Diagramas UML - Diagrama de clases,» Magazine Pro on Genesis Framework, [En línea]. Available: https://diagramasuml.com/diagrama-de-clases/. [Último acceso: 16 02 2022].

[24] Microsoft, «Visual Studio - Descargas,» Microsoft, 2022. [En línea]. Available: https://visualstudio.microsoft.com/es/downloads/. [Último acceso: 22 02 2022].

[25] Microsoft, «Download SQL Server Management Studio (SSMS),» Microsoft Docs, 12 08 2021. [En línea]. Available: https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15. [Último acceso: 22 02 2022].

[26] Microsoft, «Distribución de una aplicación de Windows 10 desde un servidor IIS,» Microsoft Docs, 07 02 2022. [En línea]. Available: https://docs.microsoft.com/es-es/windows/msix/app-installer/web-install-iis. [Último acceso: 22 02 2022].

[27] Oracle Corporation, «Descripción de capas lógicas (Descripción general técnica de Sun Java Enterprise System 5),» Docs.oracle.com, 2010. [En línea]. Available: https://docs.oracle.com/cd/E19528-01/820-0888/aaubb/index.html. [Último acceso: 21 10 2021].

[28] S. D. Moquillaza Henríquez, H. Vega Huerta y L. A. Guerra Grados, «Programación en N capas,» *Revista de Información de Sistemas e Informática,* vol. VII, nº 2, pp. 57-67, 2010.

[29] A. Vázquez Cendron, Arquitectura en capas: análisis y estudio de caso del modelo arquitectónico N-capas y sus variantes, La Plata: Tesis de Grado - Universidad Nacional de La Plata, 2018.

[30] Microsoft, «Walkthrough: Create a simple WCF service in Windows Forms,» Microsoft Docs, 08 05 2021. [En línea]. Available: https://docs.microsoft.com/es-es/visualstudio/data-tools/walkthrough-creating-a-simple-wcf-service-in-windows-forms?view=vs-2022. [Último acceso: 02 03 2022].

[31] Microsoft, «Create a Windows Forms app in Visual Studio with C#,» Microsoft Docs, 21 01 2022. [En línea]. Available: https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022. [Último acceso: 02 03 2022].

ANEXOS

ANEXO A. Modelo de la entrevista para la toma de requerimientos.

ANEXO B. Historias de Usuario.

ANEXO C. Detalle de las tablas del diagrama relacional.

ANEXO D. Diagramas de actividades.

ANEXO E. Diagramas de secuencia.

ANEXO F. Sketches del prototipo.

ANEXO G. Código del prototipo.

ANEXO H. Manual de usuario.

ANEXO I. Modelo de la entrevista de satisfacción de los requerimientos.

ANEXO A

En este anexo se presenta el modelo de encuesta para la obtención de requerimientos del prototipo.

1. **Actualmente, ¿Posee un software que permita la gestión de la información de los compromisos de CENACE?**
   - o Sí
   - o No
2. **¿Ha tenido problemas al momento de realizar búsquedas acerca de la información de los compromisos de CENACE?**
   - o Sí
   - o No
3. **Indique cuáles problemas ha tenido al momento de realizar una búsqueda de la información de compromisos. Seleccione uno o varios**
   - □ Datos repetidos (Datos duplicados)
   - □ Datos eliminados (No se encuentra información de un compromiso)
   - □ Datos incompletos (No tiene coherencia)
   - □ Otros: _____
4. **¿Está de acuerdo que se desarrolle un software que permita gestionar toda la información acerca de los compromisos de CENACE?**
   - o Sí
   - o No
5. **¿Qué tipo de software desearía para la gestión de información?**
   - o Aplicación de Escritorio
   - o Aplicación web
   - o Aplicación en dispositivos móviles
6. **¿Qué criterios considera necesario para el filtro de búsqueda de los diferentes compromisos? Seleccione uno o varios.**
   - □ Según un intervalo de tiempo (Fecha inicio - fecha final)
   - □ Según el identificador de la reunión
   - □ Según el nombre de empresas responsables

□ Según la fecha plazo de cumplimiento de compromisos

□ Otro: _____

7. **El sistema permitirá realizar búsquedas de los compromisos pasados, ¿Considera oportuno modificar, eliminar estos eventos pasados?**

   o Sí

   o No

8. **¿Desea que el sistema notifique a los responsables de un compromiso próximo a cumplirse su fecha límite o compromisos vencidos mediante un correo electrónico?**

   o Sí

   o No

9. **¿Desea un módulo que permita la configuración del envió de los correos electrónicos?**

   o Sí

   o No

10. **¿Desea un módulo de alerta para visualizar los compromisos que están próximos a caducar y caducados?**

    o Sí

    o No

11. **¿Considera oportuno el inicio de sesión de la aplicación para acceder a todos los servicios que dispone?**

    o Sí

    o No

ANEXO B

HISTORIAS DE USUARIO

Tabla B.0.1 Historia de Usuario “Agregar reuniones al sistema”

HISTORIA DE USUARIO					
ID:	HU01	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Agregar reuniones al sistema					
Descripción: El usuario administrador desea agregar reuniones al sistema y toda la información relacionada con la reunión como lo son: tipo de reunión, fecha de reunión, tema de convocatoria y lista de compromisos.					

Tabla B.0.2 Historia de Usuario “Automatizar el identificador de la reunión”

HISTORIA DE USUARIO					
ID:	HU02	**Rol:**	Administrador	**Prioridad:**	2
Nombre de Historia: Automatizar el identificador de la reunión.					
Descripción: El usuario administrador desea automatizar el identificador de la reunión según el tipo de reunión (Comité de Análisis de Falla o Reunión Técnica), el número de reunión y el año de la reunión.					

Tabla B.0.3 Historia de Usuario “Agregar uno o varios compromisos a las reuniones”

HISTORIA DE USUARIO					
ID:	HU03	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Agregar uno o varios compromisos a las reuniones					
Descripción: El usuario administrador desea agregar uno o varios compromisos a las reuniones y toda la información relacionada al mismo como lo son: detalle de compromiso, fecha plazo del compromiso y lista de responsables.					

Tabla B.0.4 Historia de Usuario “Agregar uno o varios responsables de un compromiso”

HISTORIA DE USUARIO					
ID:	HU04	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Agregar uno o varios responsables de un compromiso					
Descripción: El usuario administrador desea agregar uno o varios responsables a un compromiso y toda la información relacionada con este, como lo son: nombre, apellido número de cédula, empresa y correo electrónico. Se debe verificar que el usuario solo pueda ingresar números en el campo de número de cédula y que no existan contactos con el mismo número de cédula.					

Tabla B.0.5 Historia de Usuario “Agregar empresas”

HISTORIA DE USUARIO					
ID:	HU05	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Agregar empresas					
Descripción: El usuario administrador desea agregar empresas para luego asociarlas a los responsables. Se debe verificar que no existan dos empresas con el mismo nombre.					

Tabla B.0.6 Historia de Usuario “Buscar un contacto al momento de seleccionarlo como responsable”

HISTORIA DE USUARIO					
ID:	HU06	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Buscar un contacto al momento de seleccionarlo como responsable					
Descripción: El usuario administrador desea un módulo que permita al usuario buscar un contacto por el número de cédula al momento de seleccionar a los responsables de un compromiso de una reunión.					

Tabla B.0.7 Historia de Usuario “Editar una reunión”

HISTORIA DE USUARIO					
ID:	HU07	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Editar una reunión.					
Descripción: El usuario administrador desea editar las reuniones que se encuentren en el sistema y toda la información relacionada con ella como lo son: tipo de reunión, fecha de reunión, tema de convocatoria y lista de compromisos.					

Tabla B.0.8 Historia de Usuario “Administrar lista de compromisos”

HISTORIA DE USUARIO					
ID:	HU08	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Administrar lista de compromisos.					
Descripción: El usuario administrador desea administrar los compromisos asociados a una reunión y toda la información asociada a ella como: detalle de compromiso, fecha plazo del compromiso y lista de responsables. Desea poder agregar nuevos compromisos a la reunión, eliminar compromisos de una reunión y editar la información de los compromisos.					

Tabla B.0.9 Historia de Usuario “Administrar lista de responsables”

HISTORIA DE USUARIO					
ID:	HU09	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Administrar lista de responsables					
Descripción: El usuario administrador desea administrar los responsables relacionados con un compromiso y toda la información asociada a ella como: nombre, apellido número de cédula, empresa y correo electrónico. Desea poder agregar nuevos responsables al compromiso, eliminarlos de la lista de responsables y editar cada uno de ellos. Se debe verificar que al momento de editar un usuario, el nuevo número de cédula no esté siendo ocupado por otro contacto.					

Tabla B.0.10 Historia de Usuario “Visualizar las reuniones por intervalo de fechas”

HISTORIA DE USUARIO					
ID:	HU10	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Visualizar las reuniones por intervalo de fechas					
Descripción: El usuario administrador desea visualizar las reuniones almacenadas en el sistema por medio de un filtro de búsqueda de un rango de fechas, dando como resultado las reuniones realizadas entre una fecha de inicio y una fecha de fin. Además, desea que al momento de visualizar las reuniones pueda realizar acciones con las reuniones como editar y eliminar las mismas.					

Tabla B.0.11 Historia de Usuario “Visualizar las reuniones por código de reunión”

HISTORIA DE USUARIO					
ID:	HU11	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Visualizar las reuniones por código de reunión					
Descripción: El usuario administrador desea visualizar las reuniones almacenadas en el sistema por medio de un filtro de búsqueda del código de reunión, dando como resultado la reunión que contenga ese código. Además, desea que al momento de visualizar las reuniones pueda realizar acciones con las reuniones como editar y eliminar las mismas.					

Tabla B.0.12 Historia de Usuario “Visualizar todas las reuniones”

HISTORIA DE USUARIO					
ID:	HU12	**Rol:**	Administrador	**Prioridad:**	2
Nombre de Historia: Visualizar todas las reuniones					
Descripción: El usuario administrador desea visualizar todas las reuniones almacenadas en el sistema. Además, desea que al momento de visualizar las reuniones pueda realizar acciones con las reuniones como editar y eliminar las mismas.					

Tabla B.0.13 Historia de Usuario “Eliminar una reunión”

HISTORIA DE USUARIO					
ID:	HU13	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Eliminar una reunión					
Descripción: El usuario administrador desea eliminar una reunión del sistema y toda su información asociada al mismo como lo es la lista de compromisos y de responsables respectivamente.					

Tabla B.0.14 Historia de Usuario “Visualizar los responsables por número de cédula”

HISTORIA DE USUARIO					
ID:	HU14	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Visualizar los responsables por número de cédula.					
Descripción: El usuario administrador desea visualizar los responsables almacenados en el sistema por medio de un filtro de búsqueda del número de cédula, dando como resultado el contacto que contenga ese número de cédula. Además, desea que al momento de visualizar al responsable pueda realizar acciones como editar.					

Tabla B.0.15 Historia de Usuario “Visualizar todos los responsables”

HISTORIA DE USUARIO					
ID:	HU15	**Rol:**	Administrador	**Prioridad:**	2
Nombre de Historia: Visualizar todos los responsables.					
Descripción: El usuario administrador desea visualizar todos los responsables almacenados en el sistema. Además, desea que al momento de visualizar al responsable pueda realizar acciones como editar.					

Tabla B.0.16 Historia de Usuario “Visualizar las empresas por nombre”

HISTORIA DE USUARIO					
ID:	HU16	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Visualizar las empresas por nombre.					
Descripción: El usuario administrador desea visualizar las empresas almacenadas en el sistema por medio de un filtro de búsqueda del nombre, dando como resultado la empresa que contenga ese nombre. Además, desea que al momento de visualizar la empresa pueda realizar acciones como editar.					

Tabla B.0.17 Historia de Usuario “Visualizar todas las empresas”

HISTORIA DE USUARIO					
ID:	HU17	**Rol:**	Administrador	**Prioridad:**	1
Nombre de Historia: Visualizar todas las empresas.					
Descripción: El usuario administrador desea visualizar todas las empresas almacenadas en el sistema. Además, desea que al momento de visualizar la empresa pueda realizar acciones como editar.					

Tabla B.0.18 Historia de Usuario “Editar una empresa”

HISTORIA DE USUARIO					
ID:	HU18	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Editar una empresa.					
Descripción: El usuario administrador desea editar una empresa y su información asociada a la misma como lo es el nombre de empresa. Al momento de editar, se debe verificar que el nuevo nombre de la empresa no sea uno ya en uso.					

Tabla B.0.19 Historia de Usuario “Visualizar los compromisos por intervalos de fechas”

HISTORIA DE USUARIO					
ID:	HU19	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Visualizar los compromisos por intervalos de fechas.					
Descripción: El usuario administrador desea visualizar los compromisos almacenados en el sistema por medio de un filtro de búsqueda de un rango de fechas, dando como resultado los compromisos de las reuniones realizadas entre una fecha de inicio y una fecha de fin. Además, desea que al momento de visualizar los compromisos pueda realizar acciones con los mismos como editar/dar seguimiento y eliminar las mismas.					

Tabla B.0.20 Historia de Usuario “Visualizar los compromisos por código de reunión”

HISTORIA DE USUARIO					
ID:	HU20	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Visualizar los compromisos por código de reunión.					
Descripción: El usuario administrador desea visualizar los compromisos almacenados en el sistema por medio de un filtro de búsqueda del código de reunión, dando como resultado los compromisos de la reunión que contenga ese código. Además, desea que al momento de visualizar los compromisos pueda realizar acciones como editar/dar seguimiento y eliminar las mismas.					

Tabla B.0.21 Historia de Usuario “Visualizar todos los compromisos”

HISTORIA DE USUARIO					
ID:	HU21	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Visualizar todos los compromisos.					
Descripción: El usuario administrador desea visualizar todos los compromisos almacenados en el sistema. Además, desea que al momento de visualizar los compromisos pueda realizar acciones como editar/dar seguimiento y eliminar las mismas.					

Tabla B.0.22 Historia de Usuario “Eliminar un compromiso”

HISTORIA DE USUARIO					
ID:	HU22	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Eliminar un compromiso.					
Descripción: El usuario administrador desea eliminar un compromiso del sistema y toda su información asociada al mismo como lo es la lista de responsables.					

Tabla B.0.23 Historia de Usuario “Editar un compromiso”

HISTORIA DE USUARIO					
ID:	HU23	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Editar un compromiso.					
Descripción: El usuario administrador desea editar un compromiso, es decir toda la información relacionada con el compromiso como lo son: detalle de compromiso, fecha plazo del compromiso y lista de responsables.					

Tabla B.0.24 Historia de Usuario “Seguimiento un compromiso”

HISTORIA DE USUARIO					
ID:	HU24	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Seguimiento un compromiso.					
Descripción: El usuario administrador desea realizar un seguimiento a un compromiso, para ello selecciona un porcentaje de seguimiento del compromiso de 0% a 100% con intervalos de 10%. Además, al seleccionar un seguimiento del 100% desea agregar un cumplimento al mismo.					

Tabla B.0.25 Historia de Usuario “Cumplimiento de un compromiso”

HISTORIA DE USUARIO					
ID:	HU25	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Cumplimiento de un compromiso.					
Descripción: El usuario administrador desea que al momento de seleccionar un seguimiento de 100% de un compromiso, colocar una fecha de cumplimiento del compromiso y además observaciones sobre el mismo.					

Tabla B.0.26 Historia de Usuario “Administración de la configuración general”

HISTORIA DE USUARIO					
ID:	HU26	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Administración de la configuración general					
Descripción: El usuario administrador desea visualizar y modificar los datos de la configuración general de la aplicación, como lo son: el correo de salida de las notificaciones, la contraseña del correo electrónico de salida, el número de días antes de la caducidad de un compromiso para la notificación a los responsables y el número de días posteriores a la caducidad para el envío de una alerta por correo electrónico.					

Tabla B.0.27 Historia de Usuario “Notificaciones por correo electrónico”

HISTORIA DE USUARIO					
ID:	HU27	**Rol:**	Administrador	**Prioridad:**	5
Nombre de Historia: Notificaciones por correo electrónico					
Descripción: El usuario administrador desea que se envíen dos tipos de notificaciones a los responsables de los compromisos que no tengan un seguimiento del 100% en sus tareas. Una notificación antes de la fecha plazo del compromiso de manera de advertencia y una notificación después de la fecha plaza de manera de alerta. Los tiempos del envío de correos se darán en relación a la configuración general de la aplicación.					

Tabla B.0.28 Historia de Usuario “Visualizar los compromisos próximos a expirar”

HISTORIA DE USUARIO					
ID:	HU28	**Rol:**	Administrador	**Prioridad:**	3
Nombre de Historia: Visualizar los compromisos próximos a expirar					
Descripción: El usuario administrador desea visualizar una lista de compromisos que se encuentren próximos a llegar su fecha plazo para tener un control personal sobre ellos. Esta lista de compromisos solamente aparecerá si existen compromisos que se encuentren dentro de los días de la configuración general y que sean compromisos que no tengan un seguimiento del 100%.					

Tabla B.0.29 Historia de Usuario “Visualizar los compromisos caducados”

HISTORIA DE USUARIO					
ID:	HU29	**Rol:**	Administrador	**Prioridad:**	4
Nombre de Historia: Visualizar los compromisos caducados					
Descripción: El usuario administrador desea visualizar una lista de compromisos que se encuentren caducados para tener un control personal sobre ellos. Esta lista de compromisos solamente aparecerá si existen compromisos que se encuentren dentro de los días de la configuración general y que sean compromisos que no tengan un seguimiento del 100%. Además, desea que al momento de visualizar los compromisos pueda realizar acciones con los mismos como no volver a mostrar en la lista y enviar un correo de alerta a los responsables.					

ANEXO C

DETALLE DE LAS TABLAS DEL DIAGRAMA RELACIONAL.

Tabla C.0.1 Detalle de la tabla `tblEmpresa`

tblEmpresa		
Atributo	**Tipo de dato**	**Descripción**
`IdEmpresa`	`Int`	Identificador único y auto numerado que permite distinguir a cada empresa.
`nombreEmpresa`	`varchar(50)`	Nombre de la empresa.

Tabla C.0.2 Detalle de la tabla `tblResponsable`

tblResponsable		
Atributo	**Tipo de dato**	**Descripción**
`idResponsable`	`Int`	Identificador único y auto numerado que permite distinguir a cada responsable.
`nombreResponsable`	`varchar(50)`	Nombre del responsable.
`apellidoResponsable`	`varchar(50)`	Apellido del responsable.
`numCedula`	`varchar(10)`	Número de cédula del responsable.
`correoReponsable`	`varchar(50)`	Correo electrónico del responsable.
`idEmpresa`	`Int`	Identificador único de la tabla `tblEmpresa` que sirve para relacionar un responsable con una empresa.

Tabla C.0.3 Detalle de la tabla `tblResponsableCompromiso`

tblResponsableCompromiso		
Atributo	**Tipo de dato**	**Descripción**
idResponsableCompromiso	Int	Identificador único y auto numerado que permite distinguir a cada elemento de la tabla.
idResponsable	Int	Identificador único de la tabla `tblResponsable` que sirve para relacionar un responsable con uno o varios compromisos.
idCompromiso	Int	Identificador único de la tabla `tblCompromiso` que sirve para relacionar un compromiso con uno o varios responsables.

Tabla C.0.4 Detalle de la tabla `tblCompromiso`

tblCompromiso		
Atributo	**Tipo de dato**	**Descripción**
idCompromiso	Int	Identificador único y auto numerado que permite distinguir a cada compromiso.
DetalleCompromiso	varchar(1500)	Descripción breve sobre la actividad del compromiso.
PlazoCompromiso	Date	Es la fecha límite que se tiene para la realización por completo de la actividad del compromiso.
notificación	Int	Esta variable nos ayuda a identificar si el compromiso ha sido notificado a los responsables o no por medio de un número.
seguimiento	Int	Este campo permite conocer el porcentaje de seguimiento del compromiso.
alertaProximo	Int	Este campo permite distinguir si un compromiso ha sido notificado al usuario administrador que está próximo a llegar a su fecha plazo.
idReunion	Int	Identificador único de la tabla `tblReunion` que sirve para relacionar una reunión con uno o varios compromisos.

Tabla C.0.5 Detalle de la tabla `tblCumplimientoCompromiso`

tblCumplimientoCompromiso		
Atributo	**Tipo de dato**	**Descripción**
idCumplimiento Compromiso	Int	Identificador único y auto numerado que permite diferenciar a un elemento de los demás.
Oportunidad	Int	Es un campo que se calcula de forma automática en función de la fecha de cumplimento y la fecha plazo del compromiso.
FechaCumplimiento	Date	Es la fecha de cumplimiento de un compromiso.
Observaciones	Varchar (1000)	Es una breve descripción de las posibles observaciones que se presenten al momento de culminar el compromiso.
IdCompromiso	Int	Identificador único de la tabla `tblCompromiso` que sirve para relacionar un compromiso con su cumplimiento.

Tabla C.0.6 Detalle de la tabla `tblReunion`

tblReunion		
Atributo	**Tipo de dato**	**Descripción**
IdReunion	Int	Identificador único y auto numerado que permite diferenciar a una reunión.
NumeroReunion	Int	Es un campo que permite conocer el número de reuniones existentes según su tipo.
FechaReunion	Date	Es la fecha en que se realizó la reunión.
Borrado	Int	Este campo permite saber si el elemento ha sido borrado o no, sirve como respaldo al momento de querer recuperar una reunión que ha sido eliminada.
IdTipo	Int	Identificador único de la tabla `tblTipo` que sirve para relacionar un tipo de reunión con una reunión.

Tabla C.0.7 Detalle de la tabla `tblTipo`

tblTipo		
Atributo	**Tipo de dato**	**Descripción**
idTipo	Int	Identificador único y auto numerado que permite diferenciar a un tipo de reunión.
nombreTipo	varchar(50)	Es el nombre del tipo de reunión.

Tabla C.0.8 Detalle de la tabla `tblConfiguracionGeneral`

tblConfiguracionGeneral		
Atributo	**Tipo de dato**	**Descripción**
idConfigGeneral	Int	Identificador único y auto numerado que permite diferenciar a un elemento de los demás.
numAntes	Int	Es un campo que determina el número de días antes de la fecha plazo de un compromiso para el envío de una notificación a los responsables.
numDespues	Int	Es un campo que indica el número de días después de la fecha plazo de un compromiso para el envío de una notificación a los responsables.

Tabla C.0.9 Detalle de la tabla `tblCorreoSalida`

tblCorreoSalida		
Atributo	**Tipo de dato**	**Descripción**
idCorreoSalida	Int	Permite identificar de manera única a un elemento de los demás almacenados.
correo	varchar(100)	Este campo define el correo electrónico de salida.
Pass	varchar(100)	Este campo indica la contraseña del correo de salida para el envío de notificaciones a los responsables de los compromisos.

ANEXO D

DIAGRAMA DE ACTIVIDADES

Agregar Reunión

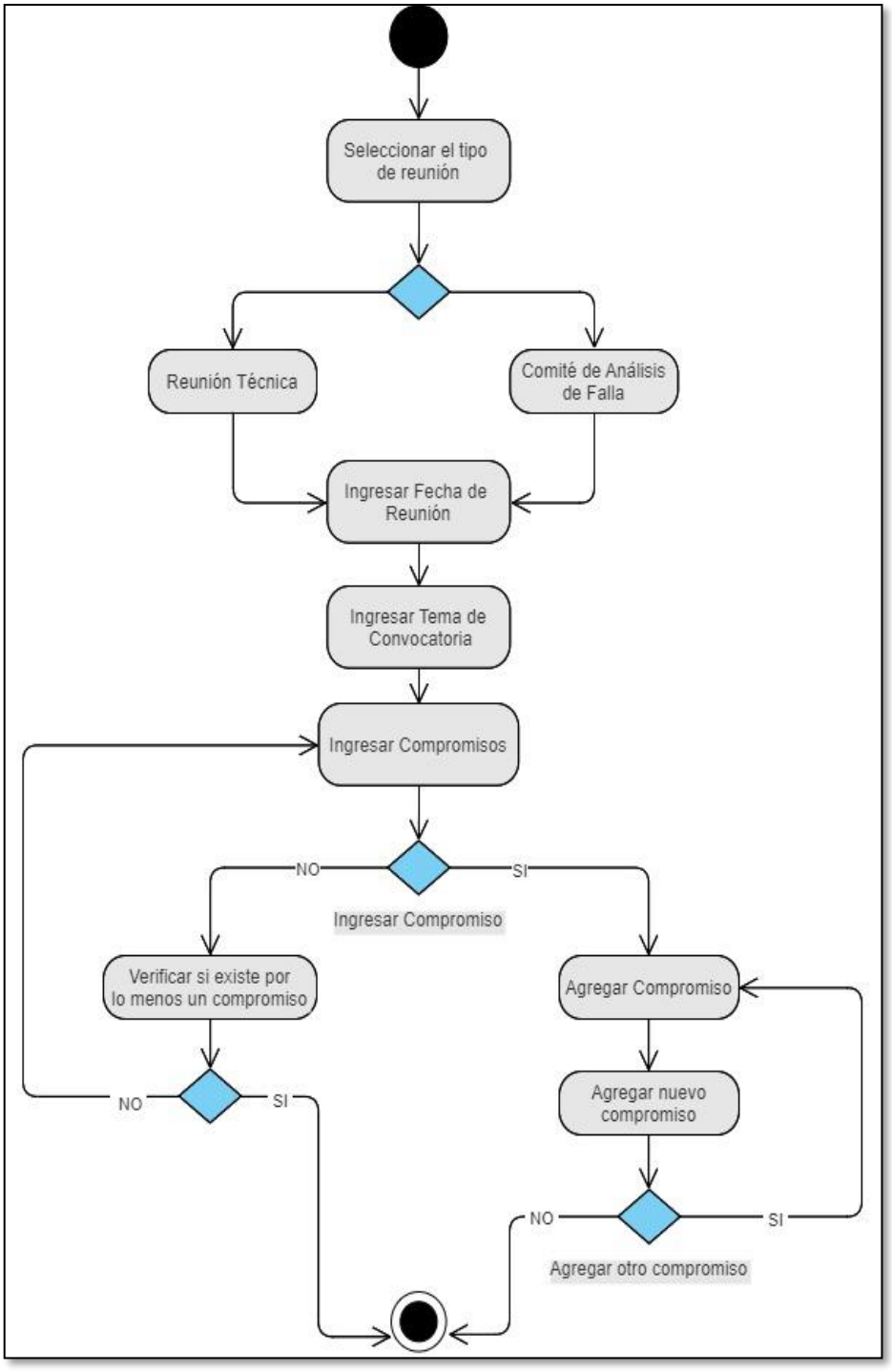

Figura D.0.1 Diagrama de actividades para agregar una reunión

Agregar Compromisos

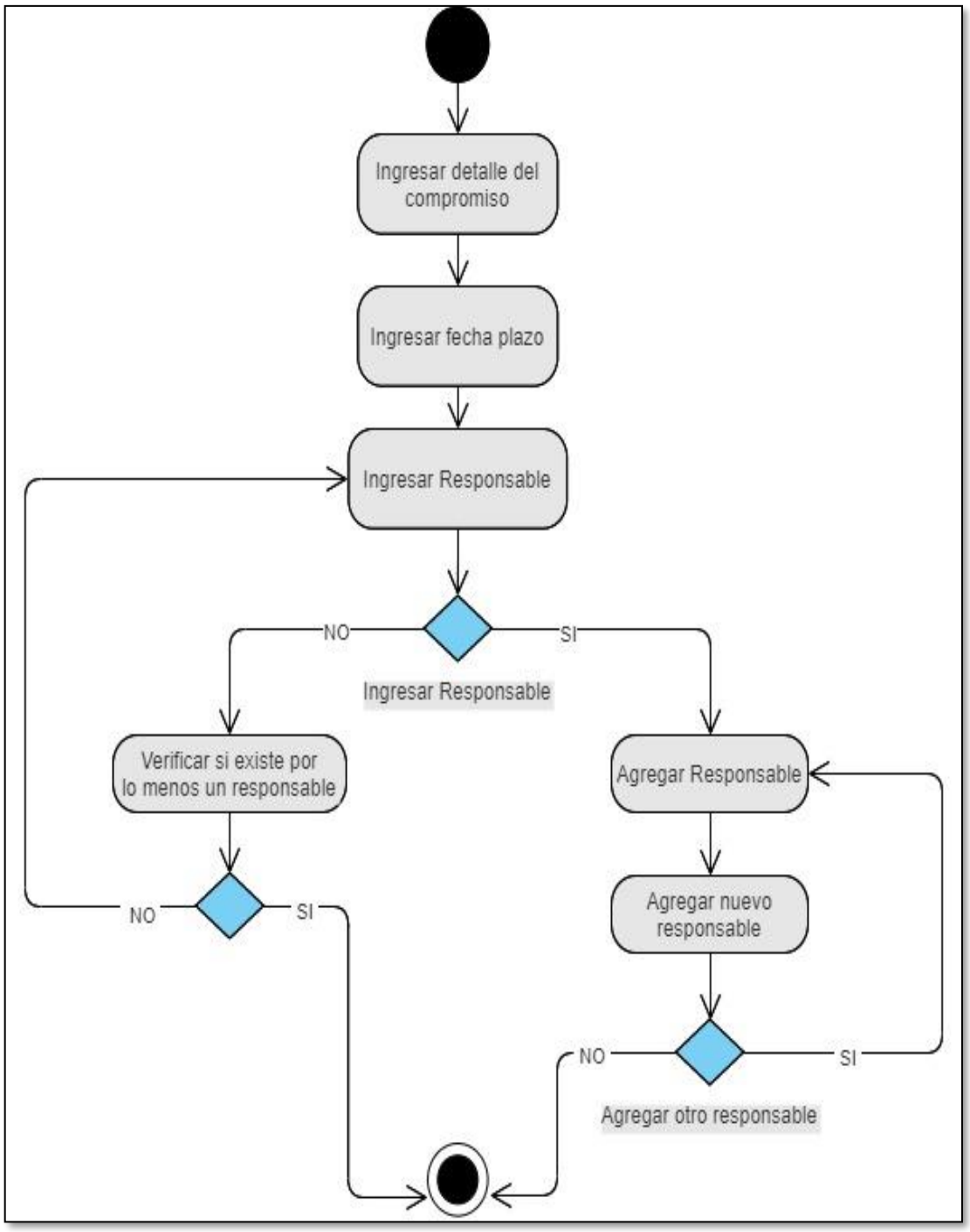

Figura D.0.2 Diagrama de actividades para agregar un compromiso

Agregar Responsable

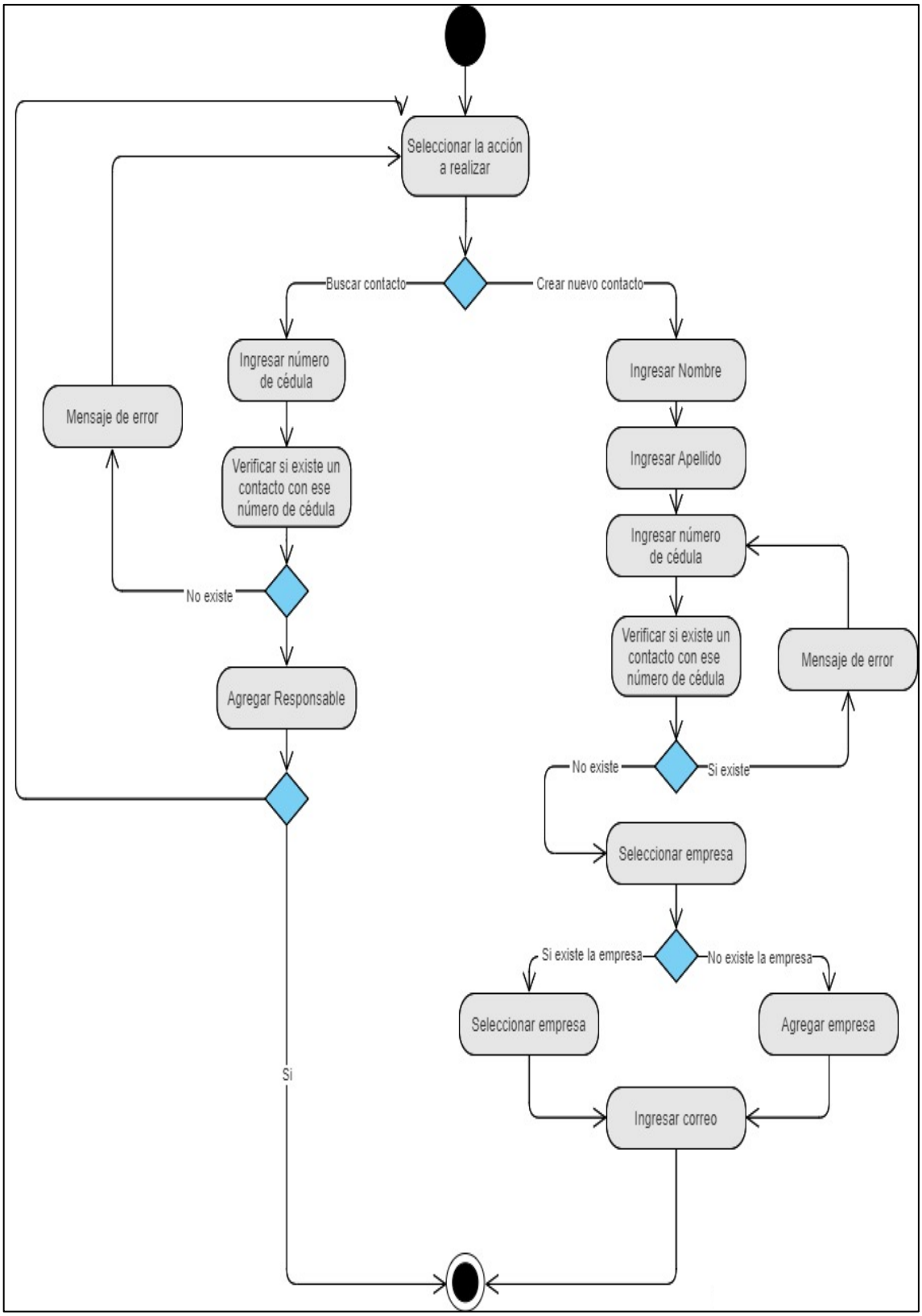

Figura D.0.3 Diagrama de actividades para agregar un responsable

Agregar Empresa

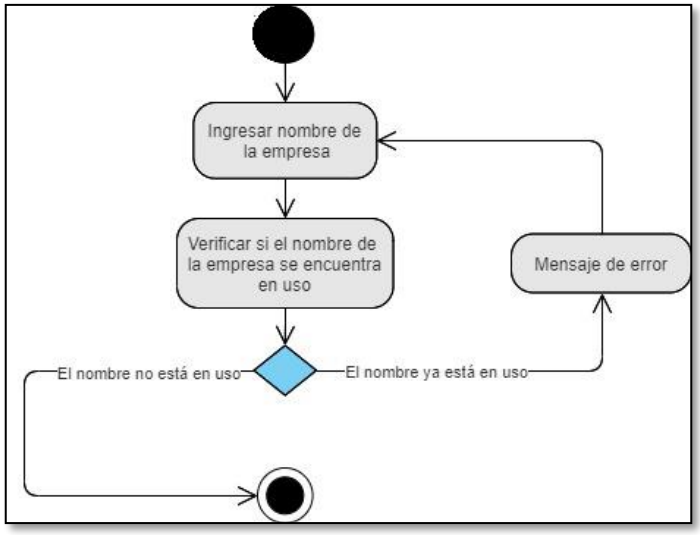

Figura D.0.4 Diagrama de actividades para agregar una empresa

Administrar Reunión

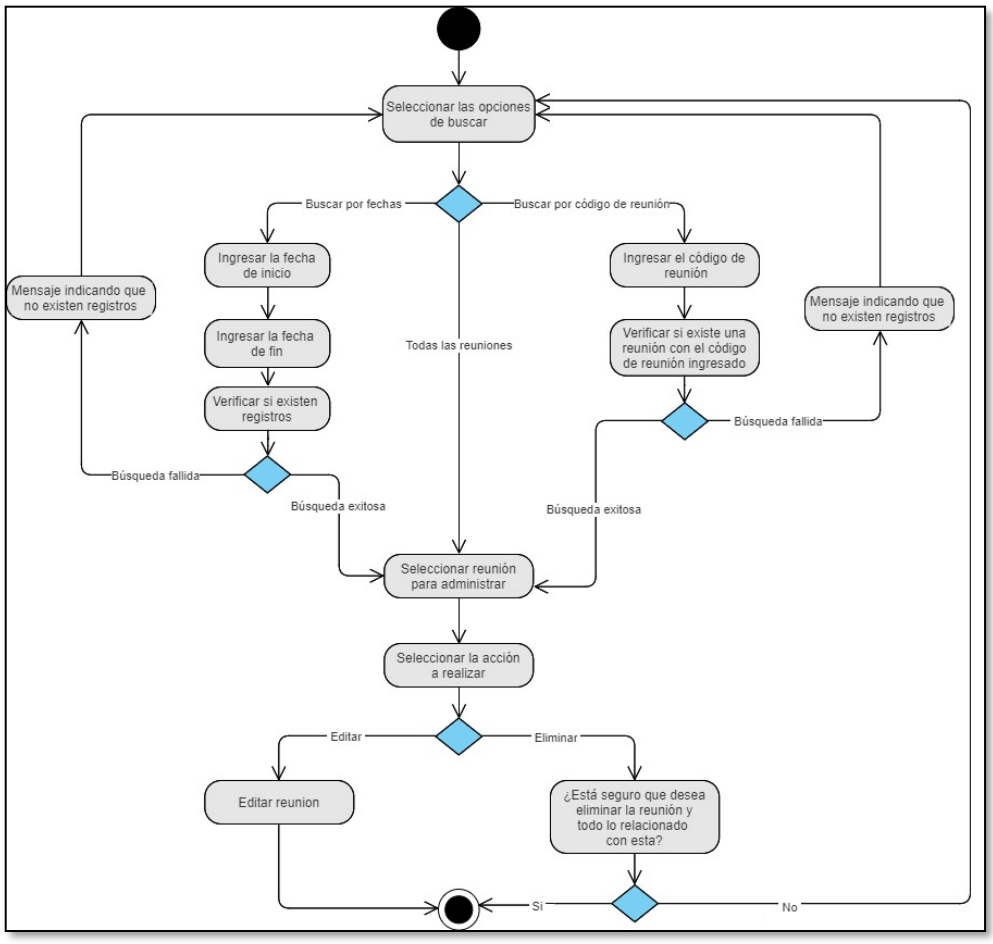

Figura D.0.5 Diagrama de actividades para administrar una reunión

Editar Reunión

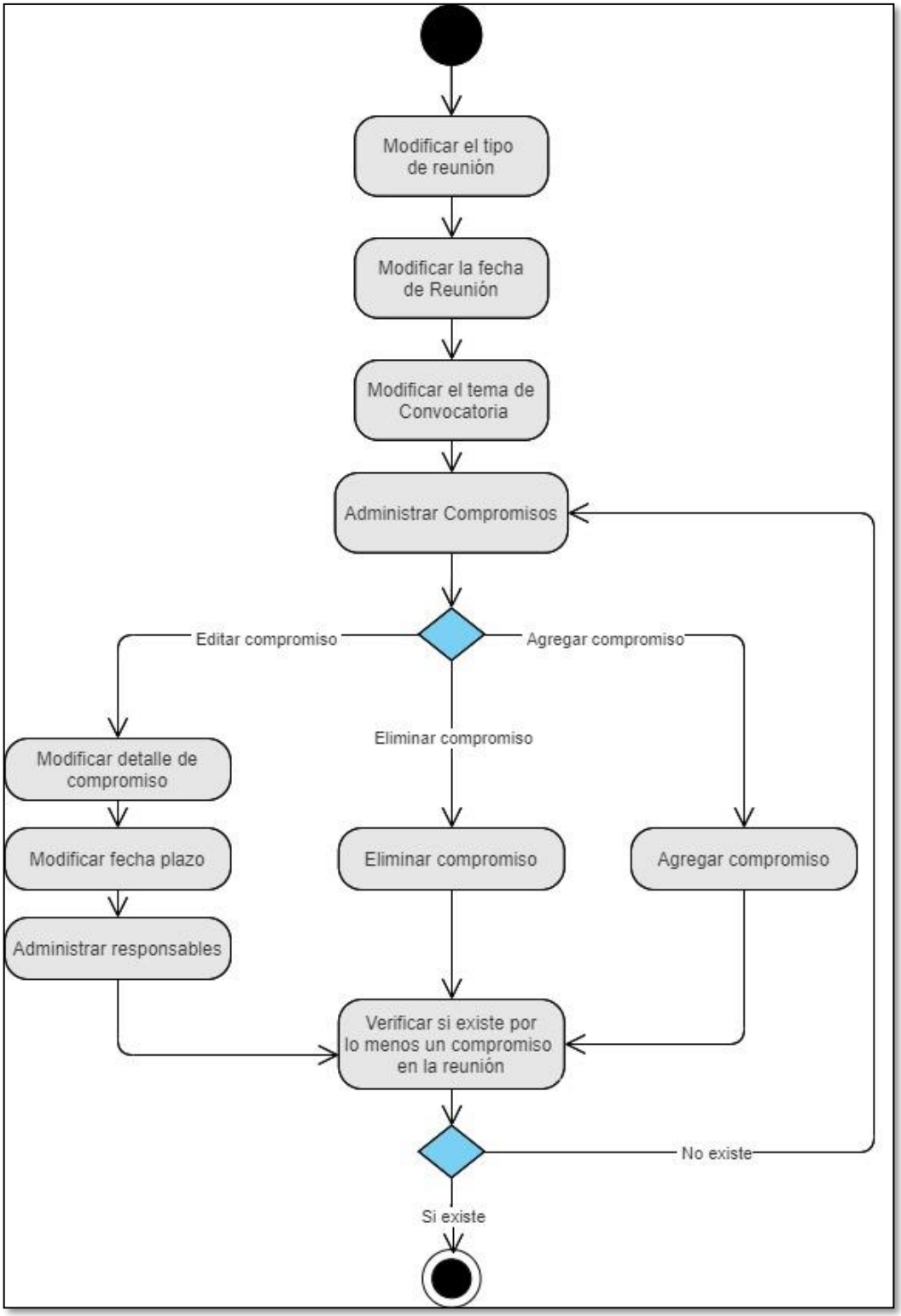

Figura D.0.6 Diagrama de actividades para editar reunión

Administrar responsables

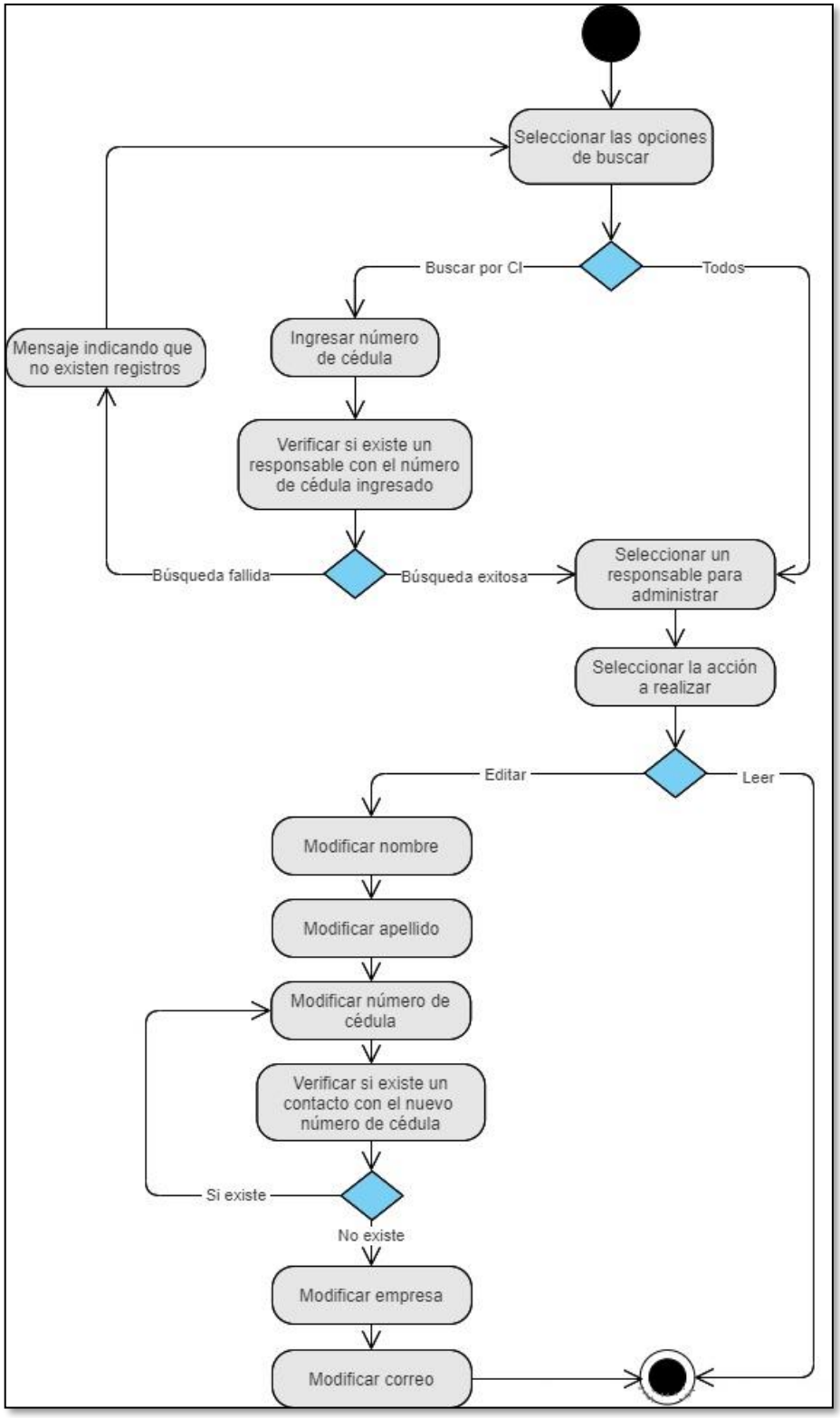

Figura D.0.7 Diagrama de actividades para administrar responsables

Administrar empresas

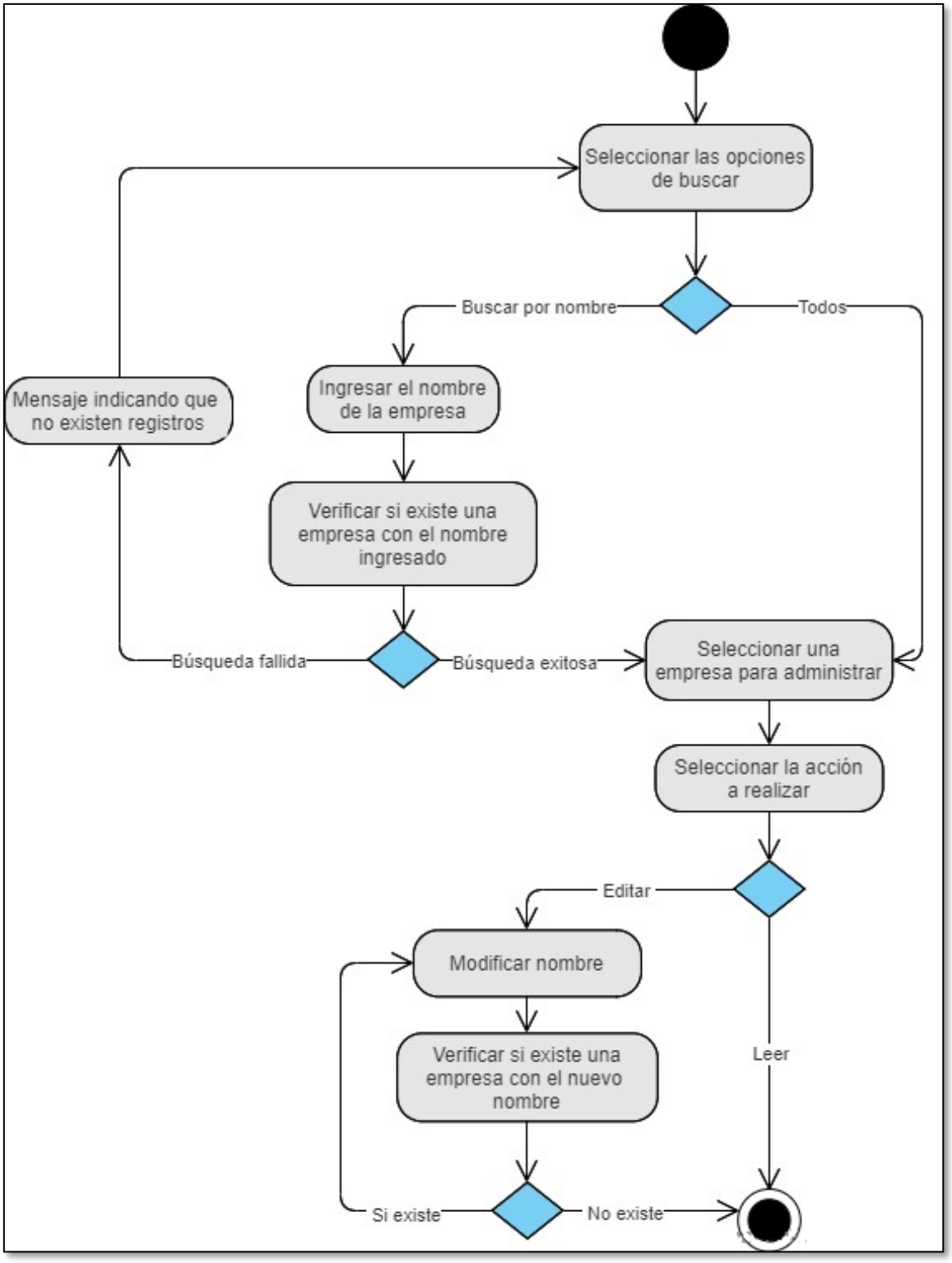

Figura D.0.8 Diagrama de actividades para administrar empresas

Seguimiento de compromisos

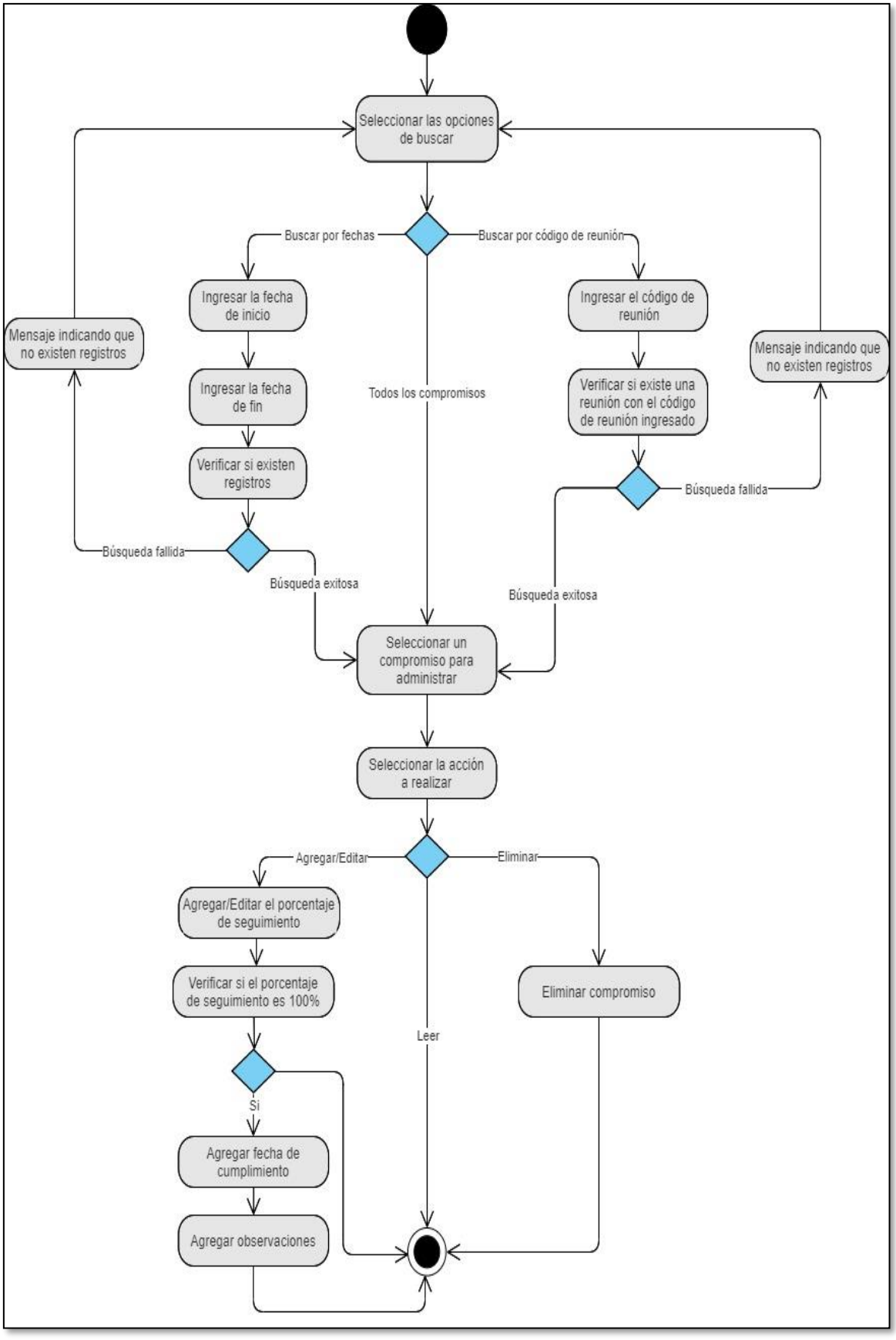

Figura D.0.9 Diagrama de actividades para seguimiento de compromisos

Editar configuración

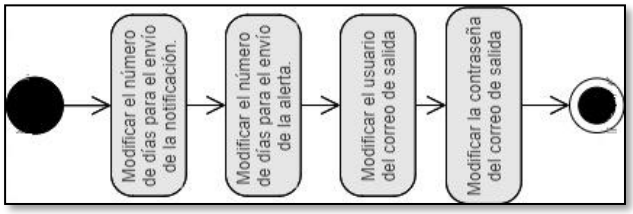

Figura D.0.10 Diagrama de actividades para editar configuración

Administración notificaciones de inicio

Al utilizar el prototipo se le puede presentar al usuario notificaciones de inicio de los compromisos próximos a caducarse y caducados. El usuario podrá administrar estas listas de compromisos como se indica en la Figura D.0.11.

En la Figura D.0.11 se puede observar que en la notificación “Próximos compromisos”, se verifica en función de ‘n’ días. La variable ‘n’ está dado por la configuración del prototipo como el número de días antes de la fecha plazo de un compromiso para el envío de una notificación.

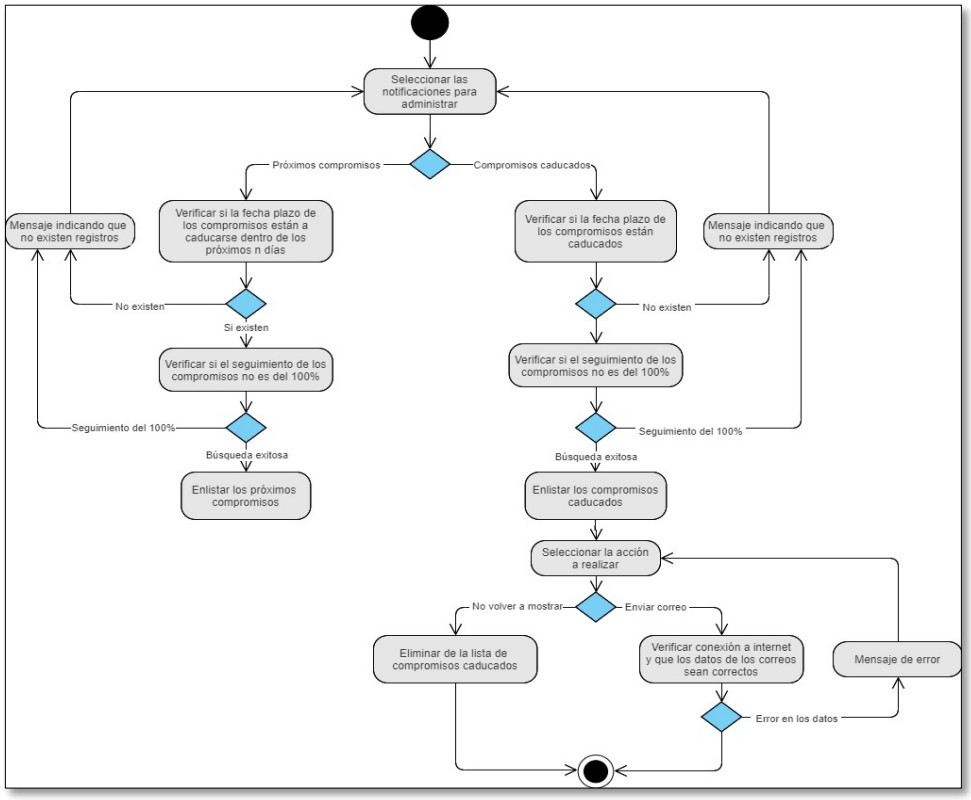

Figura D.0.11 Diagrama de actividades para administrar notificaciones de inicio

Notificaciones del sistema

El sistema envía de manera automática una notificación antes y después de la fecha plazo de un compromiso a sus responsables correspondientes según el número de días propuestos en la configuración general. La Figura D.0.12 describe esta acción.

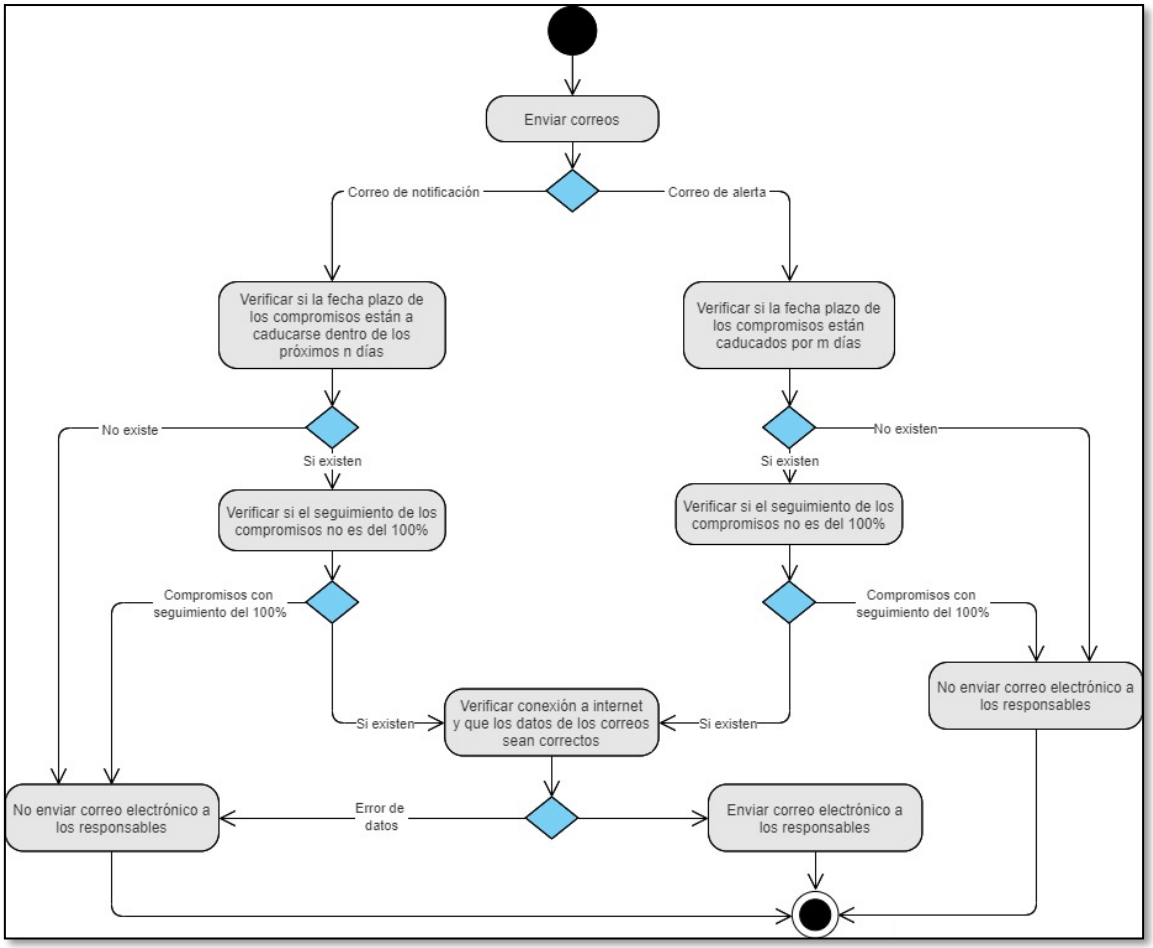

Figura D.0.12 Diagrama de actividades para notificaciones de sistema

ANEXO E

DIAGRAMA DE SECUENCIA

Agregar Reunión

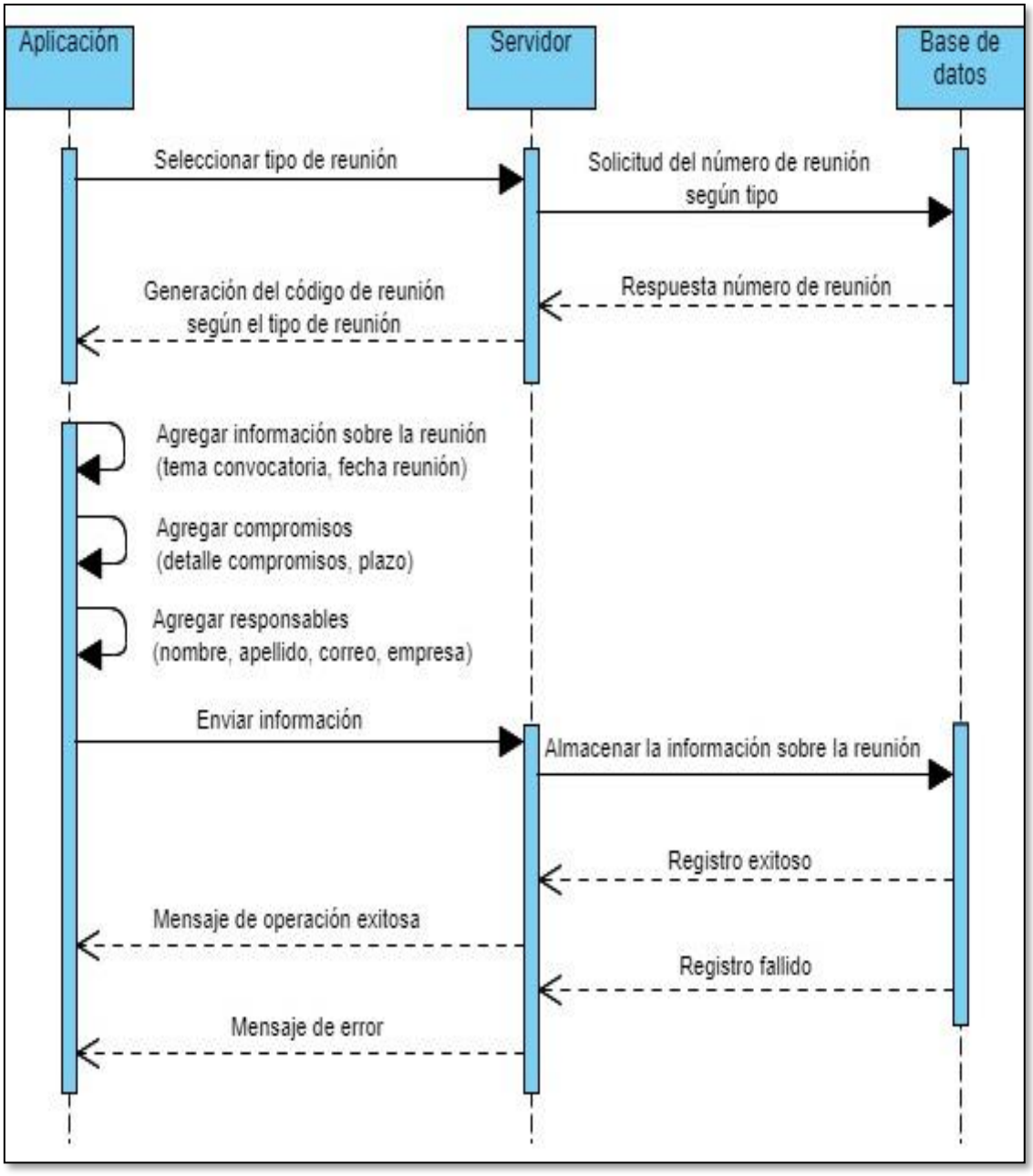

Figura E.0.1 Diagrama de secuencias para agregar reunión

Agregar Responsable

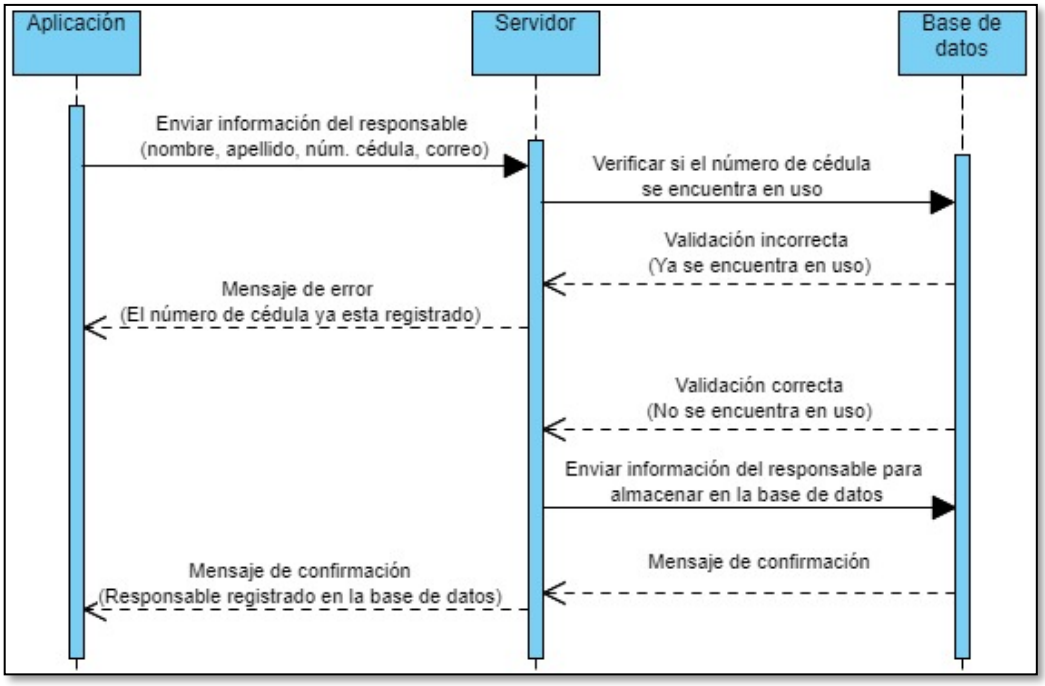

Figura E.0.2 Diagrama de secuencia para agregar responsable

Agregar Empresa

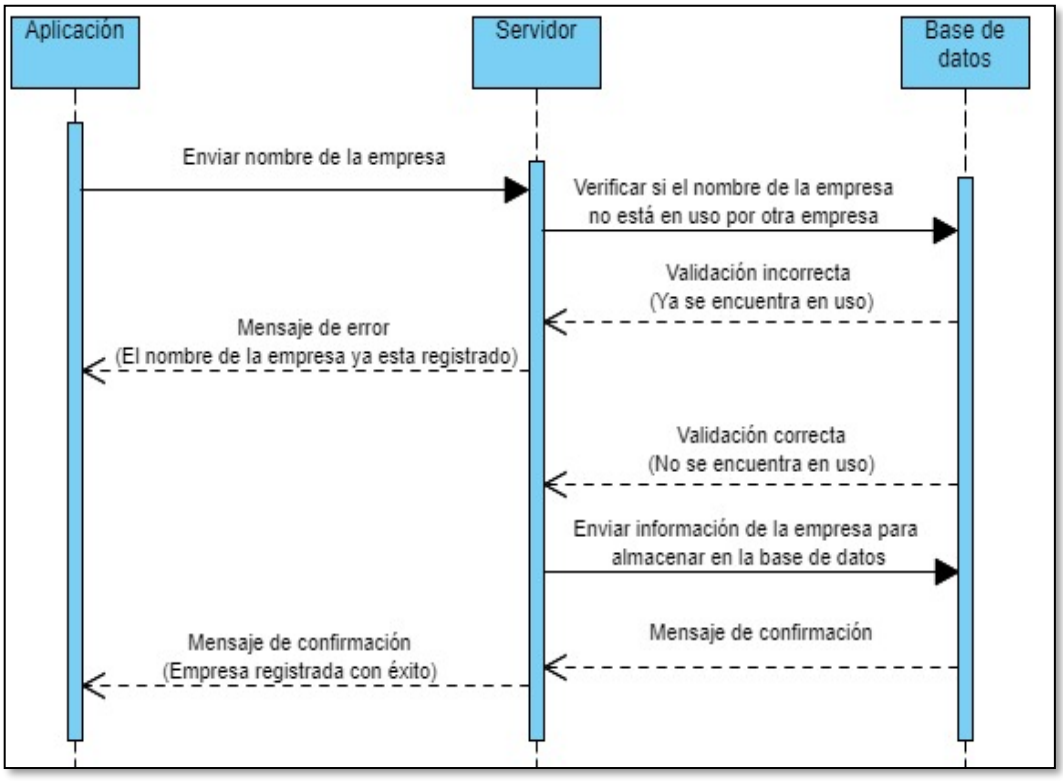

Figura E.0.3 Diagrama de secuencia para agregar empresa

Buscar Reunión

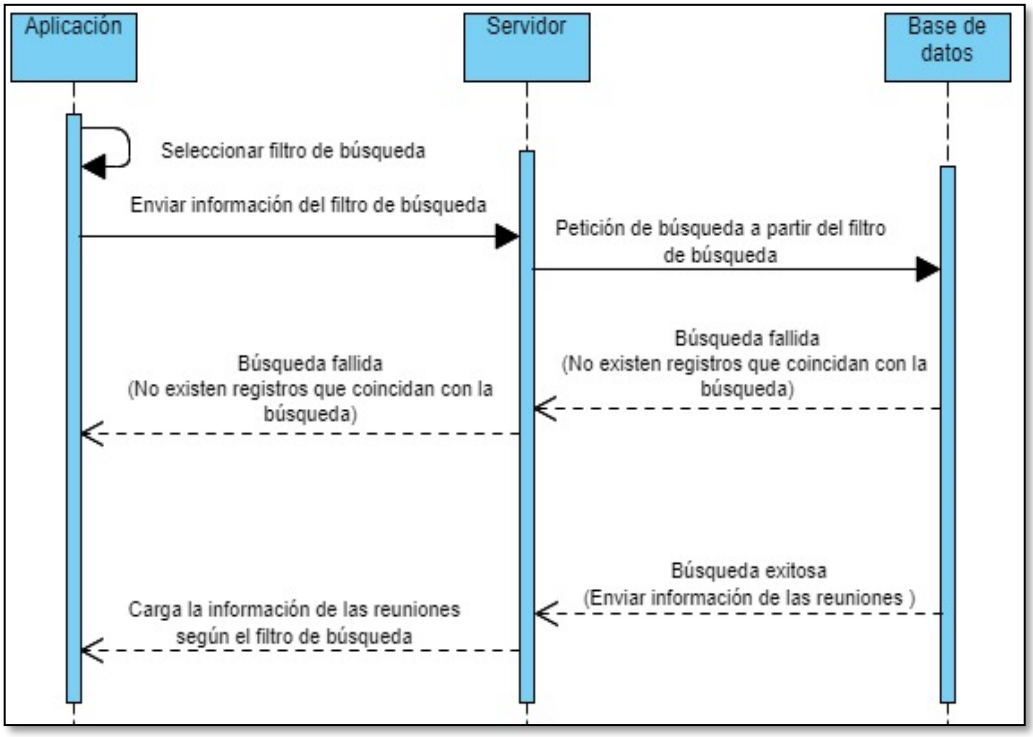

Figura E.0.4 Diagrama de secuencia para buscar reunión

Buscar Responsable

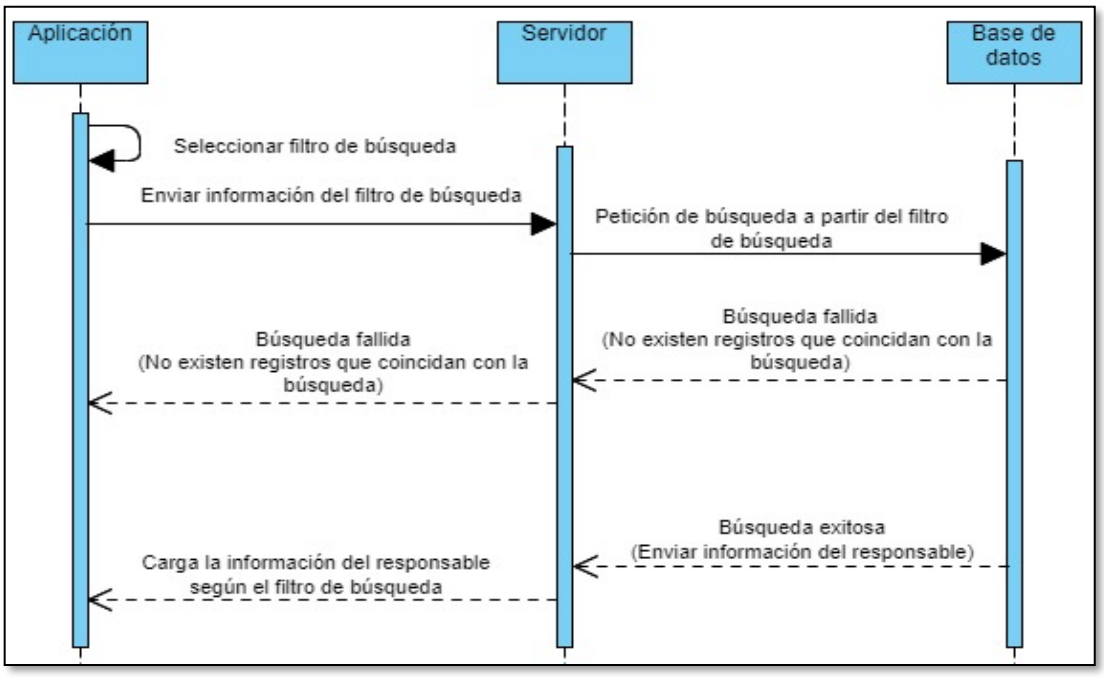

Figura E.0.5 Diagrama de secuencia para buscar responsable

Buscar Empresa

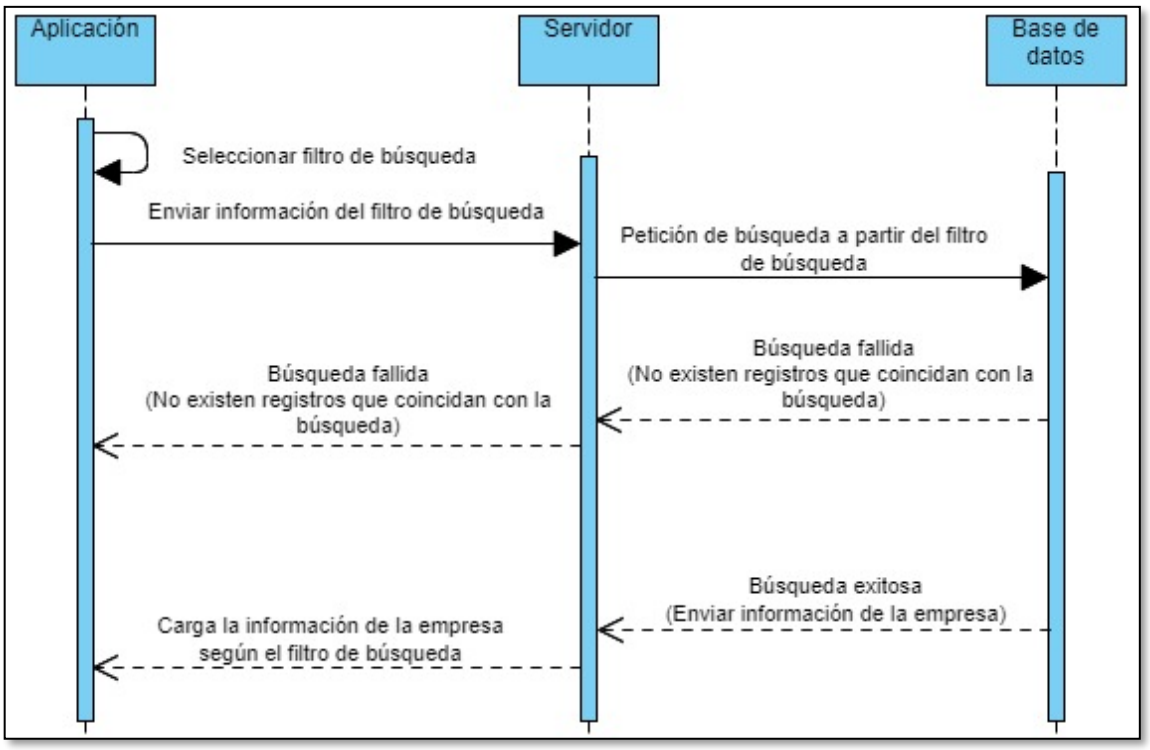

Figura E.0.6 Diagrama de secuencia para buscar empresa

Buscar compromisos

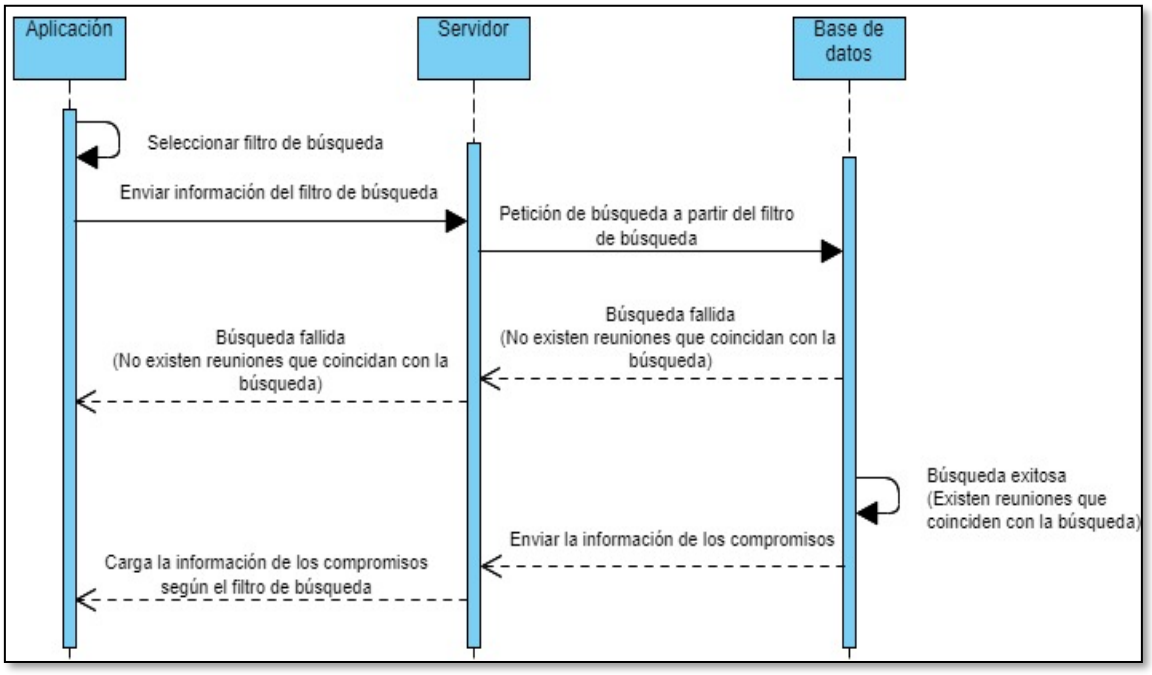

Figura E.0.7 Diagrama de secuencia para buscar compromisos

Modificar reunión

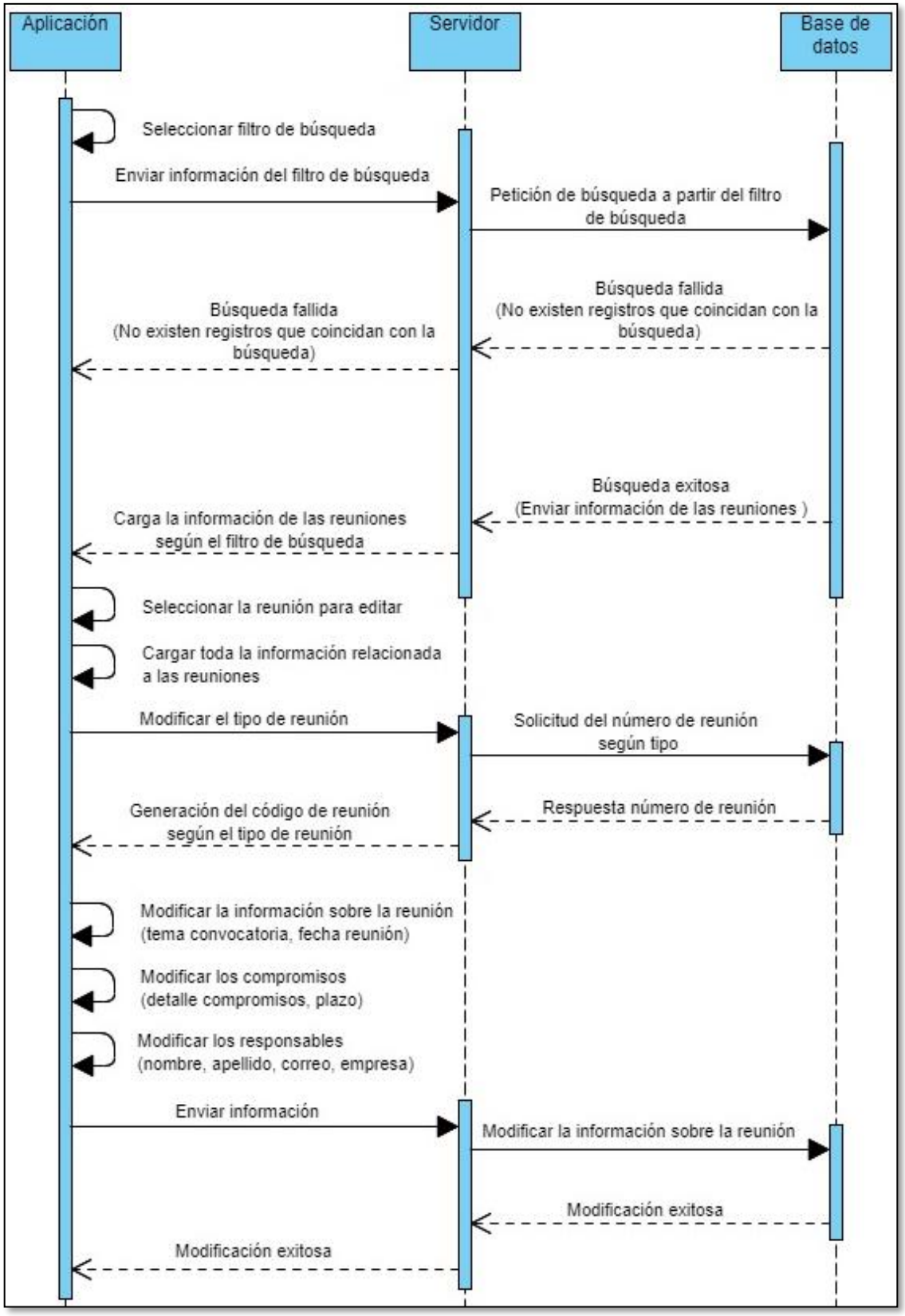

Figura E.0.8 Diagrama de secuencia para modificar reunión

Modificar Responsable

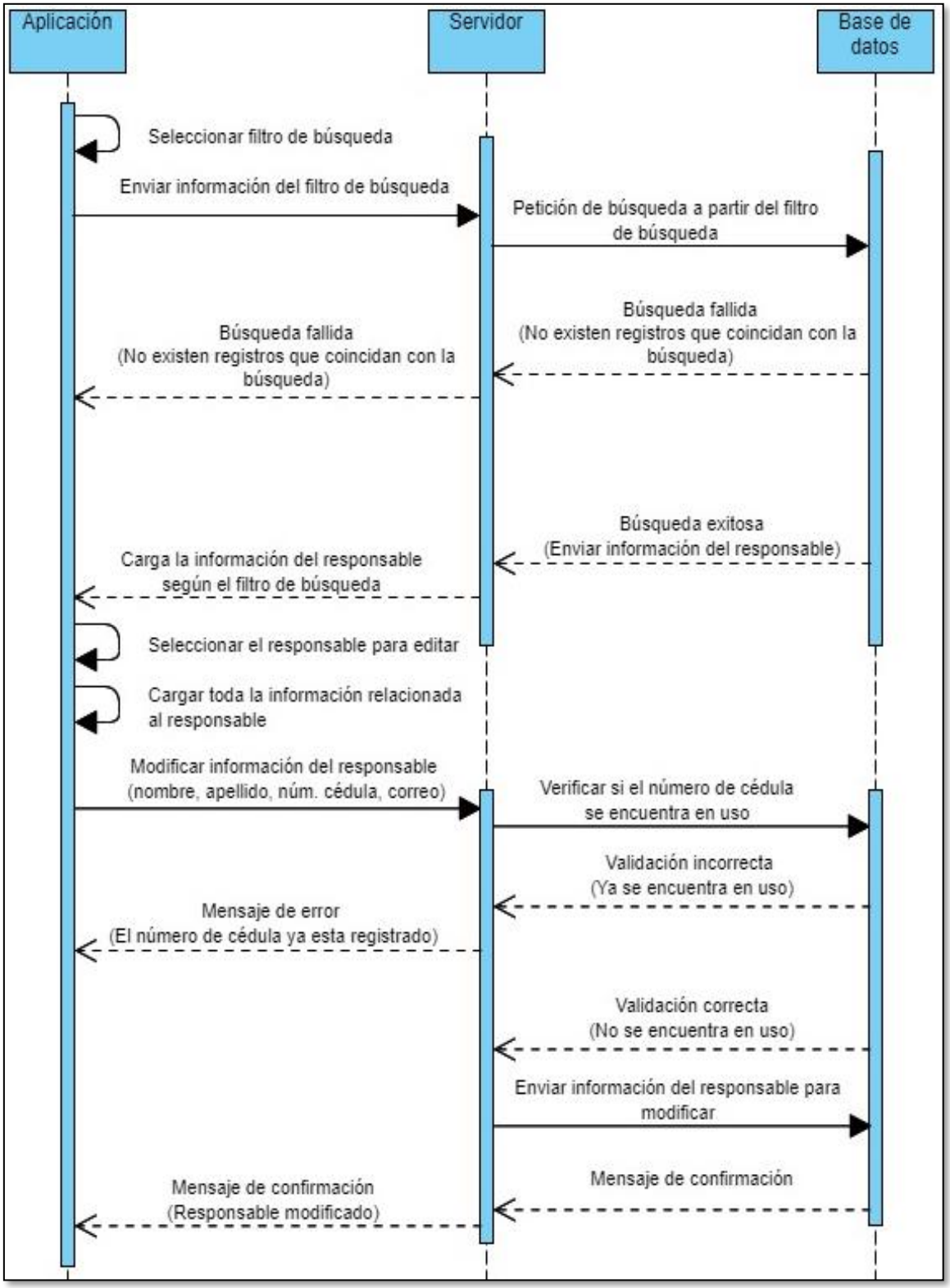

Figura E.0.9 Diagrama de secuencia para modificar responsable

Modificar Empresa

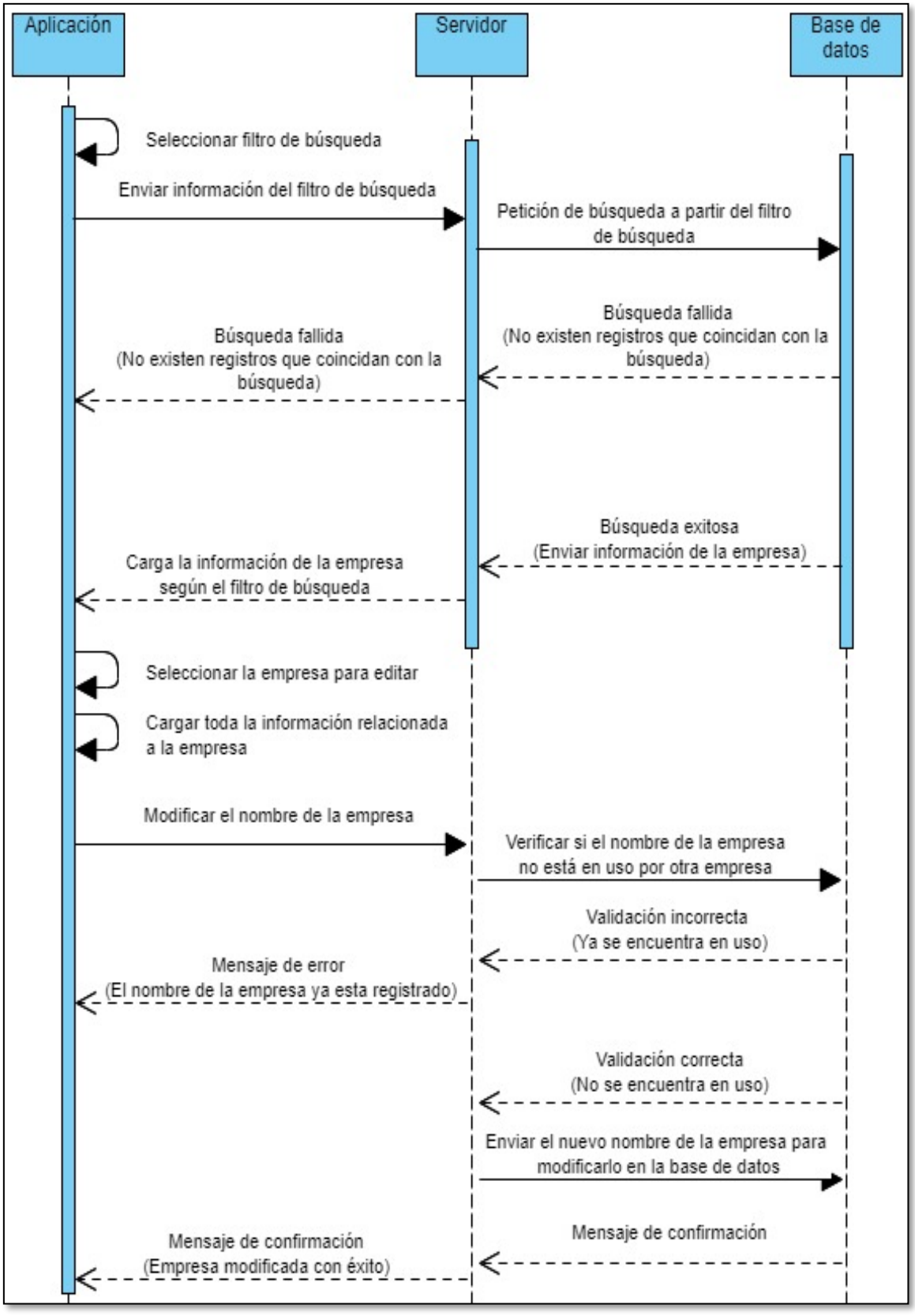

Figura E.0.10 Diagrama de secuencia para modificar empresa

Modificar Configuración

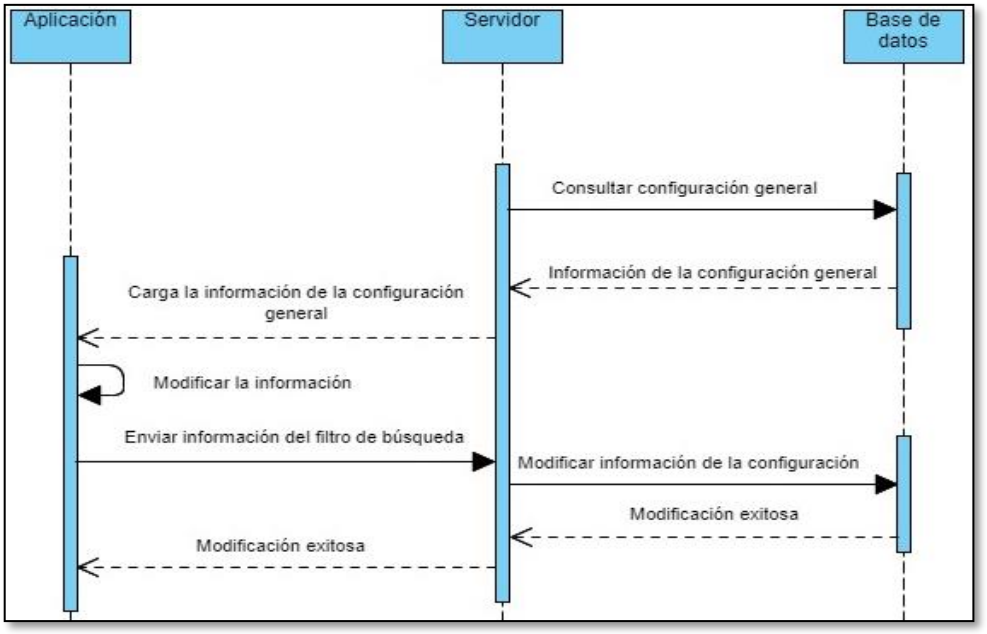

Figura E.0.11 Diagrama de secuencia para modificar configuración

Eliminar Reunión

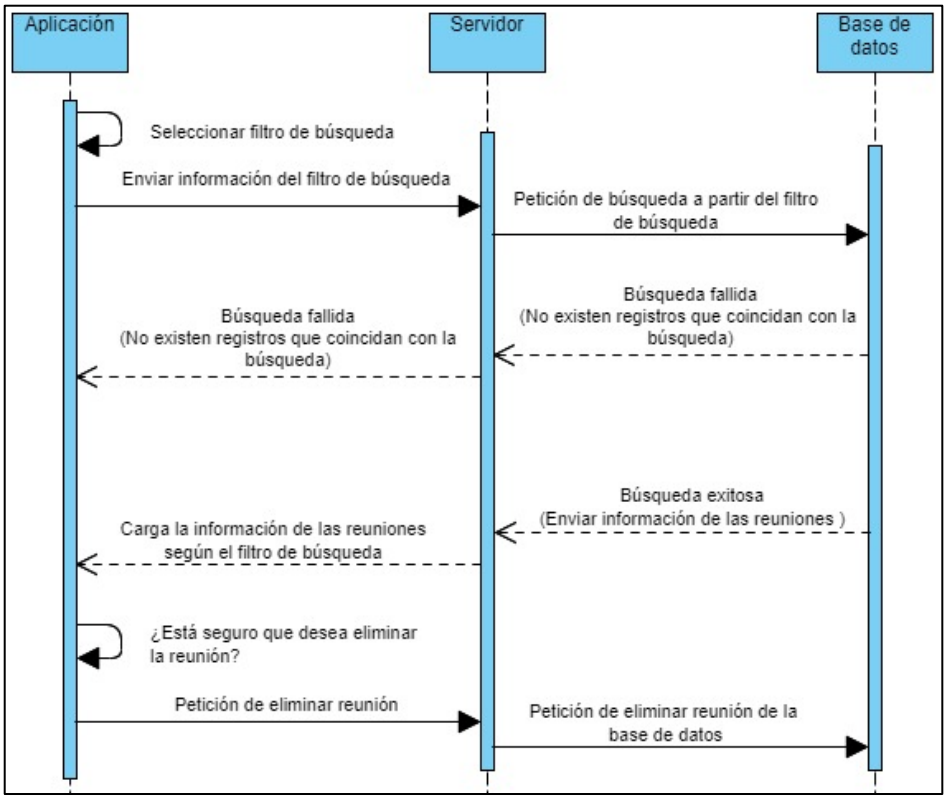

Figura E.0.12 Diagrama de secuencia para eliminar reunión

Seguimiento Compromisos

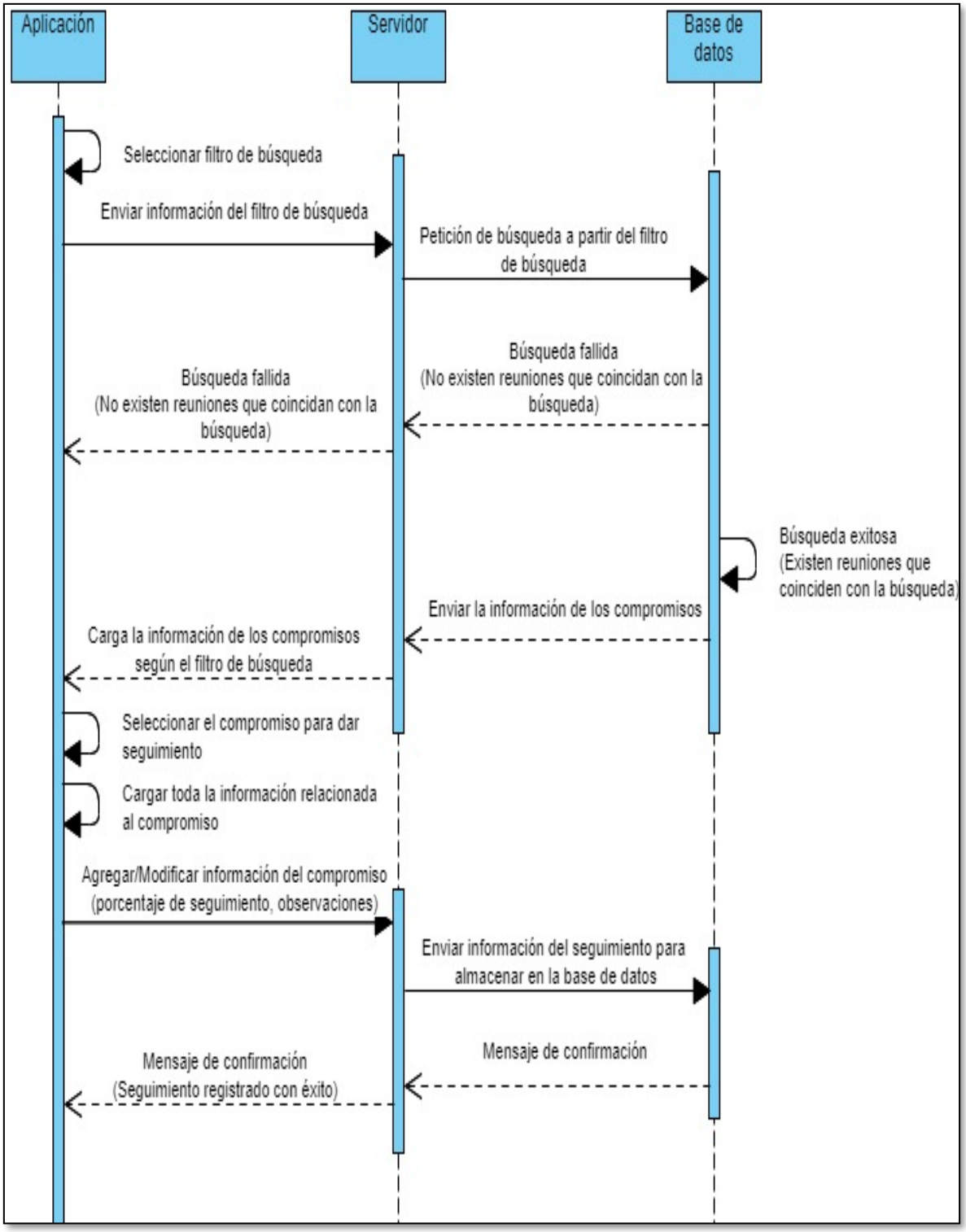

Figura E.0.13 Diagrama de secuencia para seguimiento de compromisos

Notificación próximos compromisos

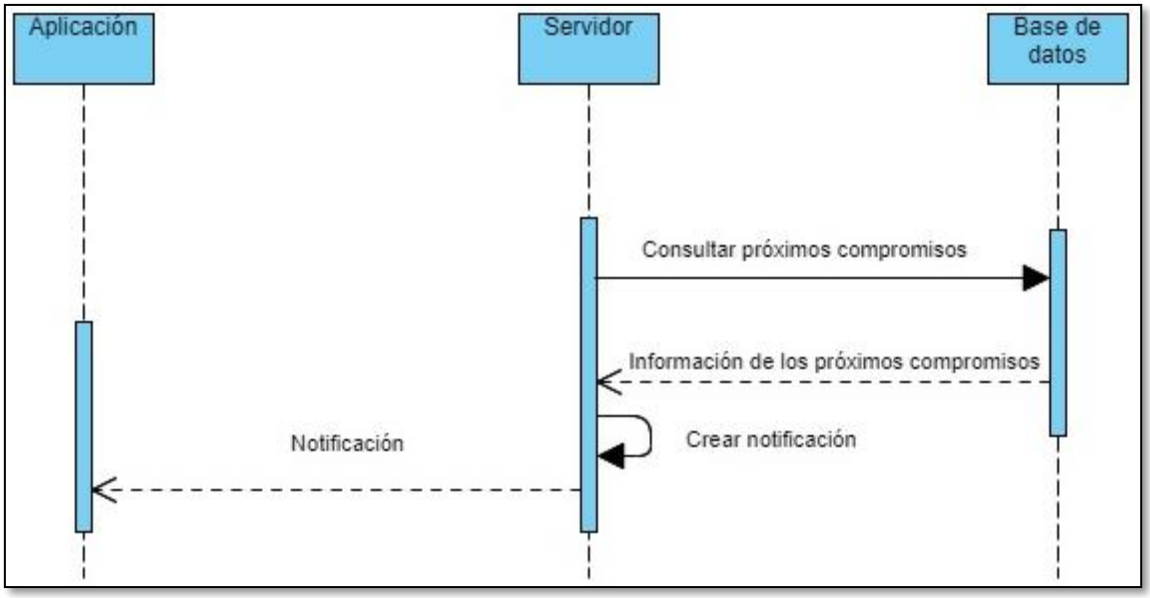

Figura E.0.14 Diagrama de secuencia para notificación próximos compromisos

Notificación compromisos caducados

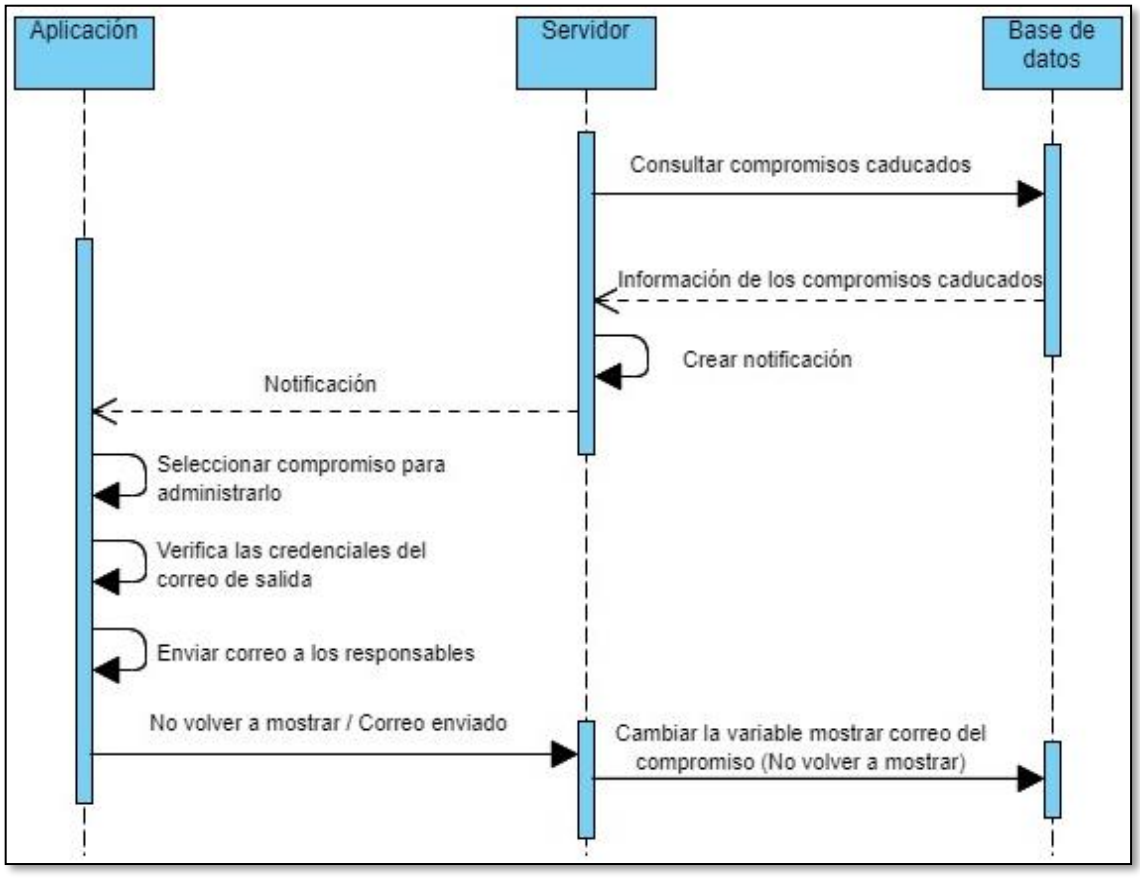

Figura E.0.15 Diagrama de secuencia para notificación compromisos caducados

Envío de alertas

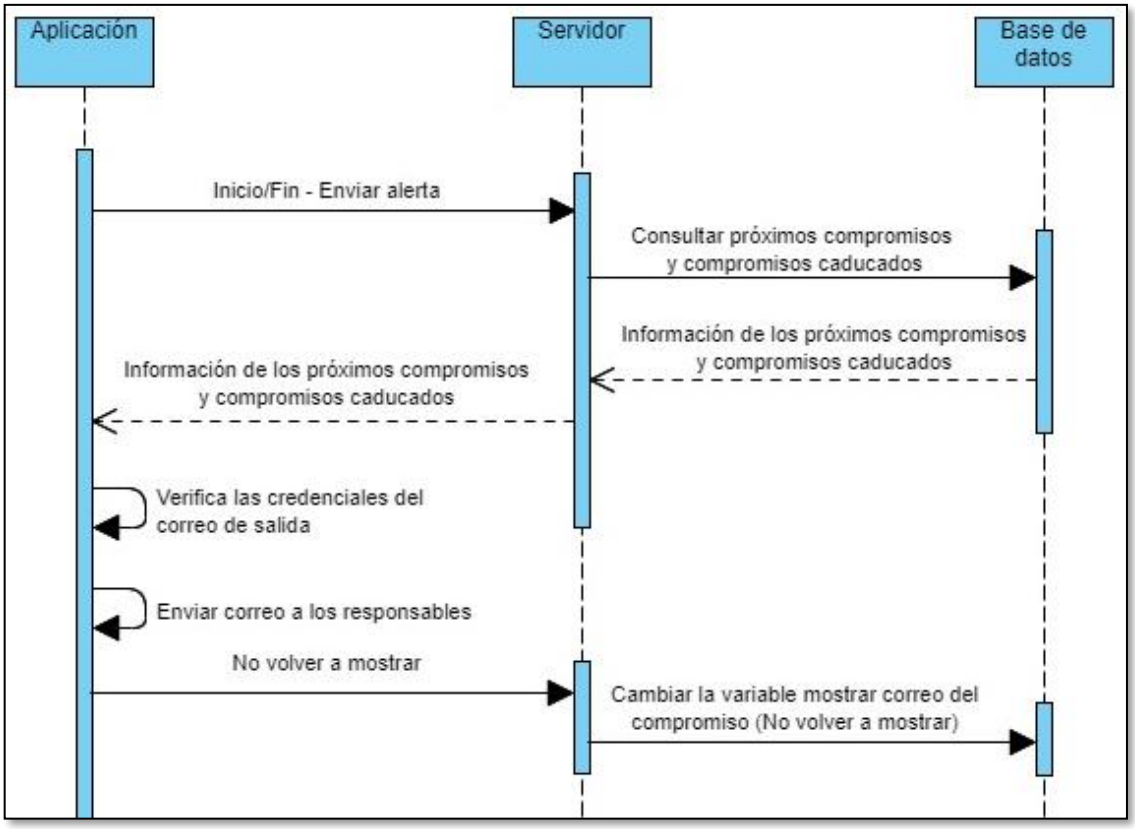

Figura E.0.16 Diagrama de secuencia para envío de alertas

ANEXO F

SKETCHES

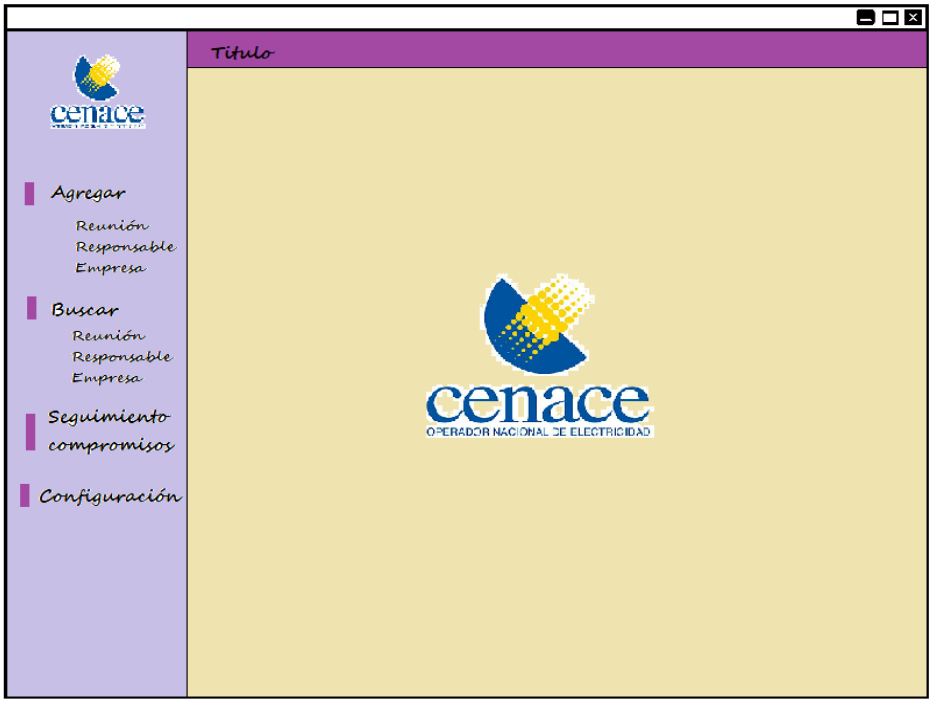

Figura F.0.1 *Sketch* de la pantalla principal

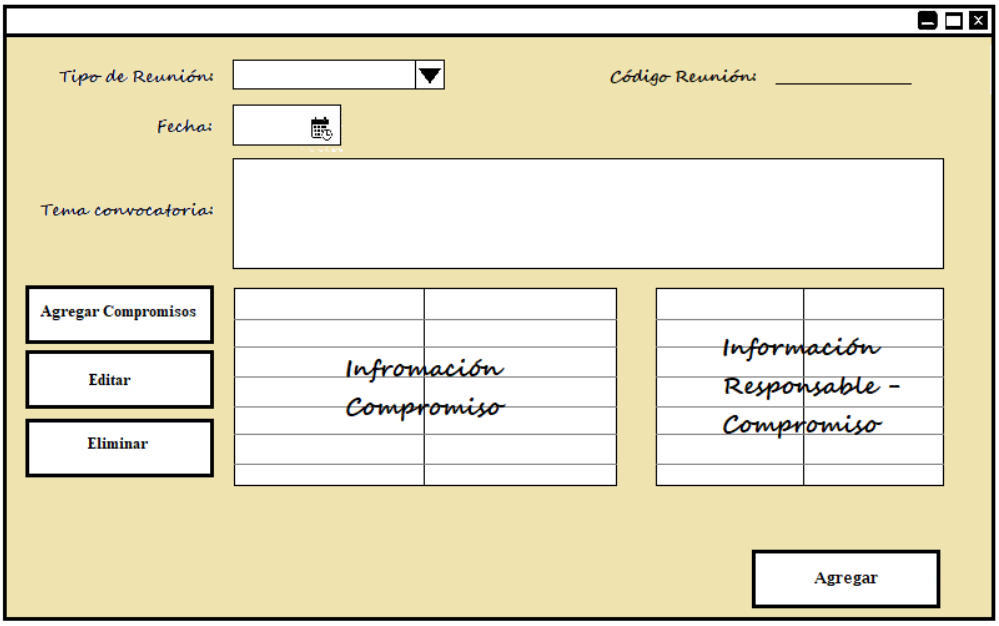

Figura F.0.2 *Sketch* para agregar o editar una reunión

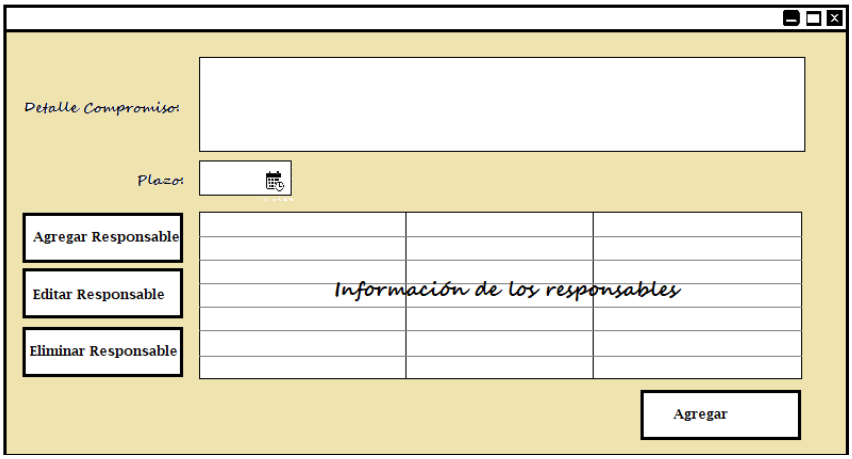

Figura F.0.3 *Sketch* para agregar o editar un compromiso

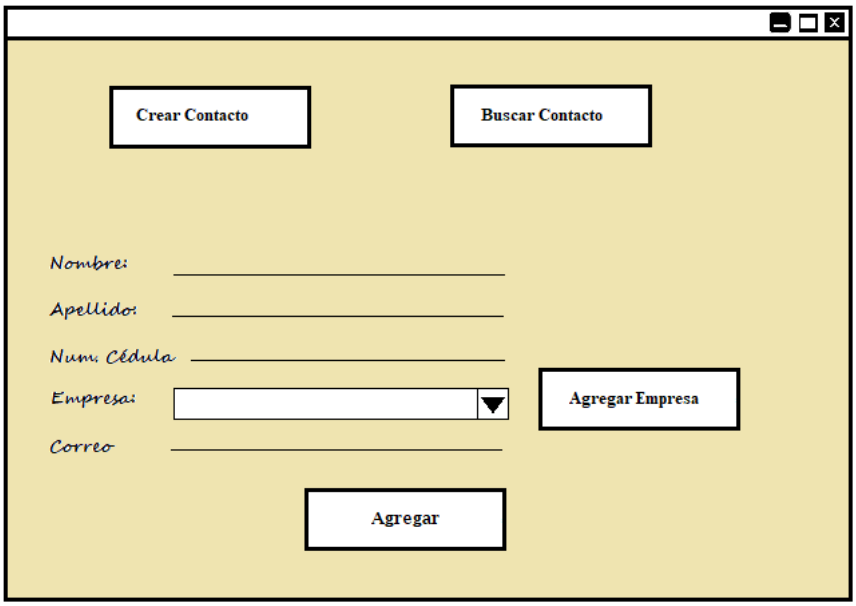

Figura F.0.4 *Sketch* para agregar o editar un responsable

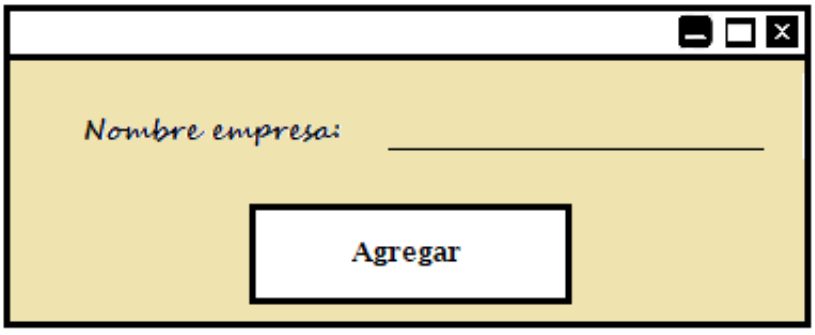

Figura F.0.5 *Sketch* para agregar o editar una empresa

Figura F.0.6 *Sketch* para buscar una reunión

Figura F.0.7 *Sketch* para buscar un responsable

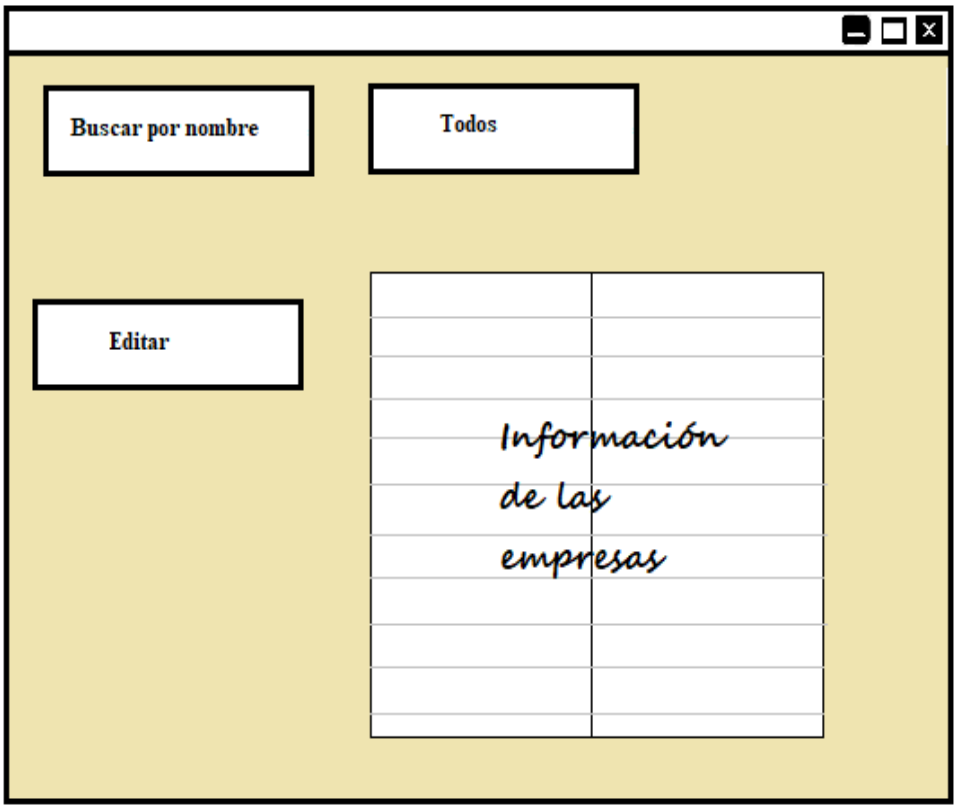

Figura F.0.8 *Sketch* para buscar una empresa

Figura F.0.9 *Sketch* para buscar un compromiso

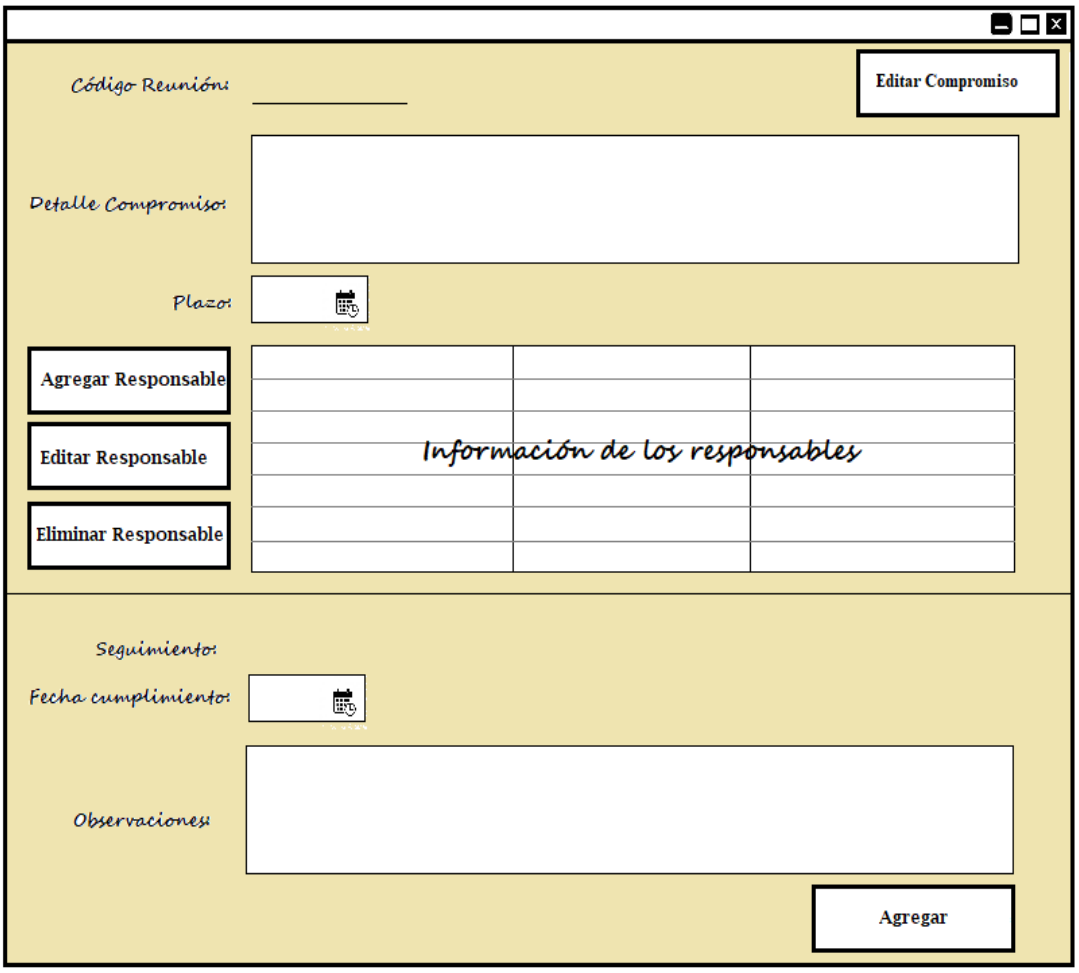

Figura F.0.10 *Sketch* para agregar seguimiento de un compromiso

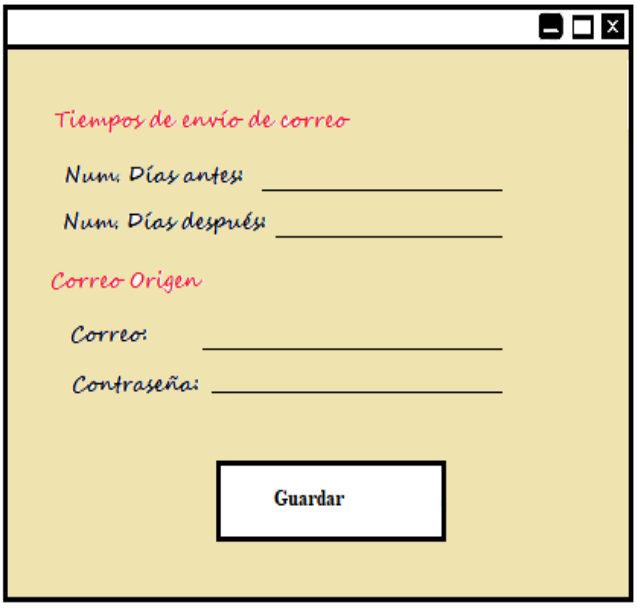

Figura F.0.11 *Sketch* para modificar la configuración del prototipo.

Figura F.0.12 *Sketch* para compromisos caducados.

ANEXO G

CÓDIGO DEL PROTOTIPO

Debido a la extensión, el código del prototipo se presenta de forma digital en el CD ajunto a este documento.

ANEXO H

MANUAL DE USUARIO

Debido a la extensión, el manual de usuario se presenta de forma digital en el CD ajunto a este documento.

ANEXO I

En este anexo se presenta el modelo de encuesta de satisfacción de los requerimientos del prototipo.

1. **La aplicación, ¿le permitió agregar una nueva reunión al sistema?**
   - Sí
   - No
2. **La aplicación, ¿generó de manera automática el código de reunión al momento de seleccionar el tipo de reunión?**
   - Sí
   - No
3. **La aplicación, ¿le permitió agregar uno o varios compromisos a una reunión específica?**
   - Sí
   - No
4. **La aplicación, ¿le permitió agregar uno o varios responsables a un compromiso específico?**
   - Sí
   - No
5. **La aplicación, ¿le permitió agregar o seleccionar una empresa para asociarla con un responsable de un compromiso?**
   - Sí
   - No
6. **La aplicación, ¿le permitió buscar un responsable anteriormente registrado para asociarlo con un compromiso?**
   - Sí
   - No
7. **La aplicación, ¿le permitió editar una reunión y toda la información relacionada a ella (Lista de compromisos y lista de responsables de compromisos)?**
   - Sí
   - No
8. **La aplicación, ¿le permitió visualizar la lista de reuniones almacenadas en el sistema?**
   - Sí

- No

9. **La aplicación, ¿le permitió visualizar la lista de empresas almacenadas en el sistema?**
   - Sí
   - No
10. **La aplicación, ¿le permitió visualizar la lista de responsables almacenados en el sistema?**
    - Sí
    - No
11. **La aplicación, ¿le permitió visualizar la lista de empresas almacenadas en el sistema?**
    - Sí
    - No
12. **La aplicación, ¿le permitió visualizar la lista de compromisos de una reunión almacenada en el sistema?**
    - Sí
    - No
13. **La aplicación, ¿le permitió dar seguimiento a un compromiso?**
    - Sí
    - No
14. **La aplicación, ¿le permitió visualizar y modificar la información sobre la configuración general del sistema?**
    - Sí
    - No
15. **¿La aplicación envía una notificación por correo electrónico a los responsables de compromisos próximos a caducar?**
    - Sí
    - No
16. **¿La aplicación muestra una lista de compromisos que se encuentra próximos a caducar y caducados?**
    - Sí
    - No
17. **Del 1 al 5, siendo 1 poco amigable y 5 muy amigable ¿Qué tan intuitiva y amigable fue la interfaz de usuario?**

    1 2 3 4 5

18. Del 1 al 5, siendo 1 muy mala y 5 muy buena ¿Qué tan conforme se encuentra con el prototipo presentado?

1 2 3 4 5