Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Tesina de Diplomado

"Prototipo de Sistema de Gestión de Llamadas y Citas con Clientes" (SGCall)

Para obtener el título de:

Licenciado en Ingeniería en Ciencias de la Computación.

Presenta:

Omar Yafer García Tecpa

Bajo la Asesoría de:

M.C. Jorge Gustavo Jiménez González

PUEBLA, PUE. FEBRERO DE 2013

Tabla de contenido

CAPÍTULO I. Introducción 4

I.1 Resumen 5

I.2 Planteamiento del Problema 5

I.3 Objetivos del Proyecto 6

I.3.1 Objetivos Generales 6

I.3.2 Objetivos Específicos 6

I.4 Metodología 6

I.5 Resultados 8

CAPÍTULO II. Marco de Referencia Teórico 9

II.1 Modelado del Análisis Estructurado 10

II.1.1 Modelado de Datos 10

II.1.2 Modelo Conceptual de la Base de Datos 10

II.1.3 Diagrama Entidad Relación 10

II.1.4 Bases de Datos Relacionales 11

II.1.5 Modelado Funcional 12

II.1.6 Modelado de Comportamiento 13

II.2 Diseño Lógico de la Base de Datos 15

II.2.1 Transformación del Modelo Conceptual de la Base de Datos al Modelo Lógico 15

II.2.2 Normalización 16

II.3 Lenguajes de Definición y Manipulación de Datos 17

II.3.1 SQL 17

II.4 Sistema Gestor de Bases de Datos 17

II.4.1 MySQL 17

II.5 Lenguajes para el Desarrollo de Páginas Web Dinámicas 18

II.5.1 HTML 18

II.5.2 CSS 18

II.5.3 PHP 18

II.6 Servidores Web (Software) 18

Apache 18

CAPÍTULO III. Especificación de los Requisitos del Sistema 19

III.1 Descripción General del Sistema 20

III.2 Objetivos 20
III.3 Alcance 20
III.4 Casos de Uso 21
III.4.1 Actores 21
III.4.2 Documentación de los Casos de Uso 22
III.4.3 Reportes 22
III.4.4 Modelo de Casos de Uso 23
III.5 Requerimientos Funcionales 24
III.6 Requerimientos No Funcionales 25
CAPÍTULO IV. Análisis y Diseño del Sistema 26
IV.1 Análisis del Sistema 27
IV.1.1 Modelo Conceptual de la Base de Datos del Sistema 27
IV.1.2 Modelo Funcional y Diagramas de Flujo de Datos (DFD's) 32
IV.2 Diseño Lógico. 34
IV.2.1 Esquema Relacional de la Base de Datos. 34
IV.2.2 Normalización 35
IV.3 Diseño de la Interfaz de Usuario. 40
CAPÍTULO V. Implementación 41
V.1 Mapa del Sitio 42
V.2 Instalación y Configuración de los Servidores Web y de Bases de Datos. 43
V.3 Creación de Tablas Utilizando el Lenguaje SQL 44
V.4 Estructura Final de la Base de Datos. 52
V.5 Consultas. 53
V.6 CSS. 57
V.7 Pantallas del Sistema 62
V.8 Reportes. 71
CAPÍTULO VI. Conclusiones y Trabajo a Futuro 73
VI.1 Conclusiones 74
VI.2 Trabajo a Futuro 75
Bibliografía 76

CAPÍTULO I. **Introducción**

I.1 Resumen

En este proyecto se pretende desarrollar una parte de un software CRM (Custumer Relationship Management) Básico denominado Sistema de Gestión de Llamadas y Citas con Clientes (SGCall) que permita controlar la información de los contactos, prospectos y clientes que tiene una empresa PYME y las citas que se agendan y realizan con ellos. El sistema será implementado en una Base de Datos Relacional y a la cual se podrá acceder mediante un navegador web.

I.2 Planteamiento del Problema

Una empresa es un conjunto de individuos que unen sus esfuerzos para alcanzar un beneficio económico(1). La **pequeña y mediana empresa (PYME)** es una empresa cuyas características están definidas por la región o país en la que ha sido constituida. En el caso de México una PYME es definida por el número de empleados y las ventas anuales con que cuenta la empresa; a partir de esta definición tendremos micro, pequeña y mediana empresa[(2)].

Las PYMEs conforman la base económica de nuestro país, debido a que son las más numerosas y que además generan una gran cantidad de empleos. Generalmente estas empresas están más enfocadas al mercado de comercio debido a las limitaciones legislativas, de infraestructura, tecnología e inversión que forman parte del mercado industrial.

En la actualidad existe un enorme mercado industrial, de servicios y comercial. Existe una gran cantidad de empresas que ofrecen servicios similares, por lo que es cada vez más difícil asegurar clientes que estén dispuestos a comprar a una compañía en específico. Hay muchas tácticas comerciales que intentan asegurar clientes: aumentar calidad del producto, promociones, disminuir costos, mejor mercadeo, etc. Sin embargo, todas las empresas están tomando este tipo de aproximaciones, intentando expandir su clientela. Además, las aproximaciones que se relacionan con la manufactura del producto son más difíciles de aplicar para las PYMEs, ya que, a diferencia de las empresas de mayor tamaño, la mayoría no tienen los recursos necesarios para la investigación y el desarrollo del producto, servicios y procesos de producción [(3) (4)].

Las PYMEs que se enfocan en el mercado comercial y de servicios tienen que mantener un contacto continuo con su clientela. Es importante para éstas planear cuándo es que se reúnen con sus clientes, o contactos que podrían ser clientes potenciales. Es útil para la PYME el construir una cartera de clientes y sobre todo poder crear en ellos la sensación de lealtad.

Para las PYMES que necesitan tener una comunicación constante empresa-cliente es necesario desarrollar estrategias mas centradas hacia el cliente. Como parte de estas estrategias se podría optar por la utilización de CRM (Customer Relationship Management) que es una forma en la que las empresas pueden comunicarse con el cliente[(3)]. CRM es una estrategia de negocios que busca

crear relaciones de largo plazo con los clientes y generar un valor económico[4]. Con el fin de aplicar ésta estrategia se podría optar por comenzar con la implementación de un Sistema de Gestión de Llamadas y Citas con Clientes (SGCall) que permita controlar la información de los contactos, prospectos y clientes que tiene una empresa PYME y las citas que se agendan y realizan con ellos.

I.3 Objetivos del Proyecto

A continuación se hace una breve descripción de los objetivos que se alcanzarán al finalizar el proyecto.

I.3.1 Objetivos Generales

Desarrollar un Prototipo de Sistema de Gestión de Llamadas y Citas con Clientes que funcione para cualquier PYME.

I.3.2 Objetivos Específicos

El sistema debe contar con las siguientes características específicas:

- Gestión de los datos referentes a los contactos, prospectos y clientes de la empresa.
- Gestión de los datos referentes a los agentes de ventas de la empresa.
- Permite agendar citas entre contactos, prospectos y/o clientes y los agentes de ventas.
- Llevar el control de los correos enviados o llamadas realizadas a los contactos. Así mismo si estos contactos han respondido favorablemente agendando una cita.
- Presentación de reportes acerca del calendario de citas de un determinado agente y de los resultados que ha obtenido en una determinada visita.

I.4 Metodología

La Metodología que se seguirá será el Análisis y Diseño Estructurado que cumplirá con las necesidades que se tienen para este proyecto y se implementará el prototipo en un Sistema Gestor de Bases de Datos Relacional.

Los pasos que se seguirán para el desarrollo del proyecto serán los siguientes:

- **Planteamiento del Problema.** Para poder asegurar que el sistema funciona como es requerido es necesario entender correctamente qué es lo que se desea realizar.

- **Revisión Bibliográfica.** Es muy importante que antes de realizar cualquier proyecto se haya hecho la investigación prudente de herramientas, términos, lenguajes y cualquier otra información que concierne al contexto en el que se implantará el sistema.
- **Especificación de Requisitos.** Una de las partes más importantes de la ingeniería de software para cualquier proyecto es el levantado correcto de requisitos con los que debe cumplir el sistema. La importancia tiene que ver mucho con que, si no se ha entendido correctamente qué se necesita del sistema, éste no realizará lo que el cliente espera.
- La especificación del **Modelo Conceptual de la Base de Datos, Modelo Funcional del Sistema y del Modelo de Comportamiento del Sistema** permitirá plasmar mediante el análisis cómo es que se representarán, almacenarán y fluirán los datos a través del sistema. Además, se describirá también como es que el sistema se comportará, es decir, sus cambios de estados y reacciones a sucesos dados.
- La elaboración de los diseños serán muy útiles para la correcta codificación del sistema ya que estos se acercan más a la manera en que se debe desarrollar el sistema.
    - **Diseño de la Base de Datos.**
    - **Diseño de la Arquitectura del Sistema.**
    - **Diseño de la Interfaz.**
    - **Diseño Procedimental.**
- **Codificación.** Esta será la parte del proyecto en la cual será necesario crear el código a partir del cual funcionará el sistema. Esta parte del proyecto es altamente dependiente de los pasos anteriores de la metodología.

I.5 Resultados

Para el desarrollo del sistema desarrollado en este proyecto fue necesaria la realización de varios procedimientos.

Inicialmente fue necesario recabar los requisitos del sistema. A partir de estos requerimientos se pudieron definir las funciones con las que debía cumplir el sistema y con los datos que se necesitaban obtener y guardar.

Sabiendo qué información es la que se necesitaba guardar se pudo comenzar a diseñar la base de datos. Para poder llevar a cabo el correcto diseño de la BD, primero se delimitaron las entidades y sus atributos. Además, se definieron las relaciones entre estas entidades. Habiendo especificado las entidades, atributos y relaciones se prosiguió elaborar el diagrama Entidad-Relación.

Posteriormente se procedió a transformar el modelo conceptual al modelo lógico. Ya teniendo el modelo lógico, se intentó aplicar la normalización a todas las tablas de la base de datos.

Para construir el sistema para acceder a la base de datos también fue necesario seguir varios pasos. Primero se definieron los casos de usos. También, se definió el diagrama de flujo de datos para el sistema.

Se determinó el mapa del sitio. Además, se definió la estructura básica para la página y las hojas de estilo que le darían el acabado buscado. Ésta estructura básica es igual para todas las páginas por lo que funciona como plantilla. Posteriormente se procedió a codificar cada uno de los casos de uso.

Se terminaron de construir todos los módulos para el sistema. Se realizaron pruebas básicas para determinar el correcto funcionamiento del sistema. Se probó cada uno de los módulos y se obtuvieron los resultados esperados.

CAPÍTULO II. **Marco de Referencia Teórico**

II.1 Modelado del Análisis Estructurado

II.1.1 Modelado de Datos

El modelado de datos es una herramienta que sirve para representar de manera abstracta alguna parte de la realidad. A este segmento de la realidad se le llama *universo de discusión*. Al modelar los datos estamos intentando describir y aislar las cualidades y relaciones que existen entre los elementos existentes en nuestro universo de discusión. También tiene como finalidad ayudar a describir la estructura que tendrá una base de datos [7] [8].

II.1.2 Modelo Conceptual de la Base de Datos

El modelo conceptual de la base de datos es un tipo de modelo cuyos conceptos son mucho más cercanos a la realidad y debido a esto está en un nivel más cercano al usuario. Usualmente utiliza conceptos como entidad, atributo y relación [7].

II.1.3 Diagrama Entidad Relación

Es un modelo de datos conceptual que es usado muy frecuentemente en el diseño conceptual de Bases de datos. Fue originalmente propuesto por Chen (1976,1977) y se describían los datos utilizando entidades, atributos, relaciones y dominio. El modelo intenta describir que atributos tiene cada entidad y la relación que existe entre cada entidad. Además, a través de este diagrama es posible delimitar la cardinalidad que hay entre una relación y las entidades que interactúan a través de ella [7].

II.1.3.1 Entidades

Una entidad es cualquier objeto, ya sea real o abstracto, que existe en la realidad y del cual deseamos recabar información. Al plasmar estos objetos de la realidad en nuestro modelo lo que intentamos es describir cuáles son sus características y propiedades. En el caso de las Bases de Datos existen también entidades débiles, las cuales no pueden existir por sí mismas, sino que necesitan la existencia de otra entidad para poder existir. Las entidades regulares son aquellas que tienen existencia por si mismas [7]. Cada entidad se describe mediante atributos, que son las propiedades específicas de la entidad [8].

II.1.3.2 Atributos

Un atributo es una característica o cualidad que es de importancia y que describe en cierta forma a una entidad [7]. Por ejemplo atributos de un globo serian color, forma, tamaño, etc.

Según el modelo ER (Entidad Relación) se tienen diversos tipos de atributos: **atributos compuestos, simples o atómicos, monovaluados, multivaluados, almacenados y derivados**. Los **atributos compuestos** son aquellos que se pueden dividir en atributos más básicos que tienen significado propio. Estos atributos compuestos se pueden jerarquizar, donde un atributo es subconjunto de otro. Sin embargo, si solo se hace referencia al atributo compuesto como un todo no hay necesidad de dividirlo. Los **atributos simples o atómicos** son aquellos que ya no pueden ser descompuestos en otros más básicos. Los atributos que tienen un solo valor para una entidad específica se denominan **atributos monovaluados**. Un atributo que puede tener diversos valores para una misma entidad se denomina **atributo multivaluado**. Los **atributos almacenados** son aquellos que solo se guardan como tal en la base de datos. Existen atributos que se determinan a partir de otros atributos, a estos se les llama **atributos derivados**.

Además de los atributos anteriormente mencionados existen otros. Hay atributos que tienen un valor diferente para cada entidad individual y se denominan **atributos clave** ya que pueden ayudar a identificar de manera única a una entidad [8].

II.1.3.3 Relaciones

Una relación describe la manera en que interactúan las entidades. Tomando el ejemplo anterior, si tuviéramos dos entidades: globo y persona, tendríamos que una posible relación que tendrían las dos sería: persona **compra** un globo. Las relaciones tienen un grado. El grado de una relación representa el número de entidades que participan en esta. La correspondencia expresa la cantidad de ejemplares de una entidad con el número de ejemplares de otra entidad que participan en la misma relación; usualmente se representa de la forma: 1:1, 1:N, M:N, etc. Las entidades tienen un papel o rol, que describe la función de estas en la relación. La cardinalidad refleja el número mínimo y máximo de ejemplares de una entidad que se relacionan con el número de ejemplares de los otros miembros de la relación. La cardinalidad usualmente se representa como: (0,1), (1,1), (0, n), (1, n), etc.

Similar a lo que sucede con las entidades, una relación puede ser regular o débil. Es regular cuando relaciona dos entidades regulares. Es débil cuando relaciona una entidad débil con una regular, o dos entidades débiles. Las entidades débiles pueden ser dependientes en existencia o dependientes en identificación. Existe dependencia en existencia cuando los ejemplares de una entidad no pueden existir sin los ejemplares de los que dependen. La dependencia en identificación exige que haya dependencia en existencia y que el ejemplar dependiente no pueda identificarse por sí mismo, necesitando el identificador del ejemplar de que depende [7].

II.1.4 Bases de Datos Relacionales

Una *base de datos (BD)* es un conjunto de datos que se relacionan entre sí, que se pueden registrar y que tienen un significado implícito. Además, estos datos tienen como finalidad reflejar objetos que existen en la realidad o universo de discurso. Las bases de datos que más se usan en la

actualidad son las Bases de Datos Relacionales. Este tipo de bases utilizan tablas que se relacionan unas con otras por medio de un campo (atributo) en común [8].

II.1.5 Modelado Funcional

Los datos que circulan a través de un sistema cambian constantemente y el propósito del Modelado funcional es tratar de reflejar cómo es que suceden estos cambios, que procesos se encargan de esto, que peticiones hacen las entidades externas y como se mueven los flujos de datos y de control [5].

II.1.5.1 Diagrama de Descomposición Funcional

El diagrama de descomposición funcional (Functional Decomposition Diagram) muestra de arriba abajo la estructura funcional de un sistema. En el diagrama se descomponen las funciones en sub-funciones, que a su vez se vuelven a descomponer en otras sub-funciones. Este tipo de diagramas ayuda a la creación de los diagramas de flujo de datos[6].

II.1.5.2 Modelo de Flujo de Datos

El modelo de flujo de datos intenta describir cómo es que los datos fluyen y se almacenan en el sistema. Trata de establecer cómo es que esta información fluye entre los procesos y como esta es transformada por estos en salidas [5].

II.1.5.2.1 Diagramas de Flujo de Datos (DFD)

Permite visualizar al sistema como una red de procesos funcionales conectados entre sí por flujos y almacenes de datos. Son muy utilizadas, sobre todo en sistemas que tienen gran importancia en sus funciones y que generalmente son muy complejos. Usualmente un Diagrama de Flujo de datos está compuesto por procesos, flujos, almacenes y terminadores.

El proceso describe una parte del sistema que toma flujos de entrada y los convierte en algún tipo de salida. Usualmente se representa con un óvalo o un rectángulo con esquinas redondeadas. Usualmente el nombre del proceso describe en cierta manera cual es su función. A veces es posible que el proceso describa a quien lo usa y no lo que hace.

El flujo de datos, usualmente representado por una flecha que entra o sale de un proceso, describe como fluyen los datos de una parte del sistema a otros. Además, debido a la forma de flecha también describen en qué dirección se mueve el flujo.

El almacén es una colección de datos que se encuentran en reposo. Usualmente uno podría considerar que ese almacén puede referirse a algún archivo o una base de datos.

Los *terminadores* representan entidades externas con las que el sistema interactúa. Usualmente los terminadores son personas, grupos e incluso algún otro sistema que necesitan realizar una acción sobre el sistema [9] [11].

II.1.5.3 Modelo de Flujo de Control

Existen diversos sistemas que no solo están ligados a la forma en que los datos fluyen a través del sistema, sino que también es necesario y que están dirigidas por sucesos y que tienen restricciones de tiempo y de rendimiento[7].

II.1.5.3.1 Diagrama de Flujo de Control (DFC)

Un Diagrama de Flujo de Control contiene los mismos procesos que se han ocupado en el DFD pero en lugar de describir en qué manera fluyen los datos se representan los flujos de control. Para este diagrama se usa una referencia de anotación (Barra sólida, también considerada como una ventana) a una *especificación de control*[7].

II.1.5.3.2 La Especificación de Control (EC)

Se utiliza la *especificación de control* para indicar como es que responde el sistema a un suceso o a un flujo de control y para saber que procesos son invocados cuando se ocurre el suceso. Aunque el EC nos describe hasta cierto punto el comportamiento del sistema pero no la manera en que los procesos actúan de manera interna debido al comportamiento del sistema.

La EC contiene un diagrama de transición de estados y a veces una tabla de activación de procesos [7].

II.1.5.3.3 Especificación del Proceso (EP)

La Especificación del Proceso sirve para describir los procesos que se muestran en el nivel más refinado del DFD. La especificación de software se puede hacer textualmente, mediante un algoritmo escrito en LDP (Lenguaje de Diseño de Programas), ecuaciones matemáticas, tablas, diagramas o gráficos. La EP junto con el DFD permite obtener un mejor entendimiento de los requerimientos de software que tiene el sistema[7].

II.1.6 Modelado de Comportamiento

El *Modelado de Comportamiento* es uno de los principales fundamentos del análisis de requisitos. Y describe cómo es que el sistema se comporta y cambian los estados dentro del mismo[7].

II.1.6.1 Diagramas de Transición de Estado (DTE)

Un Diagrama de Transición de Estados describe gráficamente los estados que existen en el sistema. Además, busca representar como es que ocurren los cambios de estado e indica que acciones son necesarias para que ocurra un suceso dado.

Un *estado* se dibuja como un rectángulo y representa un modo observable de comportamiento.

Los Cambios de Estados se representan con flechas que unen dos estados.

También por medio de estos diagramas es posible representar qué *condición* es necesario cumplir para que el sistema pase de un estado a otro. También se muestra que *acción* tomará el sistema cuando se cumpla la condición que se estableció para que se realizara el cambio de estado.

Utilizando los estados y sus cambios somos capaces de saber a qué estados se pueden ir a partir del estado actual así como tener una idea de cómo llegar al estado actual [(9) (11)].

II.2 Diseño Lógico de la Base de Datos

El Diseño Lógico de la Base de Datos supone la transformación del Modelo Conceptual de la base de datos en otro modelo que cumpla con exigencias como baja redundancia, simplicidad y que obtenga un balance entre la exigencia de los usuarios y eficiencia [(7)].

II.2.1 Transformación del Modelo Conceptual de la Base de Datos al Modelo Lógico

Según De Miguel Castaño(2000) existen tres reglas básicas que hay que seguir para la transformación del Modelo Conceptual al modelo lógico:

1) **Toda entidad se convierte en una tabla.** Cada atributo existente en la entidad se convertirá en una columna de la tabla.
    - Los atributos identificadores pasan a ser llaves primarias de la tabla.
    - Los identificadores alternativos serán catalogados como únicos y si se desea pueden ser nulos.
    - Los atributos no identificadores se convierten directamente en columnas de la tabla y pueden tomar valores nulos mientras no se indique de otra forma.
2) **Toda interrelación N:M se convierte en una tabla.** Esta nueva tabla tendrá como llave primaria la concatenación de las llaves primarias que se relacionaban por medio de esta relación. En el caso de que en la relación hubiese un atributo, este se convertirá en una columna de la tabla nueva.
3) **Para toda relación 1:N se realiza una propagación de clave o simplemente se crea una nueva tabla.** Si se realiza la propagación *de* claves, la clave de la entidad con cardinalidad máxima 1 se convertirá en una llave foránea para la entidad que tiene cardinalidad *n*. Si en su caso existieran atributos para la relación estos se podría convertir en atributos de la entidad con cardinalidad *n*. Sin embargo, es posible que se necesite crear una nueva tabla si se considera que esto será mejor que solo propagar atributos.

También hay que considerar otros aspectos a la hora de realizar la transformación:

- En el caso de la herencia podríamos optar por tomar tres posibles opciones para la transformación:
    - **Englobar todos los atributos de la entidad y sus subtipos en una sola tabla.** Esta aproximación usualmente se tomará cuando la entidad y los subtipos tienen muy pocas diferencias entre sí.
    - **Crear una tabla para el supertipo y para cada subtipo.** Es mejor usar esta aproximación cuando el supertipo y sus subtipos tienen muchas diferencias con respecto a los atributos que les conforman.

- **Crear tablas para cada subtipo, que contengan además de los atributos propios los comunes.**

II.2.2 Normalización

El objetivo primordial del proceso de normalización es el que no ocurran problemas de redundancia o de actualización en la base de datos. Según Codd la normalización consiste en someter a un esquema de relación a una serie de pruebas con las que se busca cerciorarse de que el esquema pertenece o no a una forma normal especifica. Codd propuso las tres primeras formas normales. Posteriormente la tercera forma normal fue modificada para ser más estricta convirtiéndose así en la forma normal de Boyce-Codd. Aquellos esquemas relacionales que no cumplan con éstas especificaciones serán descompuestos en otros más simples y que cumplan con ciertas características. Algunas de las características con las que los esquemas relacionales deben cumplir son:

Garantizar que no existan tuplas erróneas que representan información errónea o agregada que no es válida.

Asegurar que todas las dependencias funcionales han sido representadas en alguna relación individual.

La **Primera Forma Normal (1fn)** establece que el valor de cualquier atributo debe ser único y debe provenir de su dominio. Debido a esto un atributo multivaluado deberá ser mapeado a una nueva tabla que guardará relación con la entidad original.

La **Segunda Forma Normal (2fn)** dice que una relación (R) está en 2fn si esta en 1fn y todo atributo no primo *depende funcionalmente de manera total* de la llave primaria de la relación.

La **Tercera Forma Normal(3fn)** dicta que una relación tiene que estar en 2fn y ningún atributo no principal depende algún otro atributo no funcional.

La **Forma Normal de Boyce-Codd (FNBC)** establece que toda relación esta en FNBC si y solo si todo determinante es una clave candidata [(7) (8)].

II.3 Lenguajes de Definición y Manipulación de Datos

Un Lenguaje de Definición de Datos (DDL por sus siglas en Inglés) permite especificar los datos que integran la base de datos, su estructura y las relaciones que existen entre ellos. Permite la creación y modificación de relaciones, vistas, disparadores, índices, etc. El Lenguaje de Manipulación de Datos (DML por sus siglas en Inglés) permite añadir, eliminar y modificar los datos de la base de datos. Además, permite hacer selecciones de datos que cumplan con algún requisito dado [(8)].

II.3.1 SQL

SQL (Structured Query Languaje) es uno de los lenguajes de bases de datos más utilizados. SQL nos permite definir y manipular los datos ya que está compuesto por: DDL, DML, DML inmerso o incorporado, Control de Transacciones e Integridad[(8)].

II.4 Sistema Gestor de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es un software que permite almacenar datos y acceder a los mismos. Un SGBD permite definir y manipular los datos, darle mantenimiento de integridad a la BD y controlar la privacidad y seguridad de la BD[(9)].

II.4.1 MySQL

MySQL es un Sistema Gestor de Bases de Datos Relacional (SGDBR). Permite almacenar grandes cantidades de datos, administrar el sistema, dar diversos niveles de acceso a usuarios, proteger y hacer volcados de datos. Además, permite crear aplicaciones de bases de datos propios utilizando la mayoría de los lenguajes de programación. MySQL también es muy útil para el manejo de bases datos por medio de páginas Web Dinámicas[(10)].

II.5 Lenguajes para el Desarrollo de Páginas Web Dinámicas

II.5.1 HTML

Los navegadores web son programas diseñados para desplegar páginas web. Estas páginas están escritas en un lenguaje llamado HTML (HyperText Markup Languaje) . Mediante un conjunto de etiquetas (código adicional) es posible estructurar el contenido de la página. Para que el navegador web pueda aplicar un estilo a los elementos etiquetados mediante HTML, se utilizan Hojas de Estilo en Cascada (CSS). De esta manera el navegador primero estructura la página con el código HTML y después usando las reglas CSS da un estilo a los elementos de la misma[(11)].

II.5.2 CSS

Las Hojas de Estilo en Cascada (Cascading Style Sheets) es un lenguaje de programación que permite especificar la apariencia que tendrá una página Web. A diferencia de HTML que define la estructura que tendrá la página, CSS le dice al navegador exactamente que apariencia tendrá la página. A través de CSS se puede modificar el aspecto que tendrán las etiquetas en HTML y seleccionar de diferentes maneras exactamente qué etiquetas cambiarán su apariencia[(12)] .

II.5.3 PHP

Las páginas escritas en PHP son páginas escritas en HTML que además del código HTML tienen el programa PHP que se desea ejecutar. Cuando un cliente solicita la página, esta se pre procesa ejecutando el código PHP. Debido a esto es posible generar código HTML dinámicamente al cargar la página, al insertar etiquetas HTML en la página.

Otra de las virtudes de PHP es que es multiplataforma y de código abierto. PHP puede ser ejecutado en diversos sistemas operativos. Aunado a esto, PHP tiene gran flexibilidad al albergar diferentes tipos de extensiones que le permiten incrementar sus capacidades[(13)].

II.6 Servidores Web (Software)

Apache

Apache es un servidor multiplataforma y trabaja en varios sistemas operativos. Apache tiene un elaborado índice de directorios, un directorio de alias, negociación de contenidos, informe de errores HTTP y varias otras capacidades. Además, Apache es capaz de funcionar en plataformas virtuales[(14)].

CAPÍTULO III. **Especificación de los Requisitos del Sistema**

III.1 Descripción General del Sistema

En este proyecto se pretende desarrollar una parte de un CRM Básico denominado Sistema de Gestión de Llamadas y Citas con Clientes (SGCall) que permita controlar la información de los contactos, prospectos y clientes que tiene una empresa PYME y las citas que se agendan y realizan con ellos. El sistema se implementará en un SGBD Relacional.

III.2 Objetivos

El objetivo del sistema desarrollado en este proyecto es permitir a la PYME poder llevar a cabo una mejor planeación de las llamadas, correos y citas que se deben de llevar a cabo con el cliente para permitir una relación continua y productiva con el mismo.

III.3 Alcance

El sistema deberá:

- Gestionar los datos referentes a los agentes de ventas, contactos, prospectos y clientes de la empresa.
- Permitir agendar citas entre contactos, prospectos y/o clientes y los agentes de ventas. Además se deberá tomar en cuenta que las citas pueden ser reprogramadas.
- Llevar el control de los correos enviados o llamadas realizadas a los contactos. Así mismo si estos contactos han respondido favorablemente agendando una cita.
- Se deberán presentar reportes acerca del calendario de citas de un determinado agente y de los resultados que ha obtenido en una determinada visita.
- Limitar el acceso al sistema por medio de un USUARIO y CONTRASEÑA.

III.4 Casos de Uso

Estos escenarios identifican una línea de utilización para el sistema que va a ser construido. La correcta definición de estos escenarios permitirá describir lo que podrá realizar el sistema a partir de todos sus escenarios posibles.

III.4.1 Actores

"Un actor es algo que comunica con el sistema o producto y que es externo al sistema en sí mismo. Un actor representa una clase de entidades externas que lleva a cabo un papel" [(11)]. El actor es en concepto una entidad externa que interactúa en algún momento con el sistema.

Se distinguen para este sistema 3 tipos de Actores:

- **Responsable de Marketing:** Es el responsable del área de marketing. Este actor solo podrá acceder al sistema y visualizar los reportes.
- **Responsable de Ventas:** Es el responsable del área de ventas. Es el encargado de asignar contactos a los Agentes. Además, este actor será el equivalente a un Administrador y por lo tanto tendrá acceso a todas las funcionalidades que el sistemas ofrezca.
- **Agente de Ventas:** Un empleado de la empresa que visita a los prospectos o clientes para promover una venta. Este Actor podrá Acceder al Sistema, Agendar Citas, Gestionar Prospectos, Gestionar Clientes y Controlar Llamadas y Correos.

III.4.2 Documentación de los Casos de Uso

A continuación se presenta un listado de todos los casos de uso del sistema así como una breve descripción de lo que realiza cada caso de uso.

No.	Caso de Uso	Descripción
1	Gestión de Agentes	Creación, modificación y eliminación de los Agentes de Ventas que forman parte de la Empresa.
2	Gestión de Contactos	Creación, modificación y eliminación de los contactos cuya información está en la Base de Datos.
3	Gestión de Prospectos	Creación, modificación y eliminación de los prospectos cuya información está en la Base de Datos.
4	Gestión de Clientes	Creación, modificación y eliminación de los clientes cuya información está en la Base de Datos.
5	Agendar Cita	Agendar y modificar citas entre un agente de ventas y un prospecto, contacto o cliente.
6	Acceder al Sistema	Iniciar sesión en el sistema con el uso de un USUARIO y CONTRASEÑA
7	Control Llamadas y Correo	Se controlarán las llamadas y los correos entre los agentes de ventas y contactos. Y se indicará si los resultados fueron favorables.
8	Ver Reportes	Se podrá ver alguno de los reportes generados por el sistema

III.4.3 Reportes

A continuación se presenta un listado de los reportes que deberá generar el sistema.

No.	Reporte	Descripción
1	Calendario	Se deberá desplegar un calendario de citas de un determinado agente.
2	Resultados	Se deberá desplegar un reporte de los resultados obtenidos por el Agente de ventas en una determinada cita.

III.4.4 Modelo de Casos de Uso

La Ilustración 1 ejemplifica gráficamente la manera en que quedarán distribuidos los casos de Uso.

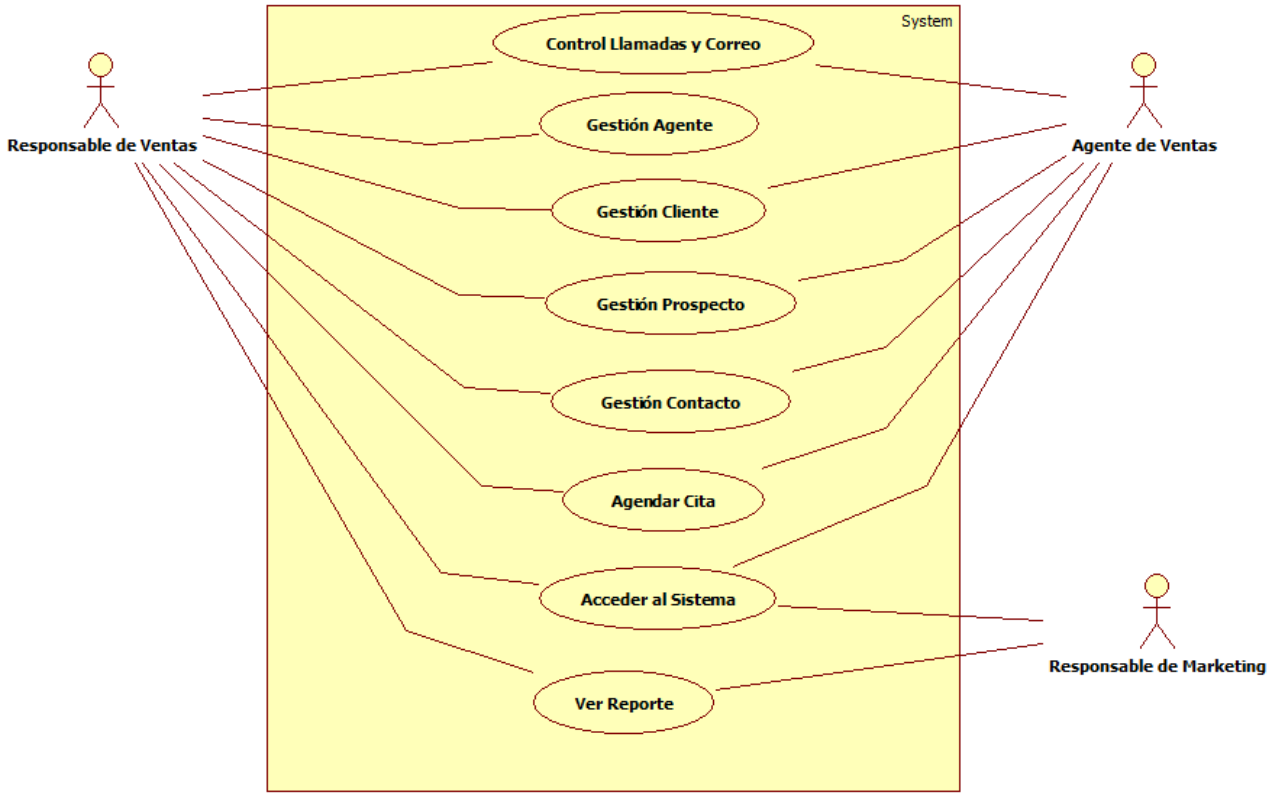

Ilustración 1. Diagrama de Casos de Uso

En la ilustración se muestra el diagrama de casos de uso. Este diagrama trata de describir que actores interactúan con determinados casos de uso.

III.5 Requerimientos Funcionales

Estos requerimientos nos describen qué funciones deberá realizar el sistema.

A continuación se presenta un listado y una breve descripción de los requerimientos que se tienen para el sistema desarrollado:

RF-1. Gestionar a Agente de ventas

Se deberá ingresar la información básica de los agentes de ventas, editar los datos y eliminar Agentes del sistema.

RF-2. Gestionar Contacto

Se deberá ingresar la información básica de los contactos, editar los datos y eliminar contactos del sistema.

RF-3. Gestionar Cliente

Se deberá ingresar la información básica de los clientes, editar los datos y eliminar clientes del sistema.

RF-4. Gestionar Prospecto

Se deberá ingresar la información básica de los prospectos, editar los datos y eliminar prospectos del sistema.

RF-5. Calendarizar Cita

Se deberá agendar una cita entre un agente de ventas y algún prospecto, contacto o cliente.

RF-6. Reportar Resultados

Se deberán reportar los resultados obtenidos de una cita, correo o llamada.

RF-7. Iniciar Sesión

Se deberá acceder al sistema ingresando un USUARIO y CONTRASEÑA.

RF-8. Ver Calendario

Se deberá mostrar el calendario de citas agendadas para un determinado Agente de Ventas.

RF-9. Ver Resultado

Se deberá mostrar un reporte en el que se muestren los resultados de una determinada cita entre un agente y algún prospecto, contacto o cliente.

III.6 Requerimientos No Funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema y que usualmente incluyen restricciones de tiempo, proceso de desarrollo y estándares[(15)].

Para el caso del Sistema SGCall tenemos como requerimiento no funcional que el sistema deberá ser accesible a partir de un navegador web.

CAPÍTULO IV. **Análisis y Diseño del Sistema**

IV.1 Análisis del Sistema

IV.1.1 Modelo Conceptual de la Base de Datos del Sistema

IV.1.1.1 Entidades, Atributos, Dominios y Relaciones

En el Capítulo II secciones 1.3.1, 1.3.2 y 1.3.3 se definieron los conceptos de entidad, atributo y relación. A continuación se hará un listado y una breve descripción de las entidades, atributos y relaciones que se modelaron para el sistema. Primero se hará una breve descripción de la entidad y posteriormente se enlistarán sus atributos. Si en el nombre de la entidad hay un número entre paréntesis esto indica que la entidad trabaja bajo alguno de los supuestos descritos en la sección 1.1.1.3 del capítulo 4.

IV.1.1.1.1 Entidades

Contacto: Algún consumidor que ha contactado a la empresa. Un contacto tendrá los siguientes atributos:

- **ID**
- **Nombre**
- **e-mail**
- **Teléfono**

Prospecto [1]: Algún consumidor que ha contactado a la empresa, que se le ha agendado una cita o se ha realizado alguna visita o se ha tenido una entrevista con él y que se ha interesado por algún producto o se le ha proporcionado una cotización. Un prospecto tendrá los siguientes atributos:

- **ID**
- **Nombre**
- **e-mail**
- **Teléfono**
- **Dirección**
- **Colonia**
- **Ciudad**
- **Municipio**
- **Estado**

Cliente [2]: Un cliente es quien ya ha comprado algún producto o contratado algún servicio y se cuenta adicionalmente con su dirección, colonia, ciudad, municipio, importe de sus compras. Un cliente tendrá los siguientes atributos:

- **Dirección**
- **Colonia**
- **Ciudad**
- **Municipio**
- **Importe de sus compras**

Agente de Ventas: Un empleado de la empresa que visita a los prospectos o clientes para promover una venta. Un agente de ventas tendrá los siguientes atributos:

- **ID**
- **Nombre**
- **RFC**

Cita [3]: Encuentro programado entre un contacto, prospecto o cliente y un agente de ventas. Una cita tendrá los siguientes atributos:

- **ID Cita**
- **Asunto**
- **Fecha**
- **Hora**
- **Lugar**
- **Resultado de la cita**

Empresa: Es una organización a la cual un contacto, prospecto o cliente pueden pertenecer. Una empresa tendrá los siguientes atributos:

- **Id Empresa**
- **Nombre**
- **Razón Social**
- **Mail**
- **Teléfonos**
- **Dirección**
- **Colonia**
- **Ciudad**
- **Municipio**
- **Estado**
- **Giro**
- **No. de empleados**
- **Ventas anuales**
- **Tamaño**

Correo/Llamada [4]: Un correo o Llamada tendrá los siguientes atributos:

- **Fecha**
- **Hora**
- **Asunto**
- **Resultado**

IV.1.1.1.2 Relaciones

A continuación se hace una breve descripción de las relaciones que existen entre las entidades existentes en la base de datos.

Agente de Ventas- Cita: Un agente de ventas debe participar en una cita.

Agente de Ventas- Llamada/Correo: Un agente de ventas debe participar en una llamada o correo.

Contacto-Cita: Un contacto puede participar en una cita. En una cita debe participar un contacto, un prospecto o un cliente.

Contacto – Llamada/Correo: Un contacto debe participar en una llamada.

Prospecto-Cita: un prospecto puede participar en una cita. Sin embargo, en una cita debe participar un contacto, un prospecto o un cliente.

Prospecto – Empresa: Un prospecto puede pertenecer a una o varias empresas. Así mismo una empresa puede tener uno o varios prospectos.

Cliente-Cita: Un cliente puede participar en una cita. Sin embargo, en una cita debe participar un contacto, un prospecto o un cliente.

Cliente – Empresa: Un cliente puede pertenecer a una o varias empresas. Así mismo una empresa puede tener uno o varios prospectos.

IV.1.1.1.3 Supuestos

En esta sección se describen los supuestos bajo los cuales trabaja la base de datos.

[1] - Se trabaja bajo la suposición de que un contacto se puede interesar en un producto convirtiéndose entonces en un Prospecto.

[2] - Se ha asumido que un cliente es un prospecto que en su momento ha realizado alguna compra o contratación, por lo que podría heredar los atributos de Prospecto.

[3] - Las citas pueden ser reprogramadas.

[4] - El correo y la llamada comparten los mismos atributos. Debido a esto, es posible agruparlos en una misma entidad y solo tener un atributo que nos diga si es correo o llamada.

IV.1.1.2 Diagrama entidad-relación del Sistema

Como vimos en la sección 1.3 del capítulo II el modelo Entidad-Relación describe de una manera abstracta algún problema del mundo real que se quiere modelar.

Al analizar la información concerniente a los datos que nos interesa guardar en el sistema se modeló la base de datos como se muestra en la Ilustración 2.

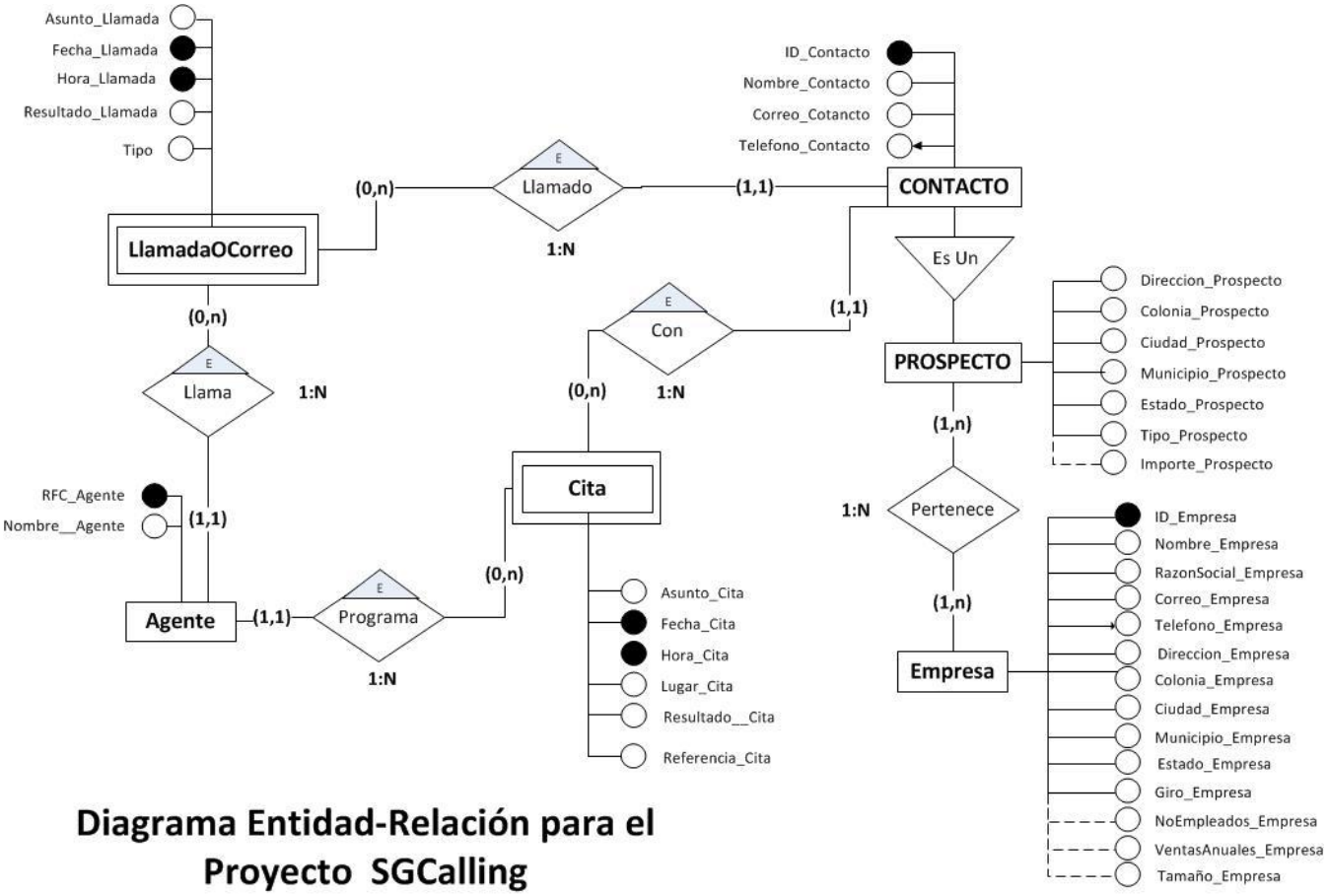

Ilustración 2. Diagrama Entidad-Relación

En la ilustración se muestra el diagrama Entidad-Relación. Este diagrama muestra como se modeló originalmente la base de datos para el sistema.

IV.1.2 Modelo Funcional y Diagramas de Flujo de Datos (DFD's)

Como se vio en el capítulo II sección 1.5.2.1 se definió el Diagrama de Flujo de Datos (DFD). El DFD permite visualizar el sistema como una red de procesos interconectados entre sí por flujos y almacenes de datos.

Los DFDs del sistema SGcall serán mostrados en esta sección.

La Ilustración 3 muestra el DFD nivel 0, mejor conocido como diagrama de contexto.

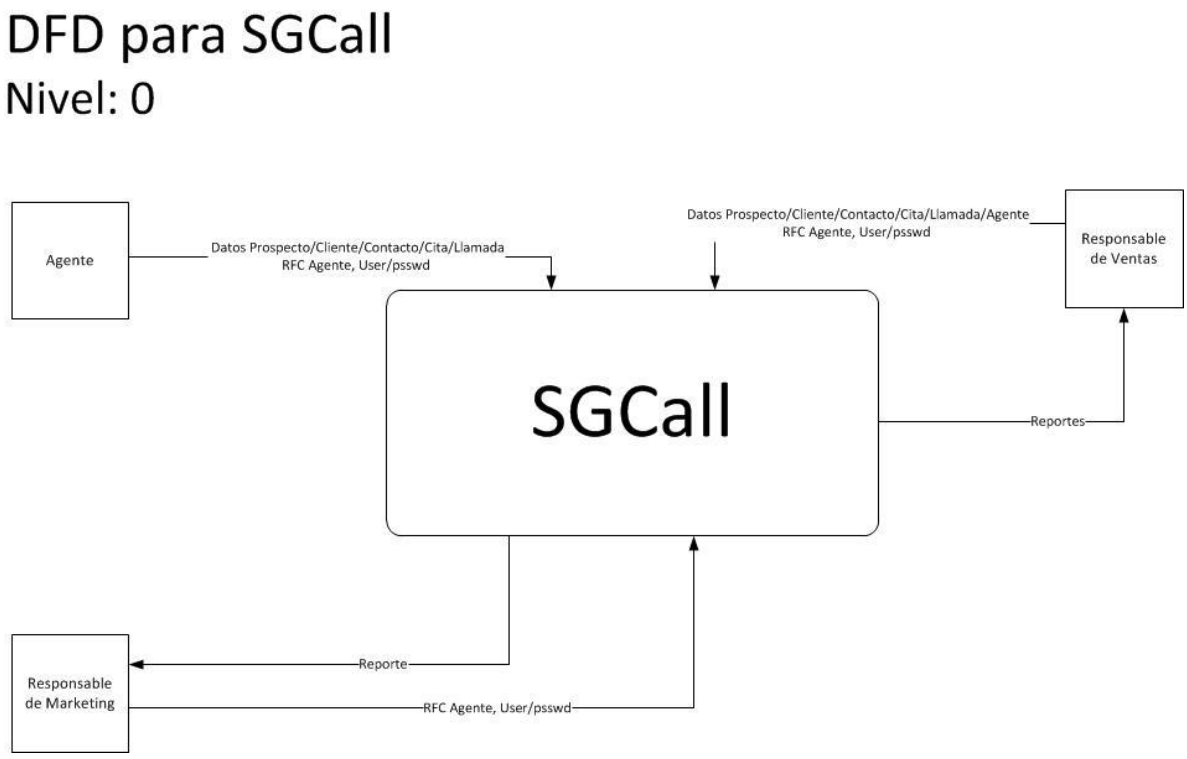

Ilustración 3. Diagrama de Flujo de Datos de Nivel 0

En la ilustración se muestra el Diagrama de Flujo de Datos de nivel 0 para el sistema.

La Ilustración 4 muestra el DFD de nivel 1 para el sistema SGCall.

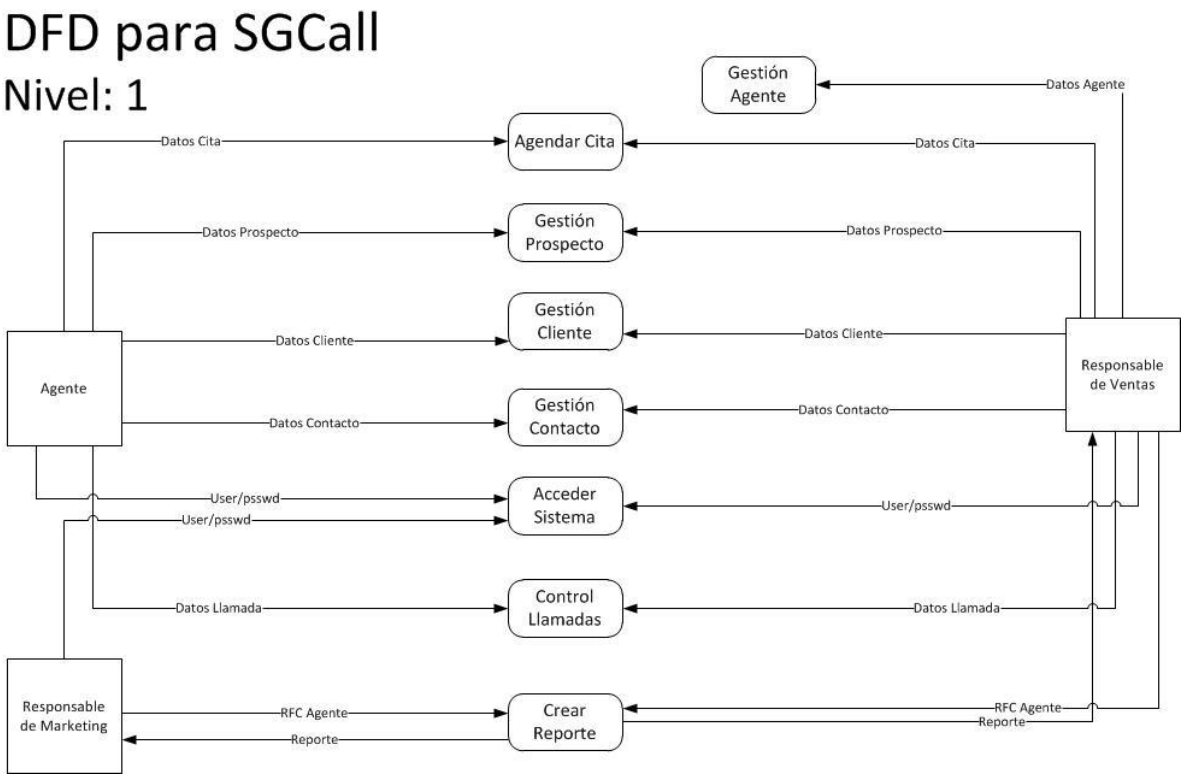

Ilustración 4. Diagrama de Flujo de Datos de Nivel 1
La ilustración muestra el DFD de nivel 1 del sistema.

IV.2 Diseño Lógico.

IV.2.1 Esquema Relacional de la Base de Datos.

En la sección 2.1 del Capítulo II se definieron varias reglas para convertir nuestro modelo Conceptual al Modelo Lógico. Para elaborar el Esquema relacional se transformó el modelo conceptual en el Modelo Lógico. La Ilustración 5 muestra el esquema relacional de la Base de Datos.

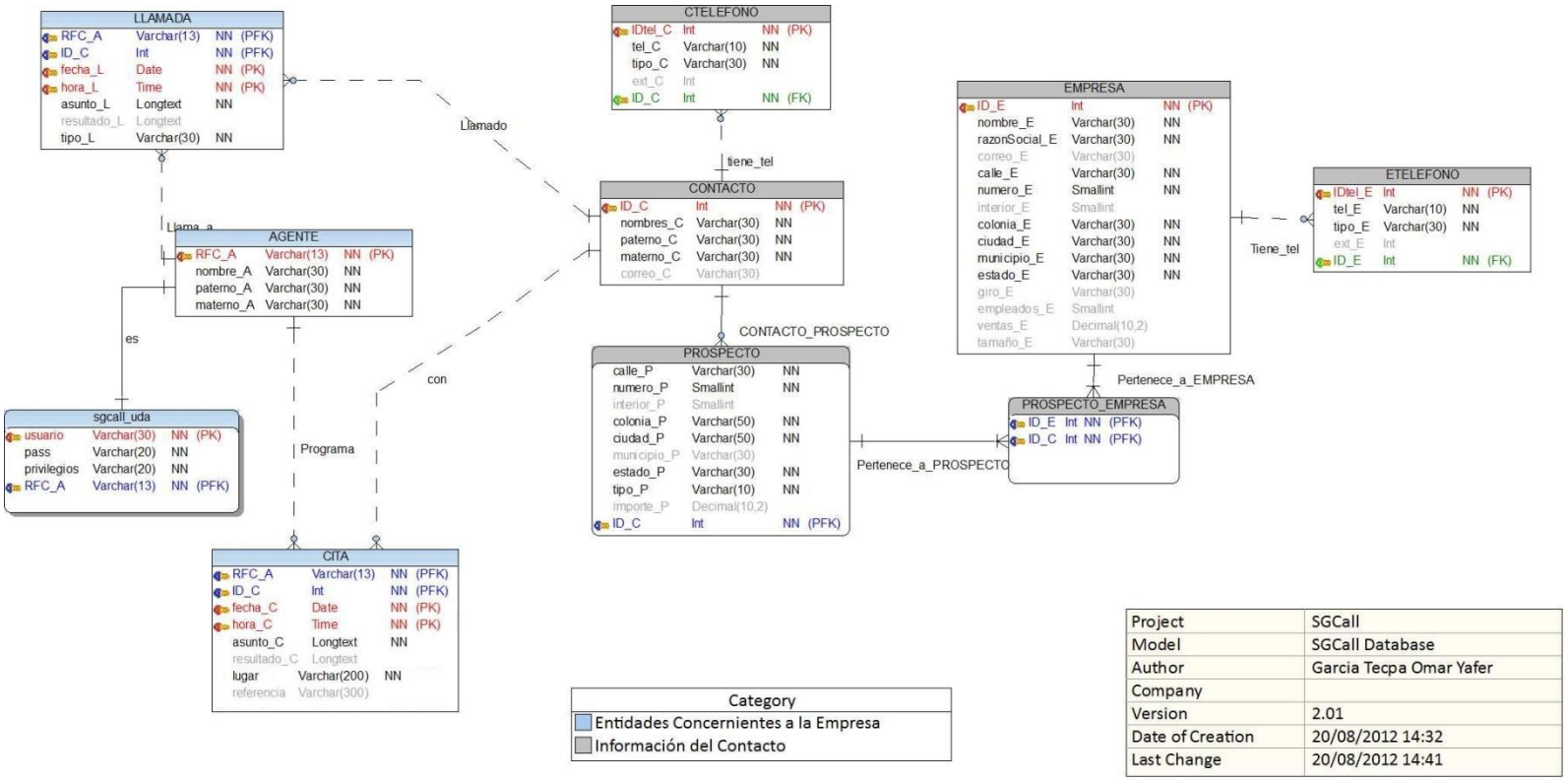

Ilustración 5. Esquema Relacional de la Base de Datos

En la ilustración se muestra el esquema Relacional de la Base de Datos. Este esquema corresponde a la transformación del Modelo Conceptual al Modelo Lógico.

IV.2.2 Normalización.

En la sección 2.2 del Capítulo II se discutieron las reglas de normalización que se utilizarán para llevar a cabo el proceso de normalización de la base de datos del sistema SGCall.

Si consideramos la siguiente afirmación:

"La Primera forma normal (1FN) es una restricción inherente al modelo relacional, por lo que su cumplimiento es obligatorio y afecta al número de valores que pueden tomar los valores de una relación". [7]

Entonces podemos aseverar que las tablas de la base de datos de SGCall están al menos en la primera forma normal.

Para efectuar el proceso de normalización tomaremos como ejemplo la tabla usuario.

Tabla Usuario

Primero definimos la tabla con su nombre, atributos, llaves foráneas y la llave primaria de la tabla.

> **usuario** (usuario, pass, privilegios, *RFC_A*)

Posteriormente procedemos a verificar las dependencias funcionales de los atributos con la llave y con los atributos no llave.

> usuario → pass, privilegios, RFC_A.

Finalmente explicamos la razón por la que está o no normalizada la tabla hasta la tercera forma normal.

> La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Además, no existe dependencia transitiva entre los atributos no claves por lo que la tabla está en **Tercera Forma Normal (3FN)**.

Ahora que hemos hecho el proceso normalización para la tabla usuario. Procederemos a hacer lo mismo para todas las tablas de la base de datos. Repetiremos el mismo proceso para el resto de las tablas.

Tabla Agente

agente (RFC_A, nombres_A, paterno_A, materno_A)

RFC_A → nombres_A, paterno_A, materno_A

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Contacto

contacto (ID_C, nombre_C, paterno_C, materno_C, correo_C)

ID_C → nombre_C, paterno_C, materno_C, correo_C

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Además, no existe dependencia transitiva entre los atributos no claves por lo que la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Ctelefono

Ctelefono (IDtel_C, tel_C, tipo_C, ID_C)

IDtel_C → tel_C, tipo_C, ID_C

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Llamada

llamada (fecha_L, hora_L, asunto_L, resultado_L, tipo_L, *ID_C, RFC_A*)

fecha_L, hora_L, ID_C, RFC_A → asunto_L, resultado_L, tipo_L

La tabla está en **Segunda Forma Normal (2FN)** ya que los atributos dependen de toda la llave y no solo de una parte de ella. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Cita

cita (fecha_C, hora_C, asunto_C, lugar_C, referencia_C, resultado_C, *RFC_A, ID_C*)

RFC_A, ID_C, fecha_C, hora_C→ asunto_C, lugar_C, resultado_C, referencia_C

Referencia_C → lugar_C

La tabla está en **Segunda Forma Normal (2FN)** ya que los atributos dependen de toda la llave y no solo de una parte de ella. Sin embargo existe una dependencia transitiva entre referencia y lugar. Por lo tanto se normalizará la tabla de la siguiente manera:

cita (fecha_C, hora_C, asunto_C, resultado_C, *RFC_A, ID_C, ID_lugar*)

RFC_A, ID_C, fecha_C, hora_C→ asunto_C, lugar_C, resultado_C

lugar(ID_lugar, lugar, referencia);

ID_lugar → lugar, referencia

NOTA: el atributo lugar se refiere a la dirección del lugar.

La tabla cita está en **Segunda Forma Normal (2FN)** ya que los atributos dependen de toda la llave y no solo de una parte de ella. La tabla lugar está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta.

Al no existir dependencias transitivas entre los atributos no claves ambas tablas están en **Tercera Forma Normal (3FN)**.

Tabla Prospecto

prospecto (ID_C, calle_P, numero_P, interior_P, colonia_P, ciudad_P, municipio_P, estado_P, tipo_P, importe_P)

ID_C → calle_P, numero_P, interior_P, colonia_P, ciudad_P, municipio_P, estado_P, tipo_P, importe_P

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Prospecto_Empresa

prospecto_empresa (ID_E, ID_C)

La tabla está en **Segunda Forma Normal (2FN)** y en **Tercera Forma Normal (3FN)** debido a que no existen otros atributos que no son llave primaria.

Tabla Empresa

Empresa(ID_E, nombre_E, razonSocial_E, correo_E, calle_E, numero_E, interior_E, colonia_E, ciudad_E, municipio_E, estado_E, giro_E, empleados_E, ventas_E)

ID_E → nombre_E, razonSocial_E, correo_E, calle_E, numero_E, interior_E, colonia_E, ciudad_E, municipio_E, estado_E, giro_E, empleados_E, ventas_E

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Tabla Etelefono

Ctelefono (IDtel_E, tel_E, tipo_E, ID_E)

IDtel_E → tel_E, tipo_E, ID_E

La tabla está en **Segunda Forma Normal (2FN)** debido a que existe una llave primaria única y no compuesta. Al no existir dependencias transitivas entre los atributos no claves la tabla está en **Tercera Forma Normal (3FN)**.

Una vez aplicada la normalización tendremos el esquema relacional que se muestra en la Ilustración 6.

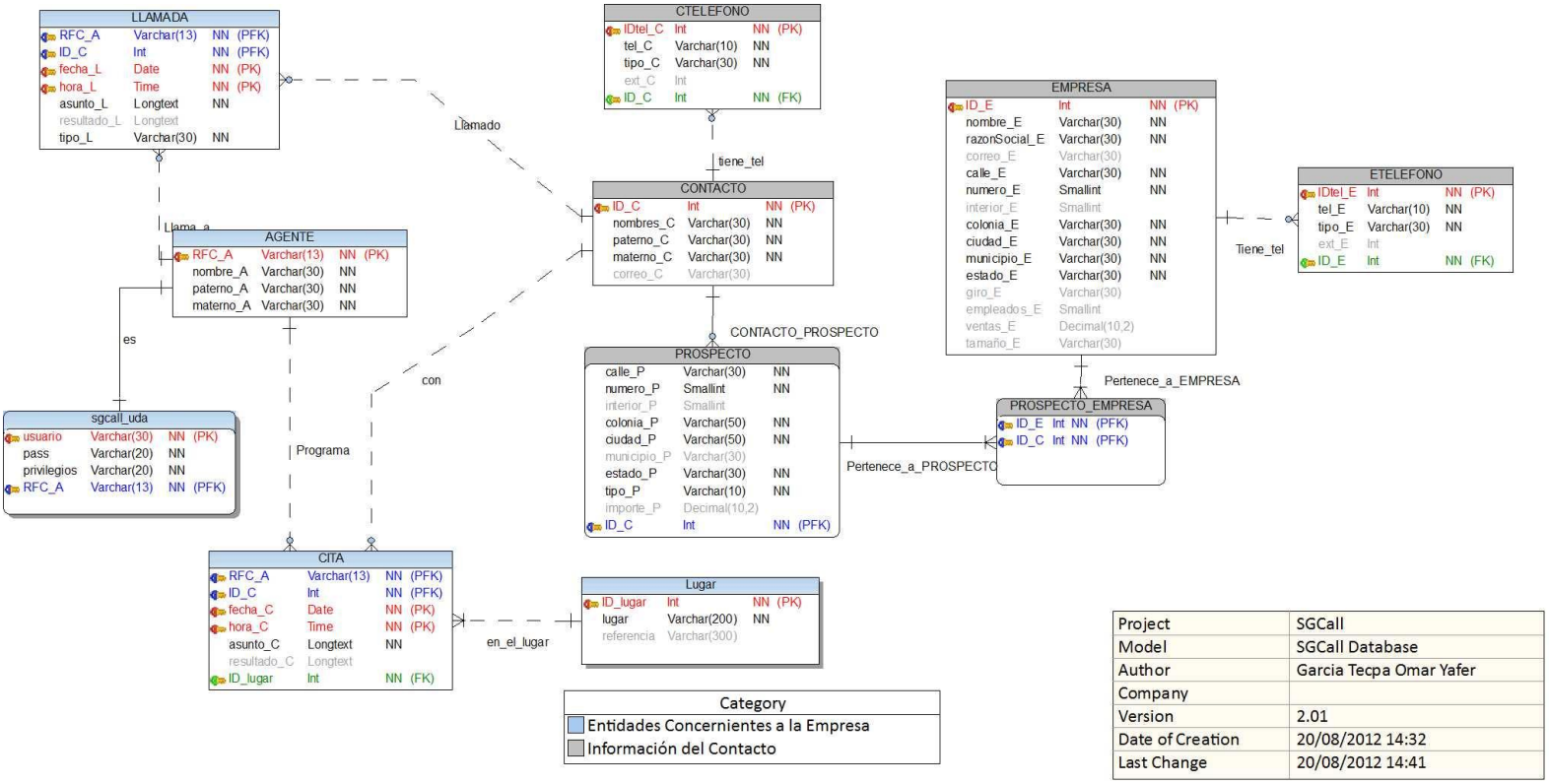

Ilustración 6. Esquema Relacional Normalizado

En la ilustración se muestra el Esquema Relacional resultante de normalizar la base de datos.

IV.3 Diseño de la Interfaz de Usuario.

En la Ilustración 7 se muestra el diseño básico de la interfaz. En la parte superior tenemos el logo y el nombre del sistema. Por debajo del logo tenemos un menú de navegación. A la derecha y debajo del menú tenemos la opción de cerrar sesión.

Para mostrar los diversos contenidos se tienen tabs. Que en la mayoría de los casos tienen las opciones: agregar, editar, Mostrar.

Ilustración 7. Diseño Básico de la Interfaz

En la ilustración se muestra el diseño básico de la interfaz que tendrá la aplicación.

CAPÍTULO V.
Implementación

V.1 Mapa del Sitio

En la Ilustración 8 se muestra el Mapa de Sitio del sistema. El mapa se construyo tomando en cuenta los casos de uso y las capacidades con las que cuenta el sistema.

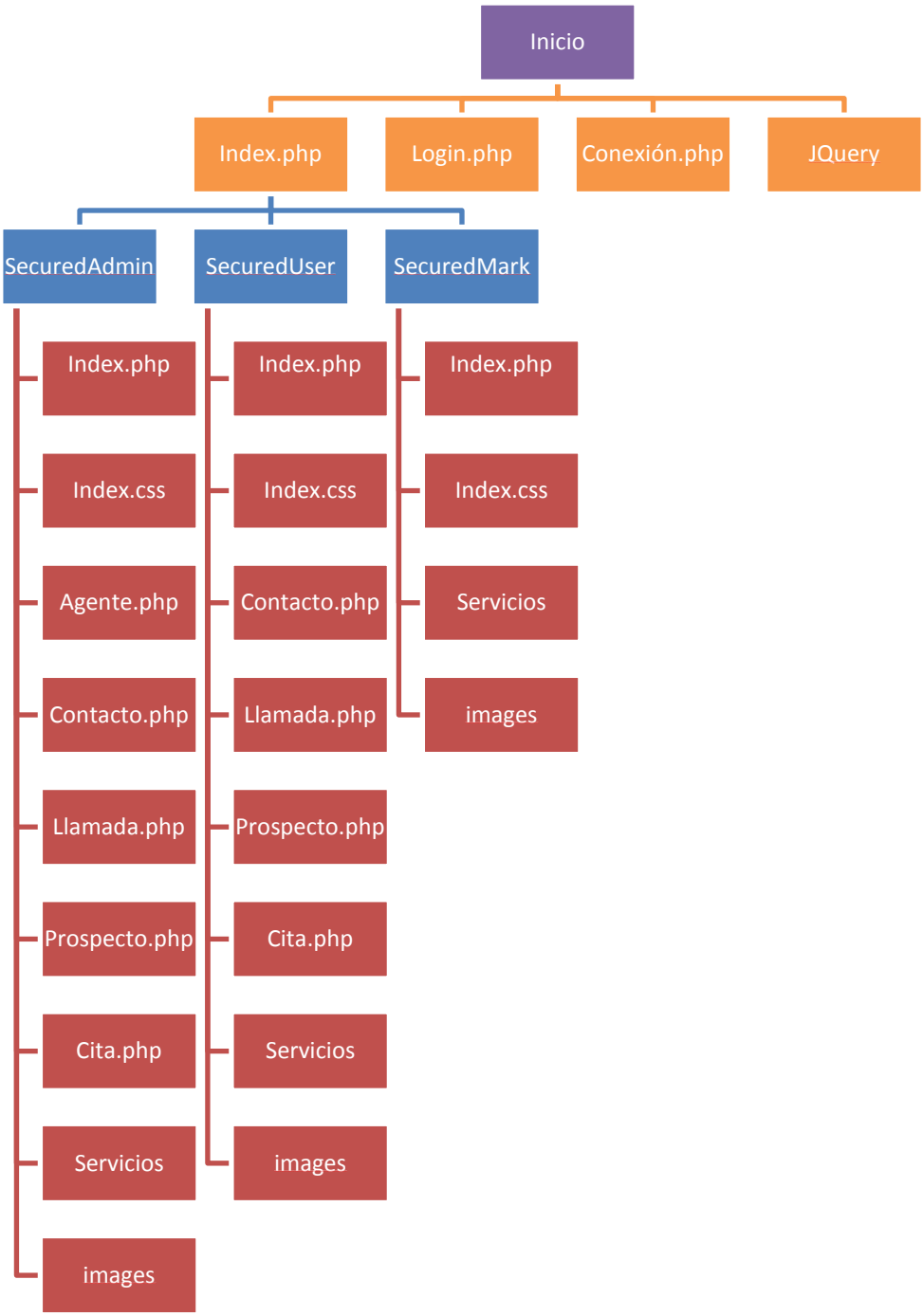

Ilustración 8. Mapa de Sitio

En la ilustración se muestra el mapa del sitio para el sistema.

V.2 Instalación y Configuración de los Servidores Web y de Bases de Datos.

Para el desarrollo y pruebas de este sistema se utilizó el software WAMP SERVER 2.2 que cuenta con el Servidor Apache 2.2, PHP 5.3 y MySQL 5.5.

WAMP SERVER se deberá instalar con su configuración estándar.

La base de datos deberá llevar por nombre: sgcall_db. El nombre de la base de datos debe ser exacto para que la configuración sea correcta. Hecho esto se deberá ejecutar los comandos SQL que están en el archivo "sgcall_db.sql", que son para la creación de la BD en limpio.

Ya teniendo la base de datos se deberá ejecutar el siguiente código SQL:

```
GRANT USAGE ON *.* TO 'SGCallCommon'@'localhost' IDENTIFIED BY PASSWORD
'*B11FB34B15D4B36B7849A86E95B6E0D8B0044008' WITH MAX_USER_CONNECTIONS 100;

GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE ON `sgcall_db`.* TO
'SGCallCommon'@'localhost';

GRANT EXECUTE ON *.* TO 'SGCallLogin'@'localhost' IDENTIFIED BY PASSWORD
'*0149CB1CA46EC324F8617144AEEA0E3436A4AD26' WITH MAX_USER_CONNECTIONS 100;

GRANT SELECT, EXECUTE ON `sgcall_db`.* TO 'SGCallLogin'@'localhost';
```

Este código permitirá crear los usuarios para la base de datos y darles los permisos necesarios. Considere que deberá tener nivel de privilegios de administrador para realizar esta acción.

Como administrador en MySQL solo deberá agregar un agente en la tabla agente y un usuario de tipo ADMIN para el agente.

Con esta acción completada solo será necesario copiar la carpeta SGCall a la carpeta raíz del servidor.

V.3 Creación de Tablas Utilizando el Lenguaje SQL

En la siguiente sección se mostrará el código para poder crear las tablas de la base de datos.

Tabla Agente

Aquí se muestra el código SQL para la creación de la tabla agente.

```
CREATE TABLE IF NOT EXISTS `agente` (
  `RFC_A` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  `nombre_A` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `paterno_A` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `materno_A` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`RFC_A`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Tabla Cita

Aquí se muestra el código SQL para la creación de la tabla cita.

```
CREATE TABLE IF NOT EXISTS `cita` (
  `RFC_A` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  `ID_C` int(11) NOT NULL,
  `fecha_C` date NOT NULL,
  `hora_C` time NOT NULL,
  `asunto_C` longtext COLLATE utf8_unicode_ci NOT NULL,
  `resultado_C` longtext COLLATE utf8_unicode_ci,
  `ID_lugar` int(11) NOT NULL,
  PRIMARY KEY (`fecha_C`,`hora_C`,`RFC_A`,`ID_C`),
  KEY `Programa` (`RFC_A`),
  KEY `con` (`ID_C`),
  KEY `place` (`ID_lugar`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `cita`
  ADD CONSTRAINT `con` FOREIGN KEY (`ID_C`) REFERENCES `contacto` (`ID_C`) ON DELETE CASCADE
ON UPDATE CASCADE,
  ADD CONSTRAINT `place` FOREIGN KEY (`ID_lugar`) REFERENCES `lugar` (`ID_lugar`) ON DELETE NO ACTION
ON UPDATE CASCADE,
  ADD CONSTRAINT `Programa` FOREIGN KEY (`RFC_A`) REFERENCES `agente` (`RFC_A`) ON UPDATE CASCADE;
```

Tabla Lugar

Aquí se muestra el código SQL para la creación de la tabla lugar.

```
CREATE TABLE IF NOT EXISTS `lugar` (
  `ID_lugar` int(11) NOT NULL AUTO_INCREMENT,
  `lugar` varchar(200) COLLATE utf8_unicode_ci NOT NULL,
  `referencia` varchar(300) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`ID_lugar`),
  UNIQUE KEY `lugar` (`lugar`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;
```

Tabla contacto

Aquí se muestra el código SQL para la creación de la tabla contacto.

```
CREATE TABLE IF NOT EXISTS `contacto` (
  `ID_C` int(11) NOT NULL AUTO_INCREMENT,
  `nombres_C` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `paterno_C` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `materno_C` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `correo_C` varchar(30) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`ID_C`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1;
```

Tabla ctelefono

Aquí se muestra el código SQL para la creación de la tabla de teléfono de Contacto (ctelefono).

```
CREATE TABLE IF NOT EXISTS `ctelefono` (
  `IDtel_C` int(11) NOT NULL AUTO_INCREMENT,
  `tel_C` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  `tipo_C` enum('CASA','OFICINA','CEL','OTRO') COLLATE utf8_unicode_ci NOT NULL,
  `ext_C` int(11) DEFAULT NULL,
  `ID_C` int(11) NOT NULL,
  PRIMARY KEY (`IDtel_C`),
  KEY `tiene_tel` (`ID_C`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;

ALTER TABLE `ctelefono`
  ADD CONSTRAINT `tiene_tel` FOREIGN KEY (`ID_C`) REFERENCES `contacto` (`ID_C`) ON DELETE CASCADE ON
UPDATE CASCADE;
```

Tabla empresa

Aquí se muestra el código SQL para la creación de la tabla empresa.

```
CREATE TABLE IF NOT EXISTS `empresa` (
  `ID_E` int(11) NOT NULL AUTO_INCREMENT,
  `nombre_E` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `razonSocial_E` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `correo_E` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
  `calle_E` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `numero_E` smallint(6) NOT NULL,
  `interior_E` smallint(6) DEFAULT NULL,
  `colonia_E` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `ciudad_E` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `municipio_E` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `estado_E` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `giro_E` enum('INDUSTRIAL','COMERCIAL','SERVICIOS') COLLATE utf8_unicode_ci DEFAULT NULL,
  `empleados_E` smallint(6) DEFAULT NULL,
  `ventas_E` float(20,2) DEFAULT NULL,
  PRIMARY KEY (`ID_E`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1;
```

Tabla etelefono

Aquí se muestra el código SQL para la creación de la tabla de teléfono de la empresa (etelefono).

```
CREATE TABLE IF NOT EXISTS `etelefono` (
  `IDtel_E` int(13) NOT NULL AUTO_INCREMENT,
  `tel_E` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  `ext_E` int(11) DEFAULT NULL,
  `ID_E` int(11) NOT NULL,
  PRIMARY KEY (`IDtel_E`),
  KEY `Empresa_Tiene_tel` (`ID_E`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1 ;

ALTER TABLE `etelefono`
  ADD CONSTRAINT `Empresa_Tiene_tel` FOREIGN KEY (`ID_E`) REFERENCES `empresa` (`ID_E`) ON DELETE
CASCADE ON UPDATE CASCADE;
```

Tabla llamada

Aquí se muestra el código SQL para la creación de la tabla de llamada.

```
CREATE TABLE IF NOT EXISTS `llamada` (
  `RFC_A` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  `ID_C` int(11) NOT NULL,
  `fecha_L` date NOT NULL,
  `hora_L` time NOT NULL,
  `asunto_L` longtext COLLATE utf8_unicode_ci NOT NULL,
  `resultado_L` longtext COLLATE utf8_unicode_ci,
  `tipo_L` enum('LLAMADA','CORREO') COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`RFC_A`,`ID_C`,`fecha_L`,`hora_L`),
  KEY `Llamado` (`ID_C`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `llamada`
  ADD CONSTRAINT `Llamado` FOREIGN KEY (`ID_C`) REFERENCES `contacto` (`ID_C`) ON DELETE CASCADE ON
UPDATE CASCADE,
  ADD CONSTRAINT `Llama_a` FOREIGN KEY (`RFC_A`) REFERENCES `agente` (`RFC_A`);
```

Tabla prospecto

Aquí se muestra el código SQL para la creación de la tabla de prospecto.

```
CREATE TABLE IF NOT EXISTS `prospecto` (
  `calle_P` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `numero_P` smallint(6) NOT NULL,
  `interior_P` smallint(6) DEFAULT NULL,
  `colonia_P` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `ciudad_P` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `municipio_P` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `estado_P` varchar(30) COLLATE utf8_unicode_ci NOT NULL,
  `tipo_P` enum('CLIENTE','PROSPECTO') COLLATE utf8_unicode_ci NOT NULL,
  `importe_P` float(20,2) DEFAULT NULL,
  `ID_C` int(11) NOT NULL,
  PRIMARY KEY (`ID_C`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `prospecto`
  ADD CONSTRAINT `CONTACTO_PROSPECTO` FOREIGN KEY (`ID_C`) REFERENCES `contacto` (`ID_C`) ON DELETE
CASCADE ON UPDATE CASCADE;
```

Tabla prospecto_empresa

Aquí se muestra el código SQL para la creación de la tabla intermedia entre empresa y prospecto.

```
CREATE TABLE IF NOT EXISTS `prospecto_empresa` (
  `ID_E` int(11) NOT NULL,
  `ID_C` int(11) NOT NULL,
  PRIMARY KEY (`ID_E`,`ID_C`),
  KEY `Pertenece_a_PROSPECTO` (`ID_C`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `prospecto_empresa`
  ADD CONSTRAINT `Pertenece_a_EMPRESA` FOREIGN KEY (`ID_E`) REFERENCES `empresa` (`ID_E`) ON DELETE
CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `Pertenece_a_PROSPECTO` FOREIGN KEY (`ID_C`) REFERENCES `prospecto` (`ID_C`) ON DELETE
CASCADE ON UPDATE CASCADE;
```

Tabla de usuario

Aquí se muestra el código SQL para la creación de la tabla usuarios.

```
CREATE TABLE IF NOT EXISTS `sgcall_uda` (
  `usuario` varchar(20) COLLATE utf8_unicode_ci NOT NULL,
  `pass` varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  `privilegios` enum('ADMIN','MARKET','USER') COLLATE utf8_unicode_ci NOT NULL,
  `RFC_A` varchar(13) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`usuario`),
  UNIQUE KEY `es` (`RFC_A`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `sgcall_uda`
  ADD CONSTRAINT `es` FOREIGN KEY (`RFC_A`) REFERENCES `agente` (`RFC_A`) ON DELETE CASCADE ON UPDATE
CASCADE;
```

Procedimientos Guardados

Aquí se muestra el código SQL para la creación de los procedimientos.

```
DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `uda_add`(IN `RFC` VARCHAR(13), IN `usuario` VARCHAR(20), IN
`pass` VARCHAR(50), IN `priv` VARCHAR(20))
    NO SQL
INSERT INTO sgcall_uda(usuario, pass, privilegios, RFC_A) values (usuario, MD5(pass), priv, RFC)$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `uda_check`(IN `user` VARCHAR(50), IN `psswd` VARCHAR(50))
    NO SQL
SELECT usuario, pass, privilegios,RFC_A FROM sgcall_uda WHERE usuario = user AND pass = MD5(psswd)$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `uda_delete`(IN `user` VARCHAR(20))
    NO SQL
DELETE
FROM sgcall_uda
WHERE usuario = user$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `uda_edit`(IN `new_user` VARCHAR(20), IN `user` VARCHAR(20),
IN `password` VARCHAR(50), IN `priv` VARCHAR(20))
    NO SQL
UPDATE sgcall_uda
SET usuario= new_user, pass=MD5(password), privilegios=priv
WHERE usuario=user$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `uda_find`(IN `usr` VARCHAR(20))
    NO SQL
SELECT *
FROM sgcall_uda
WHERE usuario = usr$$

DELIMITER ;
```

V.4 Estructura Final de la Base de Datos.

Después del análisis y la normalización aplicada a la bases de datos se obtuvo la estructura que se muestra en la Ilustración 9.

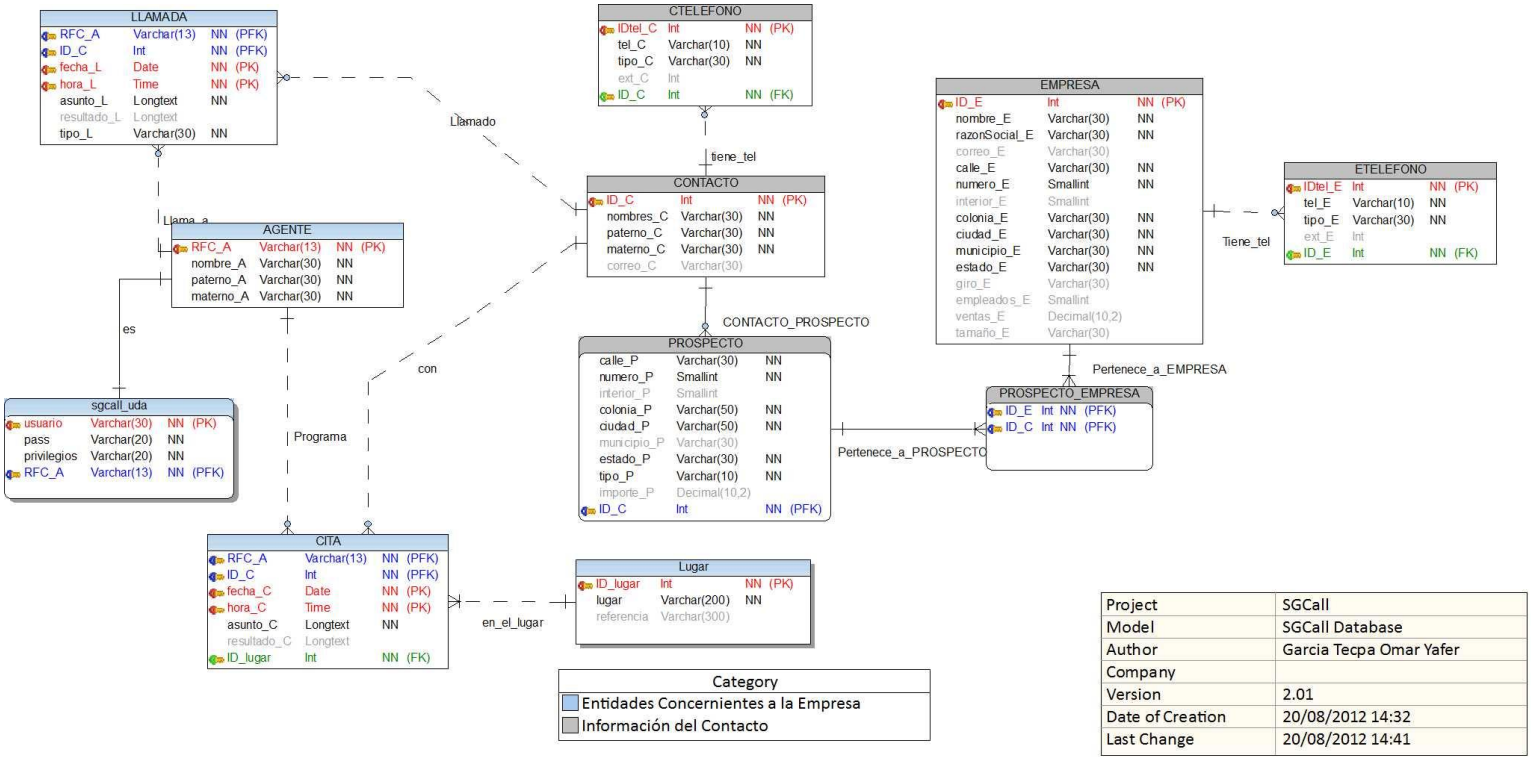

Ilustración 9. Modelo Lógico de la Base de Datos

En la ilustración se muestra el Modelo Lógico final de la base de datos.

V.5 Consultas.

En esta sección se mostrará una breve descripción y el código de algunas de las consultas realizadas en el sistema. En algunos casos se tienen palabras que tienen antepuesto un símbolo $ éstas palabras representan un valor válido para la tabla y que será el valor que ocuparemos en la consulta.

Las siguientes consultas están relacionadas con los usuarios del sistema.

Llama al procedimiento almacenado que busca a un usuario específico.

```
CALL uda_find('$user');
```

Llama al procedimiento almacenado que crea a un usuario.

```
CALL uda_add('$RFC','$user','$pass1','$priv');
```

Llama al procedimiento almacenado que actualiza a un usuario.

```
CALL uda_edit('$user','$userOld','$pass1','$priv')
```

Llama al procedimiento almacenado que elimina un usuario específico.

```
CALL uda_delete('$user');
```

Las siguientes consultas tienen que ver con los agentes.

Selecciona los agentes a los cuales todavía no se les ha asignado un usuario.

```
SELECT *
FROM agente
WHERE RFC_A NOT IN(SELECT RFC_A FROM sgcall_uda);
```

Selecciona todos los agentes en el sistema.

```
SELECT usuario, privilegios, agente.RFC_A, CONCAT(paterno_A,' ',materno_A,' ',nombre_A) AS nombre
FROM sgcall_uda, agente
WHERE sgcall_uda.RFC_A = agente.RFC_A
ORDER BY nombre;
```

Selecciona todos los agentes en el sistema junto con sus datos de usuario.

```
SELECT agente.RFC_A, paterno_A, materno_A, nombre_A, usuario, privilegios
FROM agente LEFT JOIN sgcall_uda ON agente.RFC_A = sgcall_uda.RFC_A
ORDER BY paterno_A, materno_A, nombre_A ;
```

Selecciona a un agente específico.

```
SELECT RFC_A, paterno_A, materno_A, nombre_A
FROM agente
WHERE RFC_A = '$RFC';
```

Las siguientes consultas están relacionadas con el contacto.

Selecciona todos los contactos.

```
SELECT *
FROM contacto
ORDER BY paterno_C, materno_C, nombres_C;
```

Selecciona a un contacto que aun no exista en la lista de prospectos en específico.

```
SELECT *
FROM contacto
WHERE ID_C NOT IN(SELECT ID_C FROM prospecto);
```

Selecciona todos los teléfonos de un contacto específico.

```
SELECT DISTINCT IDtel_C, tel_C, tipo_C, ext_C
FROM contacto, ctelefono
WHERE ctelefono.ID_C ='$contacto'
```

Estas consultas están relacionadas con los prospectos.

Selecciona todos los prospectos junto con su información de contacto.

```
SELECT *
FROM contacto, prospecto
WHERE contacto.ID_C = prospecto.ID_C ORDER BY paterno_C, materno_C, nombres_C;
```

Selecciona a un prospecto en específico.

```
SELECT * FROM contacto, prospecto WHERE prospecto.ID_C = '$prospecto'
```

Selecciona a las empresas relacionadas con un prospecto específico.

```
SELECT DISTINCT empresa.ID_E, nombre_E, razonSocial_E, correo_E, calle_E, numero_E, interior_E, colonia_E,
ciudad_E, municipio_E, estado_E, giro_E, empleados_E,ventas_E
FROM empresa, prospecto, prospecto_empresa
WHERE empresa.ID_E = prospecto_empresa.ID_E AND prospecto_empresa.ID_C = '$prospecto';
```

Estas consultas están relacionadas con las llamadas.

Selecciona todas las llamadas

```
SELECT *
FROM agente, contacto, llamada
WHERE agente.RFC_A = llamada.RFC_A AND contacto.ID_C= llamada.ID_C
ORDER BY paterno_A, Materno_A, nombre_A, paterno_C, materno_C, nombres_C
```

Selecciona las llamadas hechas por un agente.

```
SELECT *
FROM contacto, llamada
WHERE llamada.RFC_A='$agente' AND contacto.ID_C= llamada.ID_C
ORDER BY paterno_C, materno_C, nombres_C
```

Selecciona las llamadas a un determinado contacto hechas por un agente específico.

```
SELECT * FROM llamada
WHERE llamada.RFC_A='$agente' AND llamada.ID_C= '$contacto'
ORDER BY fecha_L, hora_L
```

Estas consultas están relacionadas con las citas.

Selecciona todas las citas.

```
select * FROM contacto, lugar, cita, agente
WHERE contacto.ID_C = cita.ID_C AND lugar.ID_lugar = cita.ID_lugar AND agente.RFC_A= cita.RFC_A
ORDER BY paterno_A, materno_A, nombre_A, paterno_C, materno_C, nombres_C, fecha_C, hora_C;
```

Selecciona las citas realizadas con un determinado agente. Además muestra los datos de los participantes.

```
SELECT * FROM agente, contacto, cita, lugar
WHERE agente.RFC_A = cita.RFC_A AND contacto.ID_C= cita.ID_C AND agente.RFC_A='$agente' AND
lugar.ID_lugar= cita.ID_lugar AND fecha_C > CURRENT_DATE AND resultado_C = ''
ORDER BY fecha_C, hora_C;
```

Selecciona las citas realizadas entre un determinado contacto, prospecto o cliente y un agente específico.

```
SELECT * FROM  cita
WHERE  cita.RFC_A='$agente' AND cita.ID_C= '$contacto'
ORDER BY  fecha_C, hora_C;
```

V.6 CSS.

En el capítulo 2 sección 5.2 vimos qué es CSS. En esta sección describiremos parte de las reglas CSS para el sistema SGCall. Se hace primero una breve descripción del funcionamiento de las reglas y posteriormente se muestra el código CSS.

- Esta regla establece el color final para el cuerpo de la página.

```
body{background:#C0C9D8 url(images/img01.jpg) repeat-x;  }
```

- Esta regla establece solamente la alineación del vínculo para cerrar sesión.

```
#logout{text-align: right;}
```

- Esta regla establece el tamaño del envoltorio de la página.

```
#wrapper{
        width:                  1023px;
        margin:             0px auto;
        padding:                0px;
        background:             url(images/call.png) no-repeat bottom right;
        }
```

- Esta regla establece cómo se mostrará el encabezado. Además, pone el logo de la página.

```
header{
        display:                block;
        width:                  1020px;
        height:                 220px;
        margin:                 23px auto;
        color:                  rgba(255,255,255,1);
        text-align:             center;
        background:             url(images/Logo.png) top left no-repeat;
        }
```

- Las siguientes tres reglas competen a como se mostrarán los títulos de la página.

```
h1{
        display:                inline-block;
        width:                  600px;
        float:                  right;
        font-size:              3em;
        }

h2{color: white;}
```

```
h4{
        margin-left:                150px;
        text-shadow:                1px 1px 3px rgba(0,0,0,0.7);
}
```

- Las cuatro reglas siguientes son para especificar cómo es que se muestra el menú y los vínculos en él. En el caso del primer y último vinculo lo que se hace es redondear los bordes exteriores para dar un acabado tipo barra.

```
#menu{
        display:                    block;
        width:                      1020px;
        height:                     60px;
        margin:                     20px auto;
        padding:                    10px;
        clear:                      both;
        }

.menuLink{
        display:                    inline-block;
        color:                      #fff;
        width:                      105px;
        height:                     43px;
        padding-top:                13px;
        margin:                     0px auto;
        background:                 url(images/menu.png) repeat-x top left;
        text-align:                 center;
        font-size:                  1.3em;
        text-decoration:            none;
        border-collapse:            collapse;
        text-shadow:                2px 2px 3px rgba(255,255,255,0.5);
        -webkit-box-shadow:         2px 2px 5px 0px rgba(0,0,0,0.3);
        -moz-box-shadow:            2px 2px 5px 0px rgba(0,0,0,0.3);
        -ms-box-shadow:             2px 2px 5px 0px rgba(0,0,0,0.3);
        box-shadow:                 2px 2px 5px 0px rgba(0,0,0,0.3);
        }

.menuLink:first-child{
        display:                    inline-block;
        border-top-left-radius:     15px;
        border-bottom-left-radius:  15px;
        }

.menuLink:last-child{
        border-top-right-radius:    15px;
        border-bottom-right-radius: 15px;
        }
```

- Este es un caso especial de los vínculos del menú. Con esta regla se cambia el fondo del vínculo para reflejar qué sección del sistema se está visitando.

```
#selected{background:                url(images/inicio.png) repeat-x top left;}
```

- Ésta regla especifica las características del pie de la página.

```
footer{
        color:                       #fff;
        font-size:                   1em;
        text-align:                  center;
        clear:                       both;
        }
```

- Ésta regla establece cómo se debe mostrar el envoltorio del contenido principal de la página.

```
#page{
        width:                       950px;
        height:                      1100px;
        margin:                      20px auto;
        }
```

- Esta serie de reglas definen el aspecto visual de los envoltorios de las tabs que se muestran en la página.

```
.tabwrap{ max-width: 80%; margin:30px auto; max-height:850px;  overflow:auto; overflow-style:marquee-block;}

.tabwrap form{ max-width: 100%; min-width: 100%;}

.tabwrap2{ width: auto; max-height:850px; margin:30px auto; overflow:auto; overflow-style:marquee-block;
}

.tabwrap table{margin:30px auto;}

.tabwrap th{text-align: left; }

.innerwrap{margin: 30px auto; }
```

- Esta última parte de las reglas CSS establecen las cualidades visuales de los elementos del form.

```
fieldset{
	border-radius:15px;
	background: #fff;
	margin: 30px auto;
	-webkit-box-shadow: 2px 2px 5px 0px rgba(0,0,0,0.3);
	-moz-box-shadow: 2px 2px 5px 0px rgba(0,0,0,0.3);
	-ms-box-shadow: 2px 2px 5px 0px rgba(0,0,0,0.3);
	box-shadow: 2px 2px 5px 0px rgba(0,0,0,0.3);
}


label{display:inline-block; width: 100px; color:blue;}

label:hover{ cursor:pointer}

input{border-radius:15px; padding:5px;}

select{border-radius:15px; padding:5px; border-color:gray;}

input[type="submit"]{ display:block;  padding:10px; border-radius:15px; font-weight:bold; float:right}


#loginwrap{margin: 0px auto;}

#loginwrap{	width:250px; margin:	200px auto; text-align:center;	}

#loginwrap th{color: white; text-align: left;}

.wrapconsulta{ max-height: 800px; max-width: 850px;  overflow:auto; overflow-style:auto; margin: 30px auto;  }
.consultas{ border-collapse: collapse;  margin: 0px auto;  }
.consultas th{
	text-align:left;
	padding: 5px 15px;
	background:url(images/ui-bg_highlight-soft_75_cccccc_1x100.png) top right  repeat-x;
	color:#FFFFFF;
	text-shadow: 1px 1px 3px rgba(0,0,0,0.7);
}

.selectedHeader{
	border-bottom-left-radius:10px; border-top-left-radius:10px;
}


.consultas th:first-child{border-bottom-left-radius:10px; border-top-left-radius:10px;}
.consultas th:last-child{border-bottom-right-radius:10px; border-top-right-radius:10px;}
.consultas td{text-align:left; padding: 5px 15px;}
.consultas tr{border-top: 1px solid gray;}
.consultas tr:first-child{border-top: none;}
```

```
legend{  font-weight: bold;text-shadow: 1px 1px 2px rgba(0,0,0,0.5);  }
.accordionContent{min-height:300px; max-height:500px;}
.prospectAccordion{min-height:650px; max-height:700px}
```

V.7 Pantallas del Sistema.

Esta sección está dedicada a mostrar pantallas del sistema en funcionamiento. Primero se mostrará una breve descripción de la imagen.

En la Ilustración 10 se muestra la página de acceso del Sistema. Esta página es la primera en mostrarse. Se gana acceso al sistema introduciendo el usuario y contraseña correctos. Habiendo ingresado al sistema, será re direccionado dependiendo del tipo de usuario.

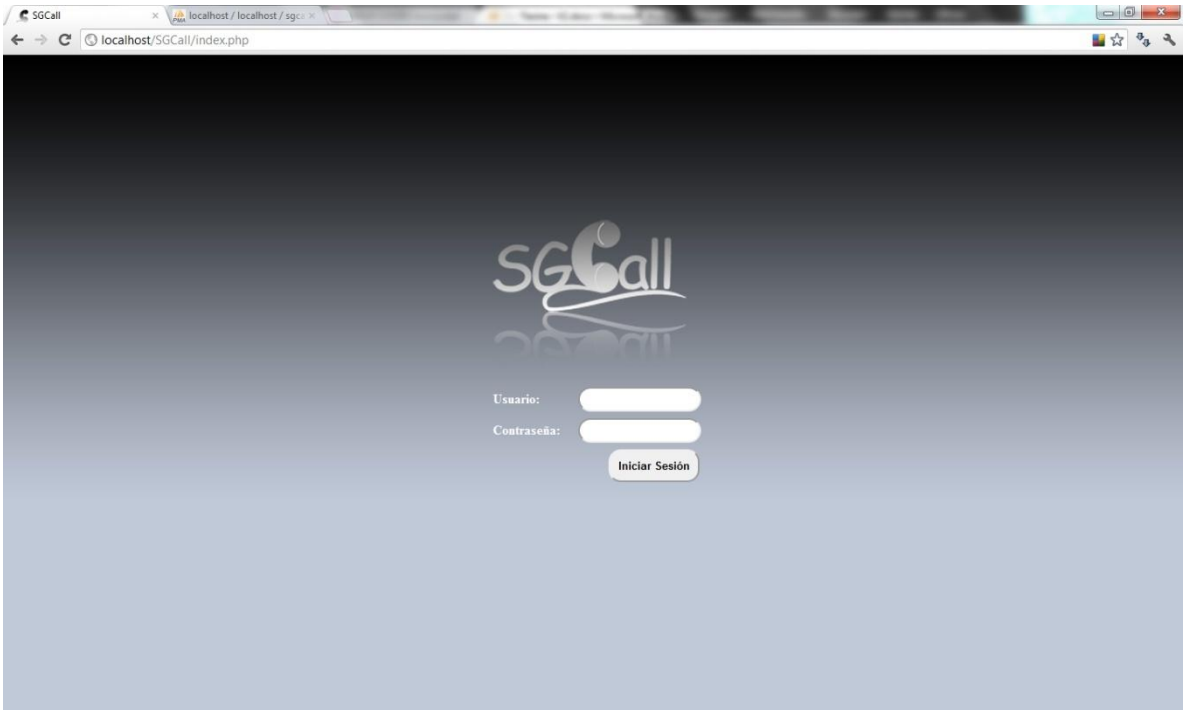

Ilustración 10. Login del Sistema
En la ilustración se muestra la página de acceso del sistema.

En la Ilustración 11 se muestra la pantalla inicial del administrador. El administrador es el que tiene la mayor cantidad de opciones. En el caso del administrador y usuario la página de inicio muestra los pendientes del agente.

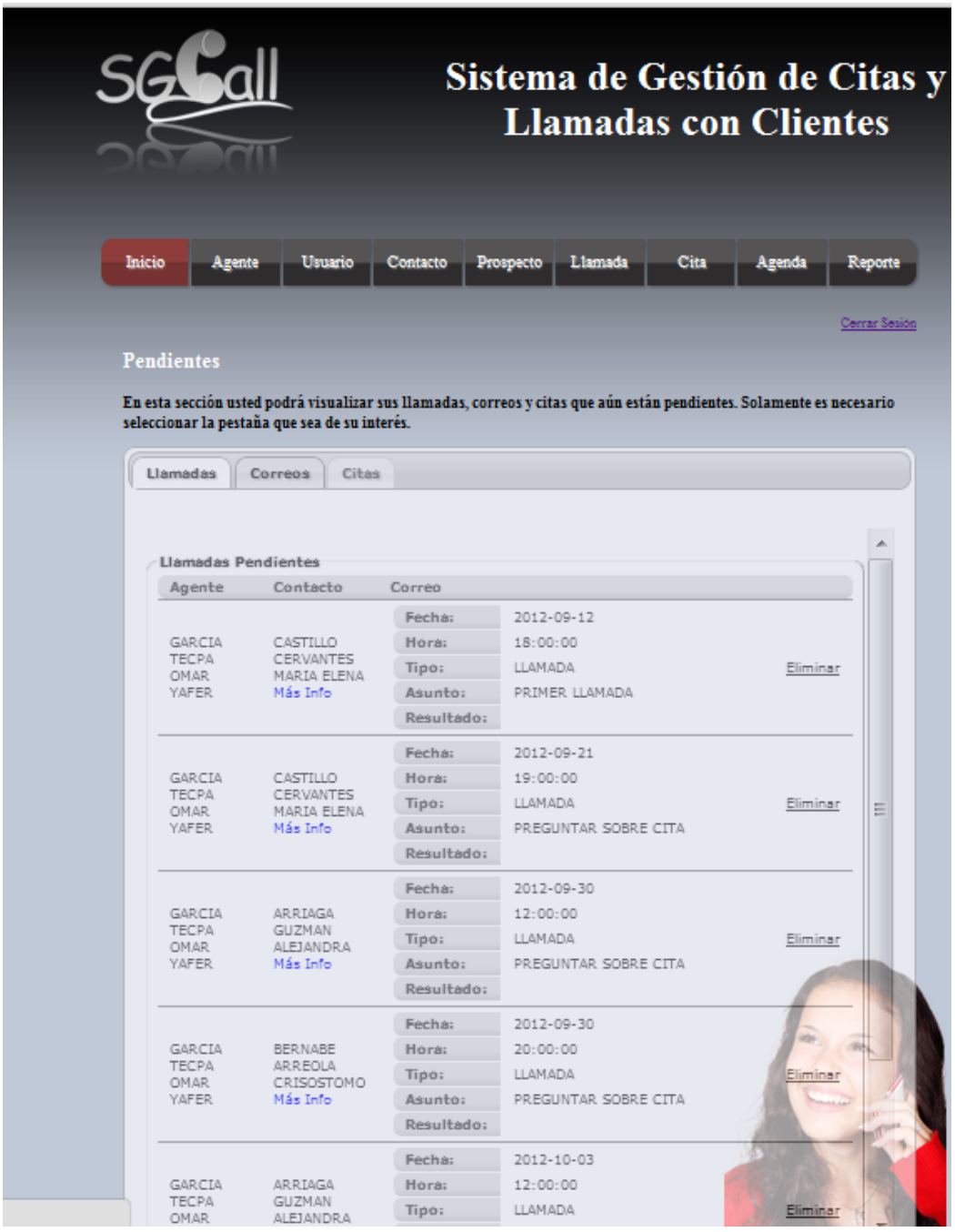

Ilustración 11. Página Inicial del Administrador

En la ilustración se muestra la pantalla principal para el administrador del sistema.

En la Ilustración 12 se muestra la página principal del agente de ventas. Como se puede observar el agente tiene menos opciones que las que tiene el administrador. Al igual que el administrador, en la página de inicio se muestran los pendientes.

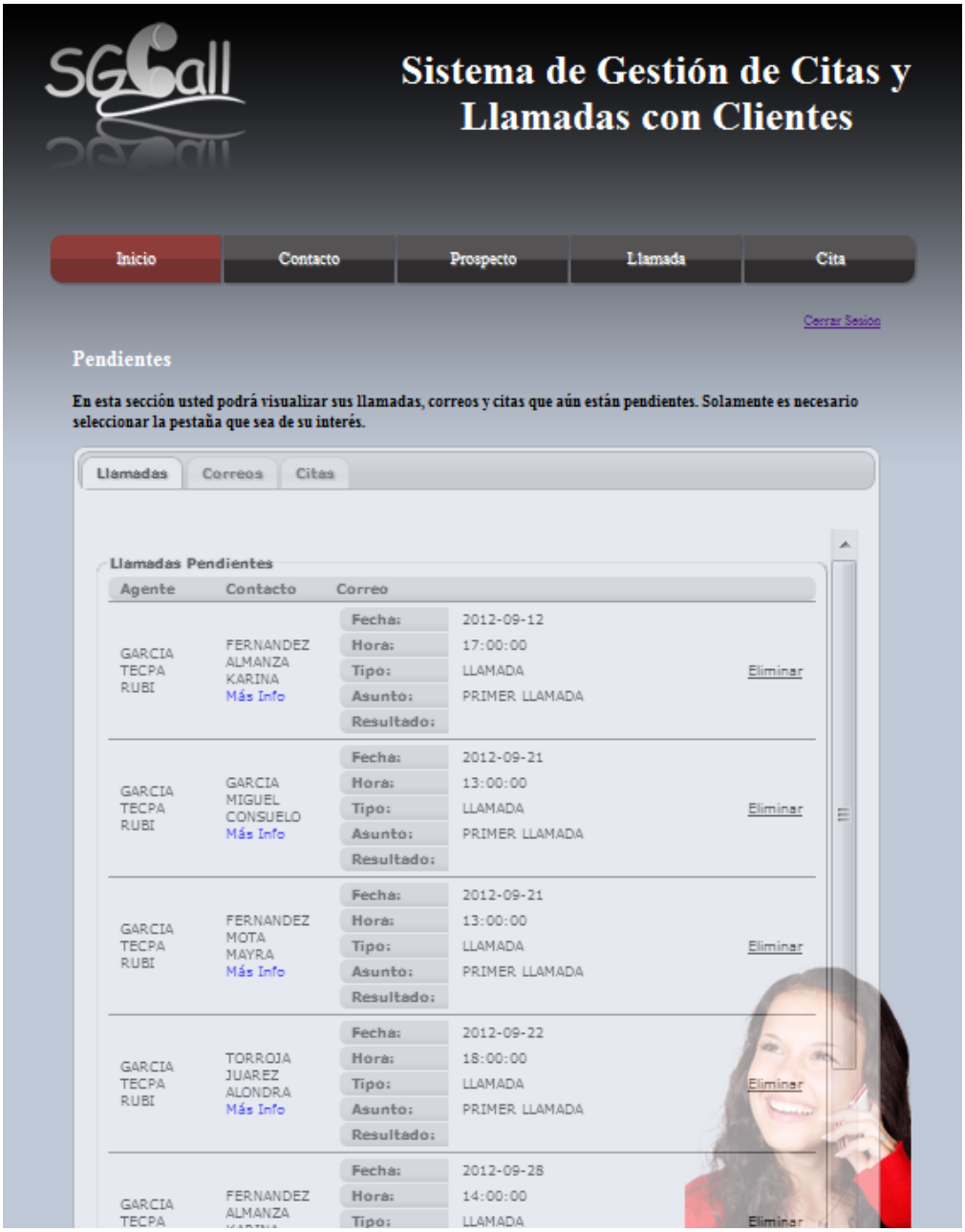

Ilustración 12. Página Inicial del Agente de Ventas

En la ilustración se muestra la página principal para el agente de ventas.

En la Ilustración 13 se muestra la página del encargado de marketing. Este tipo de usuario es el que tiene la menor cantidad de opciones. El encargado de marketing solo puede ver el calendario de actividades y los reportes.

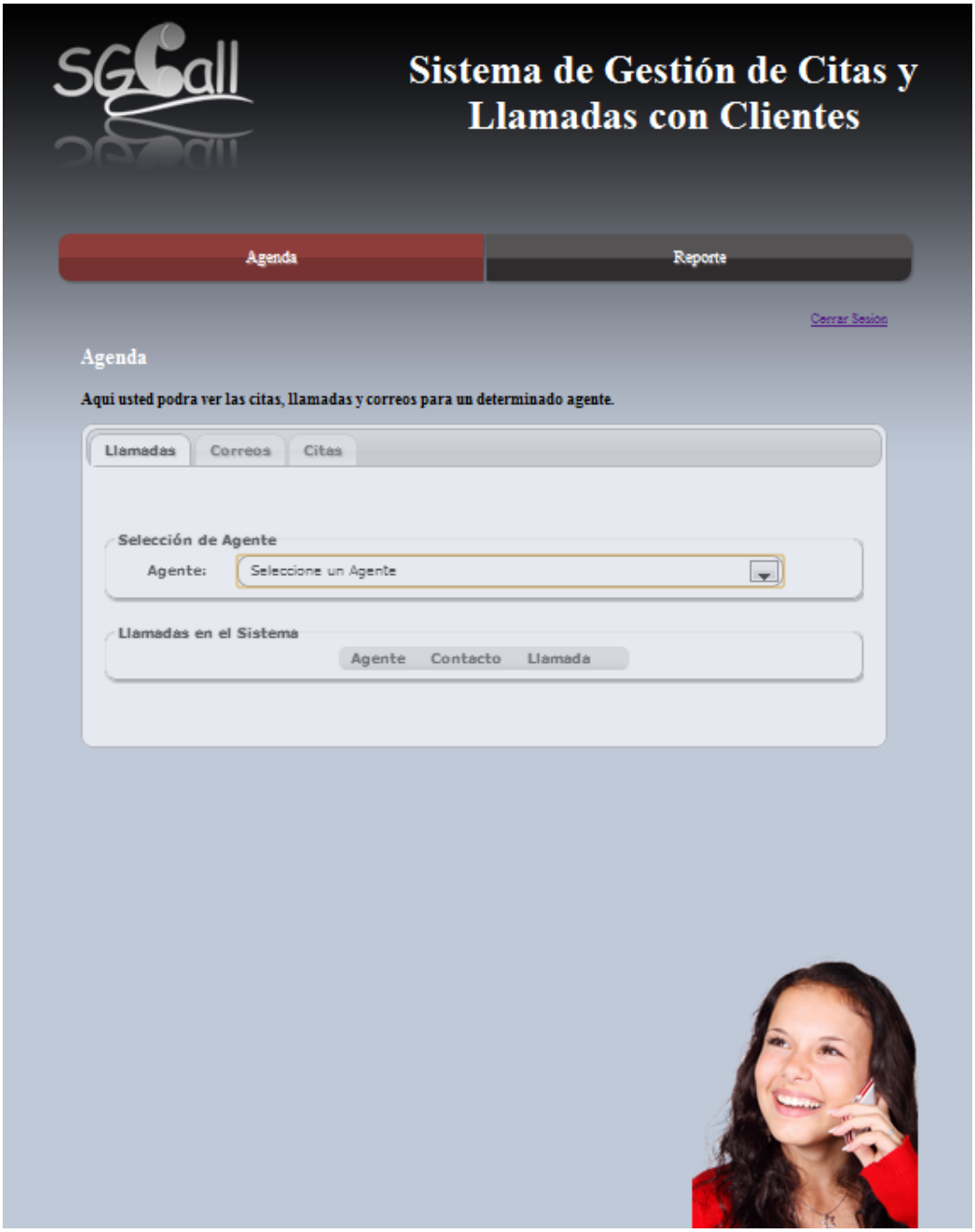

Ilustración 13. Página inicial del Encargado de Marketing
En la ilustración se muestra la página inicial para el encargado de marketing.

En las ilustraciones 14, 15 y 16 se muestra la página del agente. En cada ilustración se ha seleccionado un TAB diferente. En las páginas de agente, usuario, prospecto, llamada y cita se tiene la misma estructura. En estas páginas existen tres TABS, para agregar, editar y mostrar.

Ilustración 14. Agregar Agente

En la ilustración se muestra la tab agregar de la página Agente del Sistema.

Ilustración 15. Editar Agente

En la ilustración se muestra la tab editar de la página Agente del Sistema.

Ilustración 16. Mostrar Agente

En la ilustración se muestra la tab mostrar de la página Agente del Sistema.

Aunque las páginas agente, usuario, prospecto, llamada y cita tienen muchas similitudes, la página de prospecto tiene unas pequeñas diferencias. Debido a la gran cantidad de información que se necesita capturar con respecto al cliente, se dividió el formulario en dos. Una división para la información que es solo del prospecto, y otra que es de la empresa con la que se relaciona. Estas divisiones están presentes tanto en la TAB de agregar como en la de editar (Ilustración 17).

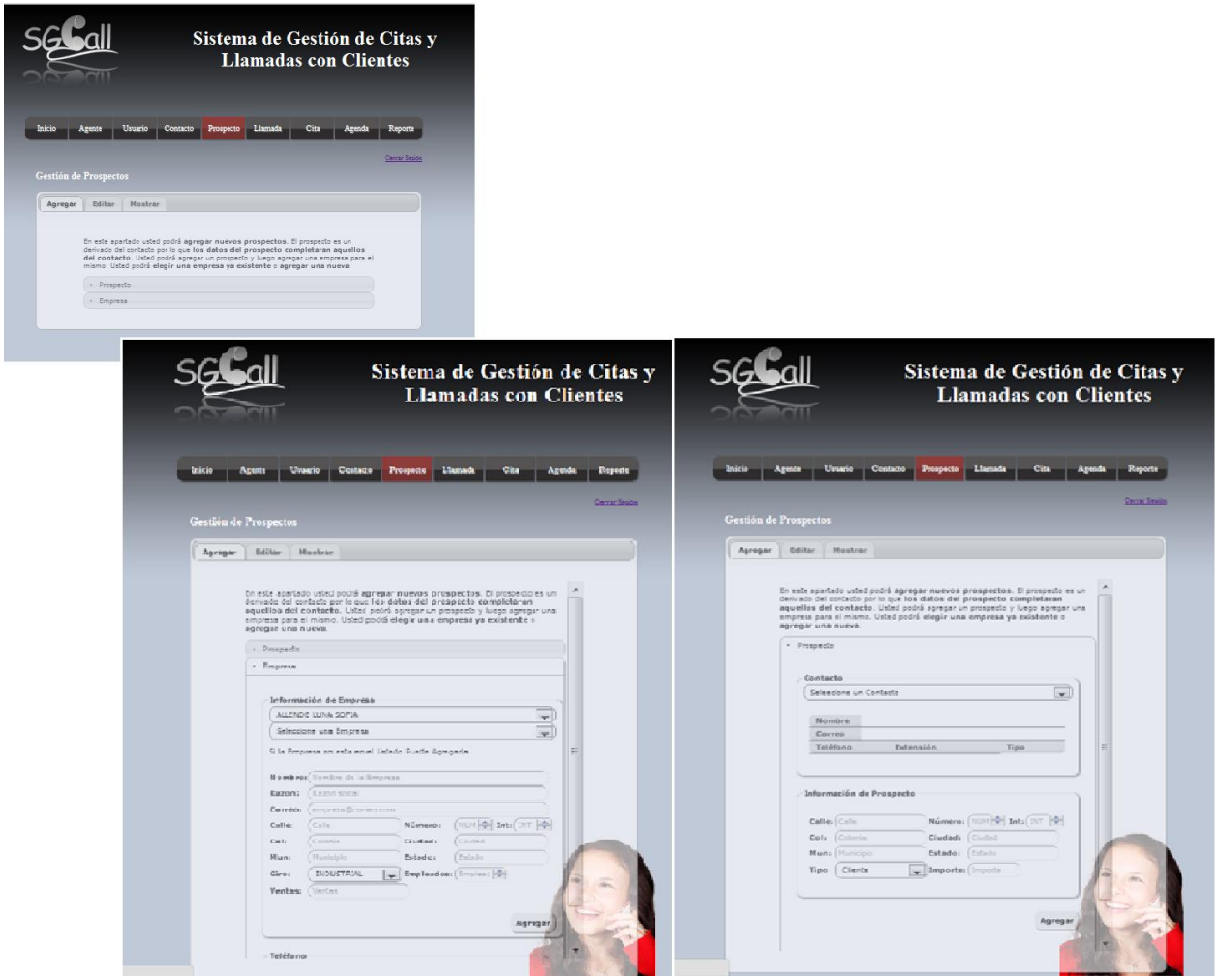

Ilustración 17. Agregar Prospecto

En la ilustración se muestra la tab agregar de la página Prospecto del Sistema.

En la página de agenda se puede seleccionar que actividad revisar: Llamadas, correos o Citas. Una vez elegida la actividad se puede elegir un agente. Habiendo elegido al agente se mostrarán las actividades que tiene el agente (Ilustración 18).

Ilustración 18. Agenda

En la ilustración se muestra la página Agenda del Sistema.

Al dar clic en el vínculo más info se mostrará información más detallada acerca del usuario tal y como se muestra en la Ilustración 19. Esta opción de mostrar más información está presente en varias partes del sistema donde se necesite ver mayor información del contacto.

Ilustración 19. Detalles

En la ilustración se muestra la Caja de Dialogo que muestra los información detallada del sistema.

V.8 Reportes.

En esta sección se describirán los reportes que son generados por el sistema.

El primer reporte nos permite seleccionar un agente y dos fechas que delimitarán un periodo. El sistema mostrará todas las llamadas, correos y citas para el agente durante el periodo. En la Ilustración 20 se muestra el reporte para un agente durante el periodo agosto-septiembre del 2012.

Ilustración 20. Reporte Delimitado por Dos Fechas
En la ilustración se muestra un reporte que corresponde al periodo delimitado por dos fechas.

El otro reporte que se formuló para el sistema es mostrar la agenda de actividades de un agente en específico. Las citas son ordenadas por fecha, hora y contacto. Esta acción se muestra en la Ilustración 21.

Ilustración 21. Reporte Agenda
En la ilustración se muestra la Agenda para un Agente.

CAPÍTULO VI. **Conclusiones y Trabajo a Futuro**

VI.1 Conclusiones

A través del presente proyecto fue posible desarrollar un proyecto de bases de datos de principio a fin. Se pudo analizar y diseñar el software, si bien mediante una metodología más simple. Sin embargo, con esta metodología fue posible hacer un correcto análisis para el sistema.

Con el uso de algunas herramientas de ingeniería de software se diseñó la manera en que se debía construir el sistema. Se definieron las funciones y capacidades con las que el sistema debería de cumplir. Además, se especificó la información que se esperaba que el sistema pudiera recabar, para que, de esta manera se definiera lo que se necesitaba guardar en la base de datos.

En cuanto a la base de datos se aplicaron el modelo entidad-relación y el modelo relacional para poder diseñarla. Una vez diseñada la base de datos se aplicó normalización. Posteriormente, se construyó la BD utilizando el lenguaje SQL y el sistema gestor de bases de datos MySQL.

Para acceder a la base de datos se creó una página web dinámica. Esta página web se construyó utilizando HTML5 para estructurar la página y se utilizó CSS3 para especificar las propiedades visuales. Además, para que la página fuese dinámica era necesario utilizar el lenguaje PHP que genera dinámicamente código HTML.

Tratando de agregar un poco más de dinamismo y mejor acabado visual se optó por el uso de JAVASCRIPT, específicamente la librería JQUERY y JQUERY UI. Con estas dos librerías se pudo dar una mejor organización visual a la página.

Todo este proceso de análisis, diseño y desarrollo permitió concluir con éxito el sistema que en esta tesina se ha presentado.

VI.2 Trabajo a Futuro

Para trabajo a futuro se planean unas mejoras para el sistema. Para disminuir la redundancia de datos y los tiempos de captura se planea utilizar las bases de datos del Servicio Postal Mexicano (SEPOMEX) para poder agregar direcciones. La implementación de ésta mejora haría más eficiente el llenado de formularios. Otra mejora pertinente será mostrar los resultados de las consultas en diferentes secciones (páginas) de 10 o 20 elementos cada una. Por último se agregará una opción que permita exportar los datos de la agenda y reportes a PDF.

Se planea que estas mejoras permitan que el sistema sea más ágil y brinde un mejor servicio.

Bibliografía

1. **Cleri, Carlos.** *El libro de las PyMEs.* Buenos Aires : Granica, 2007.

2. **SECRETARIA DE ECONOMIA.** ACUERDO por el que se establece la Estratificación de Micro, Pequeñas y Medianas Empresas. *Diario Oficial de la Federación.* 30 de Junio de 2009.

3. **Christopher, Martin, Payne, Adrian y Ballantyne, David.** *MARKETING RELACIONAL. Integrando la Calidad, el Servicio al Cliente y el Marketing.* Madrid : Diaz de Santos, 1994.

4. **García Varcárcel, Ignacio.** *CRM. GEstión de la Relación con los Clientes.* España : FC Editorial, 2001.

5. **Yourdon, Edward.** *Analisis Estructurado Moderno .* México : Prentice Hall Hispanoamerica S.A., 1997.

6. **Whitten, Jeffrey L y Bentley, Lonnie D.** *Systems analysis & design methods.* Boston : McGraw-Hill/Irwi, 2007.

7. **Pressman, Roger S.** *INGENIERÍA DEL SOFTWARE. Un Enfoque Práctico.* Madrid : McGraw Hill/Interamericana de España, S.A.U., 2002.

8. **Cobo Yera, Ángel.** *Diseño y programación de bases de datos.* Madrid : Visión Libros, 2007.

9. **Nevado Cabello, Ma Victoria.** *Introducción a las bases de datos relacionales .* Madrid : Visión Libros, 2010.

10. **Gilfillan, Ian.** *La Biblia de MySQL.* Madrid : Anaya Multimedia, 2003.

11. **Lowery, Joseph W. y Fletcher, Mark.** *HTML5 24-Hour Trainer.* Indianapolis : Wiley Publishing, Inc., 2011.

12. **Teague, Jason Cranford.** *Visual Quickstart Guide .* Berkley : Peachpit Press, 2011.

13. **Cabezas Granado, Luis Miguel.** *Manual Imprescindible de PHP.* Madrid : Anaya Multimedia, 2004.

14. **Kabir, Mohammed J.** *La Biblia de servidor Apache 2.* Madrid : Anaya Multimedia, 2003.

15. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : Pearson Educación S.A., 2005.

16. **De Miguel Castaño, Adoración, Piattini Velthuis, Mario y Marcos Martínez, Esperanza.** *Diseño de Bases de Datos Relacionales.* Madrid : AlfaOmega; Ra-Ma, 2000.

17. **Elmasri, Ramez y Navathe, Shamkant B.** *SISTEMAS DE BASES DE DATOS. Conceptos Fundamentales.* México : Addison-Wesley Iberoamericana, 1997.

18. **INEGI.** INEGI. *INEGI.* [En línea] 2004. [Citado el: 08 de Julio de 2012.] http://www.inegi.org.mx/prod_serv/contenidos/espanol/bvinegi/productos/censos/economicos/2004/industrial/estratifica2004.pdf.

19. **SECRETARIA DE ECONOMIA.** MEXICO EMPRENDE. [En línea] 2010. [Citado el: 31 de mayo de 2012.] http://www.economia.gob.mx/index.php/mexico-emprende.