UNIVERSIDAD PERUANA LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS

DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE CÓDIGO DE BARRAS CON LA METODOLOGÍA XP PARA OPTIMIZAR EL CONTROL DE ASISTENCIA EN LA JUNTA ADMINISTRADORA DE SERVICIOS DE SANEAMIENTO QUILCAS

Área de investigación: Software e Ingeniería

Líneas de investigación: Ingeniería de Software

PRESENTADO POR:

BACH. IVÁN DIEGO RIVERA MEZA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS Y COMPUTACIÓN

HUANCAYO-PERÚ

2017

_____

DR. CASIO AURELIO TORRES LOPEZ
PRESIDENTE

_____

JURADO

_____

JURADO

_____

JURADO

_____

MG. MIGUEL ÁNGEL CARLOS CANALES
SECRETARIO DOCENTE

DR. MAGNO BALDEÓN TOVAR
ASESOR METODOLÓGICO

ING. JESSICA VILCHEZ GUTARRA
ASESOR TEMÁTICO

DEDICATORIA

Dedicado a mis padres Luciano y Dora quiénes me han brindado su apoyo incondicional, a mis tíos, Clenio y Vilma que siempre cuidaron de mí, por último y no menos importante a Ximena, quién fue la que me dio aliento y motivación para culminar esta etapa de mi vida.

Bach. Iván Diego Rivera Meza

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente tesis pretende resolver el problema: ¿Cómo optimizar el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas?, cuyo objetivo es: Implementar un sistema de código de barra para optimizar el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas; para ello la hipótesis planteada es la siguiente: La implementación de un sistema de código de barras si optimiza el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas.

El tipo de metodología es aplicada, el nivel correlacional, el diseño experimental y la población son los 1162 asociados inscritos en el padrón general de la entidad conformado por 7 sectores (Pampa, Santa Cruz, 27 de mayo, Patac, Mancoculi, Unuimarca y Centenario), se trabajó con una muestra no aleatoria o dirigida el cual corresponde al sector Pampa con 362 asociados inscritos.

Como la conclusión se menciona que la implementación de un sistema de código de barras si optimizó el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas.

Palabras clave: Servicios de saneamiento, control de asistencia, sistema de escritorio, metodología XP.

ABSTRACT

This thesis aims to solve the problem: How to optimize the attendance control of the partners in the Administrative Board of Sanitation Services Quilcas ?, whose objective is to: Implement a bar code system to optimize the attendance control of the associates in the Administrative Board of Sanitation Services Quilcas; for this the hypothesis is the following: The implementation of a bar code system if it optimizes the attendance control of the associates in the Administrative Board of Sanitation Services Quilcas.

The type of methodology is applied, the correlation level, the experimental design and the population are the 1162 associates registered in the general register of the entity conformed by 7 sectors (Pampa, Santa Cruz, May 27, Patac, Mancoculi, Unuimarca and Centenario), we worked with a non-randomized or directed sample which corresponds to the Pampa sector with 362 registered members.

As the conclusion is mentioned that the implementation of a bar code system if optimized the attendance control of the partners in the Administrative Board of Sanitation Services Quilcas.

Keywords: Sanitation services, attendance control, desktop system, XP methodology.

INTRODUCCIÓN

La información hoy en día juega un papel muy importante en diversos entornos de la sociedad y para automatizar los procesos de dicha información, la tecnología es la herramienta por excelencia, si ambas interactúan mediante buenas prácticas el resultado es más que satisfactorio. En este caso se optimizó el control de asistencia a las asambleas mensuales de los asociados de la Junta Administradora de Servicios de Saneamiento Quilcas, así como la gestión de los mismos, se tendrá el padrón general, las asistencias y las inasistencias de manera segura y oportuna. Esta optimización se logró mediante el desarrollo e implementación de un sistema de escritorio con el uso del código de barras aplicando la metodología XP (eXtreme Programming) que ha permitido la flexibilidad, eficiencia y calidad del producto final para cumplir el objetivo principal que es optimizar el proceso de control de asistencias de los asociados de dicha entidad.

El trabajo de investigación se encuentra organizada en cinco capítulos, los mismo que se describen a continuación.

El capítulo I aborda el “Planteamiento del estudio”, se hace una breve descripción de la entidad, se detalla la situación problemática, los objetivos y la justificación del presente trabajo de investigación.

El capítulo II nos habla sobre el “Estado del Arte”, se describe el ámbito de la problemática mediante los avances de los mismos y se esboza los conceptos fundamentales en los que se basa la investigación

El capítulo III describe la “Metodología de la investigación”, se realiza una descripción acerca del planteamiento de la solución, se presenta la metodología XP (eXtreme Programming) analizando la viabilidad tanto técnica como económica.

El capítulo IV describe el “Desarrollo de la Investigación”, en este punto se abordará todo acerca del desarrollo del sistema de escritorio mediante los requerimientos, análisis, diseño del sistema y la codificación del mismo.

El capítulo V explica la “Discusión y Resultados”, finalmente se realizarán las conclusiones, recomendaciones, se considera los anexos que fueron útiles para desarrollo de la tesis.

Bach. Iván Diego Rivera Meza

CAPÍTULO I

PLANTEAMIENTO DEL ESTUDIO

1.1 Descripción de la organización

La Junta Administradora de Servicios y Saneamiento es una organización comunal constituida bajo la forma de Asociación Civil, con la finalidad de administrar, operar y mantener los servicios de agua y saneamiento de la comunidad, sector o anexo que están bajo su responsabilidad. De acuerdo a la ley General de Servicios de Saneamiento – 26338 y D.S. 023 – 2005, artículo 175º la JASS del distrito de Quilcas se registró en la municipalidad distrital y también presentó a Registros Públicos su acta de constitución, padrón de socios, estatuto y reglamento para tener personería jurídica. La Junta Administradora de Servicios de Saneamiento se conforma por la asamblea general, consejo directivo y los miembros del consejo directivo. La asamblea general son la máxima autoridad de la JASSQ, la conforman todos los asociados(as) inscritos en el padrón general, el consejo directivo es el grupo de personas elegidas por la asamblea general para representarlos y responsabilizarse de las diferentes funciones y competencias que demandan la

administración de la JASSQ y los miembros del consejo directivo son: presidente, secretario, tesorero, vocales y fiscales.

a) Misión

Brindar los servicios de agua potable, alcantarillado, drenaje y saneamiento a todos los asociados, garantizando el derecho de acceso a los mismos, buscando la sustentabilidad del recurso agua.

b) Visión

Ser un organismo trasparente y autosuficiente comprometido a garantizar agua potable, alcantarillado, drenaje y saneamiento del a los asociados ampliando la cobertura y preservando el entorno ecológico.

Fig. 1.1 Ubicación geográfica de la JASSQ

En la Fig. 1.1 se muestra la ubicación geográfica de la JASSQ (Junta Administradora de Servicios de Saneamiento Quilcas) como muestra de donde se realiza la investigación.

1.2 Situación problemática

La Junta Administradora de Servicios de Saneamiento Quilcas realiza diversas actividades tanto dentro como fuera de las oficinas, una de ellas es la de convocar a asambleas de carácter obligatorio ordinales a todos los asociados inscritos en el padrón general de la

entidad. Por tal motivo se realiza un control de asistencia, nombrando a un representante de cada uno de los siete sectores o barrios del distrito de Quilcas, luego se imprime siete padrones para cada vocal, esto implica un gasto de impresión innecesario, posteriormente el control se realiza de manera manual, generando demora e incomodidad a los asociados, al finalizar la asamblea cada vocal entrega al secretario la lista de asistencia, quién luego de contabilizar de forma manual elabora una lista de los inasistentes, la cual está expuesta a errores esto es perjudicial para la entidad ya que la multa es una forma de ingreso, finalmente la tesorera cobra las multas y generalmente le ocasiona problemas por los errores cometidos por el secretario al momento de elaborar la lista de inasistentes.

Adicionalmente se realizan otras tareas complementarias, que son los siguientes:

- Registro de nuevos asociados.
- Elaboración del padrón general de asociados.

Se pudo observar los problemas y sus causas respectivas siguientes:

Tabla 1

Problema - Causa

Problema	Causa
La impresión del padrón de asociados para gestionar la asistencia a las asambleas es engorrosa.	Se imprimen siete padrones, debido a los siete sectores del distrito de Quilcas.
Desorden y demora al momento de registrar la asistencia.	El registro se realiza tanto al ingresar como al salir.
La elaboración de la lista de inasistencias a para cobrar las multas está vulnerable a errores.	La lista se elabora manualmente luego de ser contabilizada.

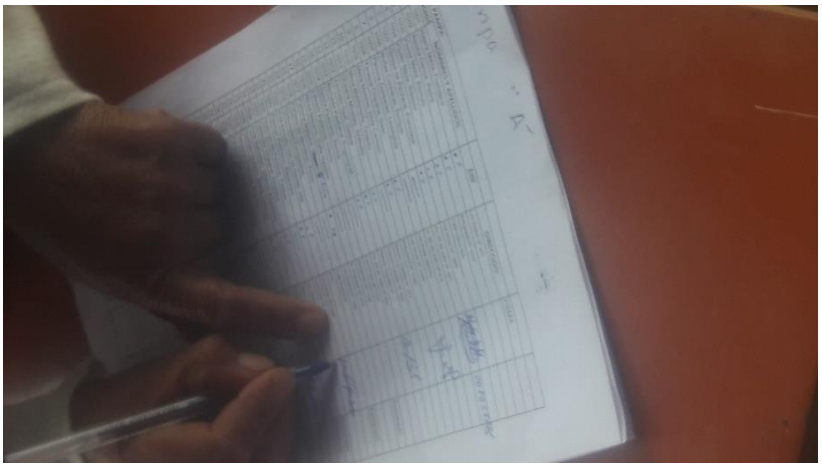


Fig. 1.2 Registro de asistencia

La Fig. 1.2 muestra como el asociado firma su asistencia.

1.3 Formulación del Problema

1.3.1 Problema General

¿Cómo optimizar el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas?

1.3.2 Problemas Específicos

a) ¿Cómo mejorar la administración del padrón general de asociados?
b) ¿De qué manera se puede agilizar el registro de asistencia de los asociados?
c) ¿Cómo se garantiza que la lista de inasistencia para el cobro de las multas sea confiable?

1.4 Objetivos

1.4.1 Objetivo General

Implementar un sistema de código de barras para optimizar el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas.

1.4.2 Objetivos Específicos

a) Diseñar una interfaz para administrar el padrón general de asociados.
b) Desarrollar un procedimiento para el registro de asistencias mediante el uso del lector de código de barras.
c) Generar la lista de inasistencia de manera automática para el cobro de las multas.

1.5 Justificación

1.5.1 Justificación práctica

La Junta Administradora de Servicios de Saneamiento Quilcas no cuenta con un sistema para el registro de asistencias; dicho proceso se realiza manualmente lo cual genera demora y es vulnerable a errores, por tanto, es necesaria el desarrollo e implementación de un sistema que registre las asistencias de manera óptima con ayuda del código de barras.

1.5.2 Justificación metodológica

La presente investigación hará uso de la metodología XP (eXtreme Programming), que se divide en tres fases dentro de las cuales se realizan varias iteraciones, en las que se realiza mayor o menor esfuerzo aplicando las buenas prácticas y disminuyendo el costo en cada etapa, teniendo como resultado un producto de calidad, al mismo tiempo que contribuirá a futuras investigaciones que aborden la misma metodología.

1.6 Delimitación

1.6.1 Espacial

El trabajo de investigación engloba la interacción de la junta directiva (presidente, secretario, tesorera, vocales) con los 1162 asociados, quiénes están inscritos en el padrón general y se dividen en 7 sectores (Pampa, Santa Cruz, 27 de mayo, Llacta, Patac, Unuimarca, Centenario) del distrito de Quilcas, provincia Huancayo, departamento Junín.

1.6.2 Temporal

El punto de inicio para el desarrollo del presente trabajo de investigación se tomará como referencia el plan de entrega del proyecto según la metodología XP (eXtreme Programming) y la duración está estimado para 4 meses (ver la tabla 23 del capítulo IV donde se detalla la delimitación temporal).

1.6.3 Económica

La inversión financiera para la elaboración del trabajo de investigación se puede visualizar en la tabla 2.

Tabla 2

Inversión total del proyecto

Concepto	Cantidad	Costo unitario	Total
Lectora de código de barras	2	S/. 180.00	S/. 360.00
Impresión de fichas.	1162	S/. 0.70	S/. 813.40
		TOTAL	**S/. 1,173.40**

CAPÍTULO II

MARCO TEÓRICO

2.1 Antecedentes

2.1.1 Antecedentes Internacionales

- En [1], trata sobre el funcionamiento del código de barras, el proceso comienza con un dispositivo que emite un rayo de luz directa sobre un código de barras. El dispositivo contiene un pequeño sensor que detecta la luz reflejada y la convierte en energía eléctrica. El resultado, es una señal eléctrica que puede ser interpretada y convertida en datos. Los códigos de barras son asignados localmente, pero son únicos a nivel mundial, se usa en más de 100 países, para diferentes aplicaciones. Esto se logra a través de la asignación de prefijos, por ejemplo, tiene información sobre un producto, persona, documento, etc. Esta información es reflejada cuando es detectada por una pistola de luz, este es leído por medio de una escala de luces y sombras la cual es almacenada y reflejada en una base de datos y a su vez la base de datos regresa la

información en una pantalla a otra forma de salida. Este procedimiento mediante el código de barras es sencillo y elemental para cualquier entidad que desea emplearlo. Esta descripción nos indica que los códigos de barra son usados en todo el mundo, lo cual nos garantiza el buen funcionamiento para aplicación en el presente trabajo de investigación.

- Según [2], nos explica acerca de los códigos de barra unidimensionales, estos códigos ocupan, en realidad, dos dimensiones; sin embargo, se llaman unidimensionales debido a que son leídos por un láser lineal y basta con leer cualquier línea transversal al código, dado que todas ellas iguales. No obstante, existen diferentes formas de llevar a cabo esta codificación. En general, la elección de un determinado tipo de código de barras depende de la aplicación para la que se desee utilizar. Los códigos de barras suelen tener representaciones: una determinada cantidad de dígitos o caracteres por unas barras verticales de diferente grosor y de caracteres, es decir, un conjunto específico de letras, números y símbolos. Todo código de barras posee unos elementos característicos. Así los separadores de inicio y de fin de cada código son combinaciones específicas de barras y espacios que indican al lector óptico dónde empieza y dónde termina el código a leer. Entre los tipos de código conocidos se tiene el código 39 es un código de longitud variable, adecuado para codificar datos alfanuméricos de carácter general, el código 2 de 5 es un sistema de codificación de propósito general para datos numéricos, el código Codabar sirve para la codificación de números incluye algunos caracteres especiales y el código 128 que es más

condensando que el código 39, permite codificar 128 caracteres ASCII. Este alcance que nos da este artículo, nos permite entender como es un código de barra unidimensional para poder emplearlo.

- En [3], llamado: “Implementación del Sistema de Código de Barras en el Sector Público: en la eficiencia de la optimización de recursos”, trata sobre los beneficios y retos de la utilización del código de barras y otros estándares GS1 para aumentar la competitividad en el sector público de Costa Rica y las organizaciones de la economía social, por medio de la implementación de los estándares internacionales y mejores prácticas, permitiendo reducir costos y aportando mayor valor a todos los participantes de la cadena de abastecimiento. Hace 40 años, los líderes se de la industria se reunieron para crear una forma única para la identificación, naciendo así lo que hoy día se conoce como Código de Barras, la cual se viene utilizando de manera eficiente hoy en día a nivel mundial. En el territorio costarricense se realizan 7000 juntas receptoras de votos lo cual son enviados en paquetes en donde son contabilizados y anotaban la hora de ingreso uno a uno por el personal, lo cual hacía engorroso el trabajo, ya que era manual. A partir de esto se buscó la solución para poder automatizar dicho proceso y de esta forma agilizar la salida y recibo del material electoral, en donde la Asociación GS1 le asesoró acerca de las soluciones disponibles, se evaluó la factibilidad y la solución se encontró en el uso de un chip llamado Código Electrónico de Productos “EPC” que captura la información vía radiofrecuencia.

- En el Sector Salud se almacena y distribuye insumos médicos y medicamentos a través de los 3 centros de distribución que posee, su distribución es aproximadamente ¢128.000.000.000 por año, tienen un inventario permanente de ¢36.000.000.000 y presupuesto de operación de¢3.900.000.000. Actualmente los insumos y medicamentos se reciben codificados, identifican unidad de consumo, unidad de distribución y han llegado hasta el GS1-128, el cual les permite manejar números de lote, fechas de vencimiento y números de orden de compra y su objetivo es llegar al GS1 Datamatrix para luego llegar al consultorio médico y los pacientes. Dicha aplicación en 2 sectores públicos demuestra que no importa la complejidad ni la cantidad que se maneje, los códigos de barra son eficientes, por ende, ayudará en la solución a nuestro problema general.

- En [4], trata sobre: “El código de Barras y su Aplicación en la Bibliotecas Universitarias de la Ciudad de Guatemala”, y explica que la identificación automática actual, un código de barras identifica en forma única en cualquier producto en Guatemala y en todo el mundo, por tan razón es importante e indispensable su aplicación. Por ello se propone una guía para implementar el código de barras en la biblioteca, con el objetivo de proporcionar una herramienta ágil. El concepto básico de código de barras según Roger C. Palmer es “la tecnología automática de identificación, la cual codifica la información en orden, variando rectángulos gruesos paralelos de barras y espacios”. Según los antecedentes de la codificación: la necesidad de producir y administrar información eficientemente ha llevado al hombre a desarrollar

sistemas cada vez más sofisticados. Uno de los grandes avances logrados en este campo ha sido el procesamiento y almacenamiento electrónico de datos a través de computadoras. La importancia del sistema de Código de Barras es que ofrece altas velocidades y exactitudes en la captura de datos, reduce considerablemente el rango de error en los datos, aproximadamente el error ocurre por cada millón de caracteres, leídos por un scanner lo que indica la alta seguridad en la información. Al momento de entrevistarse con los usuarios finales (Bibliotecarios), estos respondieron la demora al momento de registrar un préstamo, otros indicaron que necesitan una solución automatizada y otros desconocían de esta tecnología. Para la aplicación de los códigos de barra es necesario saber: Cuanto material se tiene disponible para trabajar, evaluar qué tipo de código se debe usar, asignar un número único a cada documento. Luego de eso se realiza lo siguiente: cotizar precios, preparar un presupuesto estimado de costos y tiempo para su implementación, se compra equipo material, se imprimen las etiquetas, colocar el código de barras en los libros para su circulación y préstamo. Requerimientos mínimos de equipo: Impresora, Pistolas Láser, Computadora, Software. Esta tesis nos brinda los pasos para poner en marcha el sistema que se desarrollará una vez terminado el producto.

- En [5], llamada: “Propuesta de Diseño de Implementación del Sistema de Código de Barras en el Departamento de Registros Médicos y Servicios de Apoyo al Diagnóstico en el Hospital San Juan de Dios”, la cual en su introducción

aborda sobre la tecnología y el equipamiento para la utilización de códigos de barras en los diferentes procesos hospitalarios, ya ha alcanzado el desarrollo suficiente para demostrar los beneficios y aplicaciones en el uso de la identificación de los pacientes, el etiquetado de exámenes, de envases unitarios de medicamentos; siendo lo suficientemente flexibles como para manejar todo tipo de simbologías, donde no hay riesgo de que este sistema implementado se torne obsoleto en un futuro inmediato. El objetivo general es: Elaborar una propuesta de diseño para implementar el sistema de código de barras en el Departamento de Registros Médicos y Servicios de Apoyo al Diagnóstico en el Hospital San Juan de Dios. Dentro del proceso de generación del código de barras la información básica que se incluirá será: nombre del hospital, número de cédula (expediente), nombre completo, fecha de nacimiento, género, especialidad, código y nombre del médico, calidad de asegurado, fecha y hora de atención, nombre completo del funcionario. Esta información se puede completar de acuerdo a las necesidades que presente el paciente y el servicio donde se atiende. Por otro lado, el implementar el código de barras mejorará la exactitud de la identificación de todos los documentos que se le dan apaciente durante su atención. Con ello, el Hospital se beneficiará enormemente al eliminarse el error humano. La implementación de la herramienta Código de Barras deberá funcionar basándose en la plataforma tecnológica de hardware y software existente en el Hospital San Juan de Dios. Una de sus conclusiones que se puede rescatar es: A través del Sistema de Código de Barras se lograría optimizar la trazabilidad del Expediente Médico, reducir el número de exámenes de laboratorio y gabinete que

anualmente se repiten por no poder anexarse al expediente, mejorar la oportunidad en la atención del usuario externo, y optimizar las funciones asignadas a los funcionarios del Departamento de Registros Médicos. Esta tesina nos describe la optimización en su proceso de identificación de pacientes, lo cual es una garantía de solución para nuestro trabajo de investigación.

2.1.2 Antecedentes Nacionales

- En [6], abarca sobre las “Metodologías Ágiles”, el cual en su introducción explica lo siguiente: Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual nos provee de una dirección a seguir para la correcta aplicación de los demás elementos. Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir de las buenas prácticas de la Ingeniería del Software, asumiendo el riesgo que ello conlleva. En este contexto, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico. Por estar especialmente orientadas para proyectos pequeños, las Metodologías Ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto. La historia de las Metodologías Ágiles y su apreciación como tales en la comunidad de la Ingeniería de Software tiene sus inicios en la creación de una delas metodologías utilizada como arquetipo: XP - eXtreme

Programming, que nace dela mente de Kent Beck, tomando ideas recopiladas junto a Ward Cunningham. Es así como que este tipo de metodologías fueron inicialmente llamadas “metodologías livianas”, sin embargo, aún no contaban con una aprobación pues se reconsideraba por muchos desarrolladores como meramente intuitiva. Luego, con el pasar de los años, en febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace formalmente el término “ágil” aplicado al desarrollo de software. Esta introducción sobre las metodologías ágiles, nos da a conocer la flexibilidad y eso ayuda a porqué se eligió ésta y no la tradicional para nuestro trabajo de investigación.

- En [7], que lleva de título: “Aplicativo móvil para la Asistencia de Pacientes con Alzheimer en su fase inicial”, aplican la metodología ágil XP, El presente proyecto consiste en la creación de una aplicación móvil, implementada en la plataforma Android, que dé soporte a las actividades diarias de pacientes con fase inicial de Alzheimer. El aplicativo permite mostrar la ubicación de los pacientes, así como una agenda diaria a realizar y una parte de entrenamiento. Para el desarrollo del aplicativo móvil, se usó la metodología Xtreme Programming (XP), esto debido al limitado tiempo y la facilidad de las prácticas en el desarrollo de aplicaciones. Como resultado, se implementó el aplicativo móvil, que es capaz de mostrar la ubicación del paciente a su familiar o cuidador vía Global Positioning System (Sistema de posicionamiento global) y brindar apoyo por medio de una agenda personalizada a personas que sufren de la enfermedad de Alzheimer en actividades diarias, de tal

modo que puedan tener mayor independencia. El aplicativo móvil cuenta, además, con un módulo de entrenamiento mediante el cual se pueden visualizar fotos que permiten recordar objetos o personas. El proyecto permite concluir adecuadamente las fases de desarrollo de software con sus respectivos entregables, demostrar la importancia de la tecnología para la solución de problemas en beneficio de la sociedad. Específicamente, para mejorar la calidad de vida de pacientes con Alzheimer. La metodología Xtreme Programming (XP), el cual se aplicará en el presente trabajo de investigación, al igual que la tesis presentada, contamos con un tiempo limitado y a su vez es un proyecto de menor cuantía, pero que solucionará el problema planteado a nuestro trabajo.

- En [8], titulada: “Eficiencia Operativa De La Gestión De Planillas Mediante El Software Praxis-GL En La Municipalidad Provincial De Concepción”, pretende mejorar el control de la información relacionada a las planillas, nóminas y acciones del personal relacionadas a la unidad de Personal. Las causas que originan la entropía en gestión de planillas son: controles de información y planillas manuales poco confiables, duplicidad de procesos, duplicidad de datos y poco control de acceso a la información que se maneja en la Unidad de Personal. Estos problemas ocasionan un retraso para el cumplimiento de pagos al personal, SUNAT, fondos de pensiones y proveedores. La metodología usada para la elaboración de este proyecto es la Programación Extrema(XP). Es una metodología de desarrollo ligero (o ágil) basada en una serie de valores y de prácticas de buenas maneras que persigue el objetivo de aumentar la

productividad a la hora de desarrollar programas. Este modelo de programación se basa en una serie de metodologías de desarrollo de software en la que se da prioridad a los trabajos que dan un resultado directo y que reducen la burocracia que hay alrededor de la programación. Una de las características principales de este método de programación, es que sus ingredientes son conocidos desde el principio de la informática. Los autores de XP han seleccionado aquellos que han considerado mejores y han profundizado en sus relaciones y en cómo se refuerzan los unos con los otros. El objetivo que se perseguía en el momento de crear esta metodología era la búsqueda de un método que hiciera que los desarrollos fueran más sencillos. Aplicando el sentido común. La metodología XP fue aplicada en tres etapas: Gestión del Proyecto. - Esta es la planificación de historias que se realizó desde el inicio del proyecto, tras estudiar el proyecto y mantener conversaciones con el cliente. De esta redacción inicial de historias de usuario se realizó una planificación inicial y posteriormente fue cambiada a lo largo del proyecto. Algunas de estas historias fueron eliminadas o cambiadas a lo largo del proyecto, a medida que cambiaban los requisitos del cliente o se tenía una concepción más clara del proyecto. Implementación. - Uno de los puntos centrales, es el de la base de datos, donde se describió el modelo de datos al final del desarrollo de la aplicación (ya que fueron muchos y requirieron una reestructuración del código de las diferentes historias dada la relación establecida de la base de datos con el código de la aplicación). Se realizaron diferentes prototipos de interfaz de usuario que fueron desarrollados con la aprobación del cliente. Finalmente se produjo el código fuente de la aplicación.

Pruebas. - Se realizaron las pruebas funcionales de la aplicación y se describió el modo de utilización y los posibles estados de error que pueden darse, así como los mensajes de aviso/error/confirmación que debe emitir la aplicación en estos casos. Como resultado de la aplicación práctica de esta metodología se logró la implementación de este proyecto que mejora la gestión de planillas y a su vez permite brindar un mejor control de la información referente al empleado. Se recomienda la aplicación de esta metodología en proyectos pequeños a corto plazo. Este proyecto de tesis nos recomienda que la metodología XP para proyectos pequeños, por tanto, es idóneo para el trabajo a realizar.

- En [9], titulada: “Implementación de Software para el Registro y Procesamiento de Atenciones de Salud en las Actividades de Responsabilidad Social – Caso Mina Cori huarmi”, se desarrolla en el ámbito de la actividad minera en el país. La Mina Cori huarmi desarrolla las actividades de Responsabilidad Social Empresarial mediante la Oficina de Relaciones Comunitarias que, entre otras, cuenta con un área de Salud con las especialidades de: Medicina General, Enfermería, Obstetricia y Odontología; las mismas que prestan sus servicios a la población de la zona de influencia del proyecto minero de manera gratuita. Las atenciones médicas generan unos registros físicos que son muy difíciles de procesar por la redundancia de información y requerir largos periodos de tiempo por no contar con un mecanismo computarizado. El problema descrito es abarcado con la implementación de un software para el registro y procesamiento de la información utilizando la metodología de desarrollo ágil

Extreme Programan, y herramientas tecnológicas de desarrollo de software libre, ya que otros mecanismos computarizados para la mejora de la situación, como la utilización de herramientas ofimáticas, no resultan del todo satisfactorias o presentan dificultades operativas. La metodología Extreme Programming adoptada se enfoca en la satisfacción del usuario, poniendo poco énfasis a una documentación rigurosa utilizada en otras metodologías, además de una comunicación constante con el cliente, lo cual hizo que se pueda obtener información de la organización de primera mano y en todo momento. De esta manera se logró el desarrollo del software, la escritura del código, modificaciones a la misma, así como las pruebas de aceptación. Después de la implementación del software se resolvieron los problemas de redundancia de información y se disminuyeron considerablemente los lapsos de tiempo en el procesamiento y la consolidación de datos, de este modo quedó demostrado que la alternativa adoptada ofrece una solución viable al problema planteado. La tesis presentada uso la metodología XP para enfocarse más en el cliente que en la documentación, el cual nos ayuda, ya que hacer mucha documentación nos retrasaría para poder desarrollar e implementar la solución presentada.

- En [10], titulada: “Sistema de Información basado en las recomendaciones para mejorar el rendimiento del cultivo de papa en la región Puno aplicando la metodología ágil XP”, este trabajo aplico esta metodología ya que en un una de sus conclusiones nos dice lo siguiente: “La metodología de desarrollo ágil implementada en el siguiente proyecto, ha sido de gran utilidad y ha cubierto

de manera efectiva todos los requerimientos planteados por el cliente, así mismo al ser una metodología enfocada en la programación, más que en la documentación, se ha podido terminar en corto plazo, el tiempo de desarrollo que conllevó el trabajo, también se logró un elevado porcentaje de usabilidad por, donde se observó un 95% en el grado de aceptación del sistema por parte del usuario". Asimismo, en una de sus recomendaciones los dice: "La metodología de desarrollo ágil usada para el siguiente proyecto, ya no sería recomendable en caso se desee dar una orientación más amplia del trabajo, ya que sería de vital importancia la documentación y seguimiento exhaustivo del proyecto, características con las que XP no profundiza dándole prioridad a la programación, otra alternativa sería integrar XP, a otra metodología de desarrollo como RUP y Scrum, que se orientan más a la gestión y documentación del sistema. Como se mencionó anteriormente XP, es idóneo para el tipo de trabajo que se llevará a cabo.

2.2 Bases Teóricas

2.2.1 Código de barras

En [11] indica que es muy empleado en supermercados y almacenes, en donde los artículos lo traen incorporado de tal modo que sea fácil pasar un dispositivo que permita leerlo y después de decodificarlo mandarlo a alguna computadora.

Existen varios sistemas de código de barras, todos ellos están basados en la diferencia que entre la cantidad de luz reflejada sobre partes blancas y partes negras de un impreso con segmentos paralelos. Un sistema de barras es "el dos de cinco". Cada dígito es representado por cinco barras, dos de las cuales son más anchas que las otras tres. Una barra

delgada es un dígito binario 0 y una gruesa es un 1. La barra gruesa tiene una anchura tres veces mayor que la delgada. Siempre deben aparecer dos barras anchas como medida de protección para evitar errores. La posición de las líneas anchas determina el dígito.

2.2.2 Metodología ágil

Según [12] marzo de 2001 diecisiete críticos de los modelos de mejora del desarrollo de software basados en procesos, convocados por Kent Beck, quien había publicado un par de años antes Extreme Programming Explained, libro en el que ponía una nueva metodología denominada Extreme Programming, se reunieron en Salt Lake City para tratar sobre técnicas y procesos para desarrollar software. En la reunión se acuñó el término “Métodos Ágiles” para definir a los métodos que estaba surgiendo como alternativa a las metodologías formales (CMMI, SPICE) a las que consideraban excesivamente “pesadas” y rígidas por su carácter normativo y fuerte dependencia de planificaciones detallas previas al desarrollo.

A continuación, se describen con mayor nivel de especificación las principales semejanzas y diferencias de las metodologías ágiles y las metodologías pesadas:

Prácticas de desarrollo: En las metodologías de desarrollo ágil se procura realizar procesos de software de acuerdo a las prácticas que le han dado resultados al grupo. En las metodologías pesadas se desarrolla de acuerdo a las normas sugeridas por los estándares de desarrollo.

Adaptación al cambio: En las metodologías ágiles por la misma capacidad de reacción son más adaptables a los cambios, por el contrario, en las metodologías pesadas por el nivel de formalidad en la fase de requerimientos son más resistentes al cambio.

Controlo: Por su capacidad de adaptación el proceso se hace menos controlado que en las metodologías tradicionales que ejercen mayor control en el proceso por su nivel de formalización.

Documentación: En las metodologías ágiles no se hace énfasis en la documentación ni en los artefactos a diferencia de las metodologías pesadas.

Equipo de trabajo: En las metodologías ágiles existen bajo número de participantes y roles.

2.2.3 Metodología XP

Según [20], es metodología ligera de desarrollo de aplicaciones que se basa en la simplicidad, la comunicación y la realimentación del código desarrollado.

Objetivos de XP

- La satisfacción del cliente.
- Potenciar el trabajo en grupo.
- Minimizar el riesgo actuando sobre las variables del proyecto: costo, tiempo, calidad, alcance.

Características

- Metodología basada en prueba y error para obtener un software que funcione realmente.
- Fundamentos en principios
- Está orientada hacia quien produce y usa software (el cliente participa muy activamente).
- Reduce el coste del cambio en todas las etapas del ciclo de vida del sistema
- Combina las que han demostrado ser las mejores prácticas para desarrollar software y las lleva al extremo.
- Los requisitos pueden cambiar.
- Grupo pequeño y muy integrado (2-12 personas).

Las historias de usuario representan una breve descripción del comportamiento del sistema, se realizan por cada característica principal del sistema y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos, así mismo reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

Cada historia de usuario debe ser lo suficientemente compresible y delimitada para que los programadores puedan implementarlas en unas semanas.

La plantilla a utilizarse para elaboración de las historias de usuario se muestra en la tabla 3 y cada uno de sus componentes se explica a continuación.

Tabla 3

Plantilla para las historias de usuario

<table>
<tr><th colspan="2">HISTORIA DE USUARIO</th></tr>
<tr><td>Número: Permite identificar a una historia de usuario.</td><td>Usuario: Persona que utilizará la funcionalidad del sistema descrita en la historia de usuario.</td></tr>
<tr><td colspan="2">Nombre Historia: Describe de manera general a una historia de usuario.</td></tr>
<tr><td>Prioridad en Negocio: Grado de importancia que el cliente asigna a una historia de usuario.</td><td>Riesgo en Desarrollo: Valor de complejidad que una historia de usuario representa al equipo de desarrollo.</td></tr>
<tr><td>Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una historia de usuario.</td><td>Iteración Asignada: Número de iteración, en que el cliente desea que se implemente una historia de usuario.</td></tr>
<tr><td colspan="2">Programador Responsable: Persona encargada de programar cada historia de usuario</td></tr>
<tr><td colspan="2">Descripción: Información detallada de una historia de usuario.</td></tr>
<tr><td colspan="2">Observaciones: Campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.</td></tr>
</table>

Una historia de usuario se descompone en varias tareas de ingeniería, las cuales describen las actividades que se realizarán en cada historia de usuario, así mismo las tareas de ingeniería se vinculan más al desarrollador, ya que permite tener un acercamiento con el código. (Ferreira Escutia, 2013). La plantilla a utilizarse para la elaboración de las tareas de ingeniería se muestra en la tabla 4 y cada uno de sus componentes

Tabla 4

Plantilla para las tareas de ingeniería

<table>
<tr><th colspan="2">TAREA DE INGENIERÍA</th></tr>
<tr><td>Número de Tarea: Permite identificar a una tarea de ingeniería.</td><td>Número de Historia: Número asignado de la historia correspondiente.</td></tr>
<tr><td colspan="2">Nombre de Tarea: Describe de manera general a una tarea de ingeniería.</td></tr>
<tr><td>Tipo de Tarea: Tipo al que corresponde la tarea de ingeniería.</td><td>Puntos Estimados: Número de días que se necesitará para el desarrollo de una tarea de ingeniería.</td></tr>
<tr><td>Fecha Inicio: Fecha inicial de la creación de la tarea de ingeniería.</td><td>Fecha Fin: Final concluida de la tarea de ingeniería.</td></tr>
<tr><td colspan="2">Programador Responsable: Persona encargada de programar la tarea de ingeniería.</td></tr>
<tr><td colspan="2">Descripción: Información detallada de la tarea de ingeniería.</td></tr>
</table>

las pruebas de aceptación son de vital importancia para el éxito de una iteración y el comienzo de la siguiente, con lo cual el cliente puede conocer el avance en el desarrollo del sistema y a los programadores lo que les resta por hacer. Además, permite una retroalimentación para el desarrollo de las próximas historias de usuarios a ser entregadas. Estas son comúnmente llamadas pruebas del cliente, por lo que son realizadas por el encargado de verificar si las historias de

usuarios de cada iteración cumplen con la funcionalidad esperada.

La Plantilla a utilizarse para la elaboración de las pruebas de aceptación se muestra en la tabla 5 y a continuación se definen cada uno de los componentes.

Tabla 5

Plantilla para las pruebas de aceptación

PRUEBAS DE ACEPTACIÓN	
Código: Nº Único, permite identificar la prueba de aceptación.	**Nº Historia de Usuario:** Número único que identifica a la historia de usuario.
Historia de Usuario: Nombre que indica de manera general la descripción de la historia de usuario.	
Condiciones de Ejecución: Condiciones previas que deben cumplirse para realizar la prueba de aceptación.	
Entrada/Pasos de Ejecución: Pasos que siguen los usuarios para probar la funcionalidad de la historia de usuario.	
Resultado Esperado: Respuesta del sistema que el cliente espera, después de haber ejecutado una funcionalidad	
Evaluación de la Prueba: Nivel de satisfacción del cliente sobre la respuesta del sistema. Los niveles son: Aprobada y No Aprobada.	

La propuesta original de Beck incluye los siguientes roles:

- Programador: Es el responsable de implementar las historias de usuario por el cliente. Además, estima el tiempo de desarrollo de cada historia de usuario para que el cliente pueda asignarle prioridad dentro de la iteración. Cada iteración incorpora nueva funcionalidad de acuerdo a las prioridades establecidas por el cliente. El programador también es responsable de diseñar y ejecutar los test de unidad del código que ha implementado o modificado.
- Cliente: Determina la funcionalidad que se pretende en cada iteración y define las prioridades de implementación según el

valor de negocio que aporta cada historia. El cliente también es responsable de diseñar y ejecutar los test de aceptación.

- Encargado de pruebas (tester): Es el encargado de ejecutar las pruebas regularmente, difunde los resultados dentro del equipo y es también el responsable de las herramientas de soporte para pruebas.

- Encargado de seguimiento (tracker): Una de las tareas más importante del tracker, consiste en seguir la evolución de las estimaciones realizadas por los programadores y compararlas con el tiempo de desarrollo. De esta forma, puede brindar información estadística en lo que refiere a la calidad de las estimaciones para que puedan ser mejoradas.

- Entrenador (coach): Es el responsable del proceso en general. Se encarga de iniciar y de guiar a las personas del equipo en poner en marcha cada una de las prácticas de la metodología XP.

- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo parta resolver un problema específico.

- Gestor (big boss): Es el vínculo entre el cliente y programadores. Experto en tecnología y labores de gestión. Construye el plantel del equipo, obtiene los recursos necesarios y maneja los problemas que se generan. Administra a su vez las reuniones (planes de iteración, agenda de compromisos, etc.). Su labor fundamental es de coordinación.

La programación extrema consta de 4 fases, las cuales son:

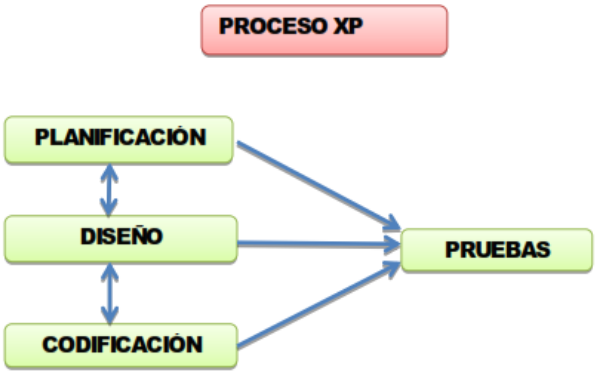


Fig. 2.1 Proceso de XP

La Fig. 2.1 muestra las 4 fases en el que se divide el proceso de la metodología XP.

Planificación

La Metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Los Conceptos básicos de la planificación son:

- Las Historias de Usuarios, las cuales son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar.

- El Plan de Entregas (Release Plan), establece que las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto.

- Plan de iteraciones (Iteration Plan), las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.

- Reuniones Diarias de Seguimiento (Stand – Up Meeting), el objetivo es mantener la comunicación entre el equipo y compartir problemas y soluciones.

Diseño

La Metodología La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

- Simplicidad: Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione.

- Recodificación (Refactoring): Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de creerlo más simple, conciso y entendible. Las metodologías de XP sugieren re codificar cada vez que sea necesario.

- Metáforas: XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, así como guiar la estructura del mismo. Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Codificación

La Disponibilidad del Cliente, Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El Involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto con la metodología XP. Al comienzo del proyecto, el este debe proporcionar las historias de usuarios. Pero, dado que estas historias son expresamente cortas y de “alto nivel”, no contienen los detalles necesarios para realizar el desarrollo del código. Estos detalles deben ser proporcionados por el cliente, y discutidos con los desarrolladores, durante la etapa de desarrollo.

Uso de Estándares**,** XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite el re codificación.

Programación Dirigida por las Pruebas (“Test-Driven Programming”), En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada sobre el final del proyecto, o el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, primero se escribe los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a los que se refiere esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona o “dirige” el desarrollo.

Programación en Pares, XP propone que se desarrolle en pares de programadores, ambos trabajando juntos en un mismo ordenador. Si bien parece que ésta práctica duplica el tiempo asignado al proyecto (y por ende, los costos en recursos humanos), al trabajar en pares se minimizan los errores y se logran mejores diseños,

compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.

Integraciones Permanentes, Todos los desarrolladores necesitan trabajar siempre con la “última versión”. Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores. Idealmente, todos los días deben existir nuevas versiones publicadas. Para evitar errores, solo una pareja de desarrolladores puede integrar su código a la vez.

Propiedad Colectiva del Código, En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, una pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o re codificar.

Ritmo Sostenido, La Metodología XP indica que debe llevarse un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es planificar el trabajo de forma a mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Pruebas

Pruebas Unitarias, Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código (“Test-Driven Programming”). Que todo código liberado pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código.

Detección y Corrección de Errores, Cuando se encuentra un error (“Bug”), éste debe ser corregido inmediatamente, y se

deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto.

Pruebas de Aceptación, Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación.

2.2.4 Plataforma Microsoft .NET

Según [13] En el año 2000 Microsoft presentó la plataforma .NET, con el objetivo de hacer frente a las nuevas tendencias de la industria del software y a la dura competencia de las plataformas Java de Sun.

.NET es una plataforma para el desarrollo de aplicaciones que integra múltiples tecnologías que han ido apareciendo en los últimos años como ASP.NET, ADO.NET, LINQ, WPF, Silverlight, etc. junto con el potente entorno integrado de Visual Studio, que permite desarrollar múltiple tipo de aplicaciones. Microsoft sólo ofrece soporte .NET para sistemas operativos Windows y las nuevas generaciones de dispositivos móviles. Respecto al resto de plataformas, el proyecto Mono (llevado a cabo por la empresa Novell) ha creado una implementación de código abierto de .NET, que actualmente te ofrece soporte completo para Linux y Windows y soporte parcial para otros sistemas operativos como por ejemplo MacOS.

Los elementos principales de la plataforma .NET son:

- .NET Framework: Es el núcleo de la plataforma y ofrece la infraestructura necesaria para desarrollar y ejecutar aplicaciones .NET
- Visual Studio y Microsoft Expression: Conforman el entorno de desarrollo de Microsoft, que permite desarrollar cualquier tipo de aplicación .NET (ya sea de escritorio, web, para dispositivos móviles, etc.) En Visual Studio el programador puede elegir indistintamente entre diversos lenguajes como C# o Visual Basic .NET y en todos ellos puede hacer exactamente lo mismo, con lo que a menudo la elección es simplemente debida a las preferencias personales de cada programador.

2.2.5 Lenguaje C#

En [13], la mejor forma de empezar a aprender un lenguaje de programación nuevo suele ser analizando algún ejemplo de código sencillo.

Analicemos los elementos del programa anterior:

- Definición del namespace: Todas las claves están incluidas dentro de un espacio de nombres concreto, que se indica dentro del código fuente mediante la instrucción “namespace”.
- Definición de la clase: La forma de definir una clase es con la palabra clave “class”, seguida del nombre que queremos dar a la clase y todos los elementos e instrucciones que pertenecen a la definición de la clase entre llaves. Cabe destacar que en C# todo son clases, con lo que todas las líneas de código deben estar asociadas a alguna clase.

- El método main: Punto de entrada a la aplicación, por donde empieza sui ejecución. Puede recibir una array llamado args, que contiene los parámetros pasados al ejecutable al lanzar su ejecución.

El lenguaje C# incorpora las siguientes sentencias de control de flujo:

En primer lugar, recordemos que los tipos de datos que pasemos como parámetro puede ser tipos valor

- if: Sentencia condicional. Un ejemplo típico sería el siguiente:

```
if (x <= 10)
{
   // Sentencias a ejecutar si la condición es cierta
}
else
{
   // Sentencias a ejecutar si la condición es falsa
}
```

Fig. 2.2 Sentencia condicional if

La figura Fig. 2.2 muestra un ejemplo de la sentencia if.

- switch: La situación switch es una forma específica de instrucción condicional, en la que se evalúa una variable y en función de su valor, se ejecuta un bloque u otro de instrucciones. Ejemplo:

```
switch (var)
{
   case 1:          // sentencias a ejecutar si es 1
                    break;

   case 2:          // sentencias a ejecutar si es 2
                    break;
   ...

   default:         // sentencias a ejecutar
                    // en caso de que ninguna de las
                    // condiciones "case" se cumplan
                    break;
}
```

Fig. 2.3 Sentencia condicional switch

La figura Fig. 2.3 muestra un ejemplo de la sentencia switch.

- for: Permite ejecutar un bosque de código un cierto número de veces. El siguiente ejemplo ejecuta un bloque de instrucciones 10 veces:

```
for (int i = 0; i<10; i++)
{
   // sentencias a ejecutar
}
```

Fig. 2.4 Bucle for

La figura Fig. 2.4 muestra un ejemplo del bucle for.

- while: El siguiente ejemplo es equivalente al anterior del for:

```
int i = 0;
while (i<10)
{
   // sentencias a ejecutar
   i++;
}
```

Fig. 2.5 Bucle while

La figura Fig. 2.5 muestra un ejemplo del bucle while.

- do – while: Igual que el anterior, excepto que la condición se evalúa al final. La diferencia fundamental es que, mientras en un bucle for o while, si la condición es falsa de entrada, no se ejecuta ninguna iteración, en un bucle do – while siempre se ejecuta como mínimo una iteración.

```
do
{
   // sentencias a ejecutar
}
while (condición);
```

Fig. 2.6 Bucle do - while

La figura Fig. 2.6 muestra un ejemplo del bucle do - while.

- foreach: Permite recorrer todos los elementos de una colección desde el primero al último. La sintaxis es la siguiente:

```
foreach (tipo nombre_variable in colección)
{
   // sentencias a ejecutar
}
```

Fig. 2.7 Constructor foreach

La figura Fig. 2.7 muestra un ejemplo del constructor foreach.

- Un ejemplo de colección son los arrays.

```
int [] nums = new int [] { 4,2,5,7,3,7,8 };
foreach (int i in nums)
{
   Console.WriteLine (i);
}
```

Fig. 2.8 Colección de arrays

La figura Fig. 2.8 es un ejemplo de colección de arrays.

2.2.6 Programación orientada a objetos

En [14], según Grady Booch, es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia. Los programas orientados a objetos constan de objetos que se comunican entre sí a través de mensajes.

- Clase: Conjunto de objetos que comparten características esenciales comunes tales como propiedades, métodos, se pueden agrupar en una clase respectiva.
- Métodos: Los métodos (operaciones o servicios) describen, el comportamiento asociado a un objeto, representan las acciones que pueden realizarse por un

objeto. La ejecución de un método puede conducir a cambiar el estado del objeto o dato local del objeto.

- Herencia: La herencia es un mecanismo por medio del cual una clase puede heredar las propiedades de otra. Asimismo, permite que se construya una jerarquía de clases que se extiende desde lo más general a lo más específico. En otras palabras, una clase derivada hereda propiedades y métodos de la clase base; permitiendo que la clase derivada reutilice la funcionalidad de la clase base.
- Sobrecarga: La sobrecarga, es una propiedad que describe una característica adecuada que utiliza el mismo nombre de operación para representar operaciones similares que se comportan de manera diferente cuando se aplican a clases diferentes.

2.2.7 Programación por capas

En [14], el estilo arquitectural en n capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan. Cuanto más se aumenta el proceso operativo de la empresa, las necesidades de proceso crecen hasta desbordar las máquinas. Es por ello que se separa la estructura de un programa en varias capas.

En adición a lo citado, podemos decir actualmente la programación por capas es un estilo de programación en la que el objetivo principal es separar la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de negocios y ésta a su vez de la capa de presentación al usuario.

El diseño que actualmente más se utiliza es el diseño en tres capas; sin embargo, la programación puede desglosarse en más capas, tal cual se presenta en el ejemplo que veremos más adelante.

Tipos de capas

- Capa de presentación

  Es la responsable de la presentación visual de la aplicación. La capa de presentación enviará mensajes a los objetos de esta capa de negocios o intermedia, la cual o bien responderá entonces directamente o mantendrá un diálogo con la capa de la base de datos, la cual proporcionará los datos que se mandarían como respuesta a la capa de presentación.

  Podemos decir que es la que se presenta al usuario, llamada también formulario o interfaz de presentación, esta captura los datos del usuario, ya sea de almacenaje, edición, o de recuperación de la información para la consulta respectiva.

- Capa de negocio

  Es la responsable de procesamiento que tiene lugar en la aplicación. Por ejemplo, en una aplicación bancaria el código de la capa de presentación se relacionaría simplemente con la monitorización de sucesos y con el envío de datos a la capa de procesamiento. Esta capa intermedia contendría los objetos que corresponden con las entidades de la aplicación. Esta capa intermedia es la que conlleva capacidad de mantenimiento y de reutilización.

  Contendrá objetos definidos por clases reutilizables que se pueden usar una y otra vez en otras aplicaciones. Estos objetos se suelen llamar objetos de

negocios y son los que contienen la gama normal de constructores, métodos para establecer y obtener variables, métodos que llevan a cabo cálculos y métodos normalmente privados, en comunicación con la capa de base de datos.

Es en esta capa donde se reciben los requerimientos del usuario y se envían las respuestas tras el proceso, a requerimiento de la capa de presentación. Se denomina capa de negocio o lógica del negocio, es aquí donde se establecen todas las reglas que deben cumplirse. En realidad, se puede tratar de varias funciones, por ejemplo, puede controlar la integridad referencial, otro que se encargue de la interfaz, tal como abrir y cerrar ciertos formularios o funcionalidades que tengan que ver con la seguridad, menús, etc., tiene los métodos que serán llamados desde las distintas partes de la interfaz o para acceder a la capa de datos, tal como se apreciará en el ejemplo. Esta capa interactúa con la capa de presentación para recibir solicitudes y presentar resultados, y con la capa de datos, para solicitar el manejador de base de datos que realice la operación de almacenamiento, edición, eliminación, consulta de datos u otra.

- Capa de datos

  Esta capa se encarga de acceder a los datos, se debe usar la capa de datos para almacenar y recuperar toda la información de sincronización del Sistema.

  Es aquí donde se implementa las conexiones al servidor y la base de datos propiamente dicha, se invoca a los procedimientos almacenados los cuales reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

  Todas estas capas pueden resistir en un único ordenador (no debería ser lo usual), pero es lo más frecuente. En sistemas complejos se llega a tener varios ordenadores sobre los cuales reside la capa de datos, y otra serie de ordenadores sobre los cuales reside la base de datos.

  Se recomienda que si el crecimiento de las necesidades o complejidad aumenta se debe separar en dos o más ordenadores, los cuales recibirán las peticiones del ordenador en que resida la capa de negocio, Esta recomendación es válida para la capa de negocio.

- Capas y niveles

  Las capas se ocupan de la división lógica de componentes y funcionalidad y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lugares. Por el contrario, los Niveles se ocupan de la distribución física de componentes y funcionalidad de servidores separados.

2.2.8 Sistema gestor de base de datos (SGBD)

Según [15] no debemos confundir una base de datos con un Sistema Gestor de Base de Datos. Una base de datos es la información almacenada, que cumple una serie características

y restricciones, pero para que la información pueda ser almacenada y el acceso a la misma satisfaga las características exigidas a una base datos, es necesario que exista una serie de procedimientos, un sistema software, que sea capaz de llevar a cabo tal labor. A este sistema software es lo que llamamos Sistema Gestor de Base de Datos (SGBD).

Un SGBD permite:

- Definir los datos a los distintos niveles de abstracción (físico, lógico y externo).
- Manipulación de los datos en la base de datos. Permitiendo insertar, modificar, borrar y consultar los datos.
- Mantenimiento de la integridad de base de datos.
- Control de la privacidad y seguridad de los datos en la base de datos.

Las funciones de un SGBD son:

- Función de verificación: Va a permitir al diseñador de la base de datos especificar los elementos que lo integran, su estructura y las relaciones que existen entre ellos, las reglas de integridad semántica, etc.,
- Función de manipulación: Los usuarios podrán recuperar la información o actualizarla. La consulta podrá ser de dos tipos: Consulta selectiva y consulta sobre la totalidad de los datos. La actualización podrá realizarse con tres operaciones diferentes: Borrado o eliminación de elementos, modificación de datos, inserción de nuevos elementos.
- Función de control: Permite funciones como, cambiar la capacidad de ficheros, obtener estadísticas de utilización y funciones de seguridad como: Copias de seguridad, re arranque del sistema, protección frente a accesos no autorizados.

2.2.9 SQL (Structured Query Language)

Según [16] ahora que tiene un conocimiento fundamental del modelo relacional, es el momento para introducirlo a SQL y sus características básicas. Como recordará de la sección “Entienda las bases de datos relacionales” vista anteriormente en este capítulo, SQL se basa en el modelo relacional, aunque no se trate de una aplicación exacta. Mientras el modelo relacional proporciona las bases teóricas de la base de datos relacional, es el lenguaje SQL el que apoya la aplicación física de esa base de datos. SQL, el lenguaje relacional casi universalmente aplicado, es diferente de otros lenguajes computacionales como C, COBOL y Java, los cuales son de procedimiento. Un lenguaje de procedimiento define *cómo* las operaciones de una aplicación deben realizarse y el orden en el cual se realizan. Un lenguaje de no procedimiento, por otro lado, se refiere a los resultados de una operación; el entorno fundamental del software determina cómo se procesan las operaciones. Esto no quiere decir que SQL respalda a la funcionalidad de no procedimiento. Por ejemplo, los procedimientos almacenados, agregados a varios productos RDBMS hace algunos años, son parte del estándar SQL:2006 y proporciona capacidades parecidas a procedimiento. Muchos de los proveedores de RDBMS añadieron extensiones a SQL para proporcionar esas capacidades de procedimiento, como Transact-SQL encontrado en Sybase y Microsoft SQL Server y PL/SQL encontrado en Oracle.

SQL aún carece de muchas de las capacidades básicas de programación de la mayoría de los lenguajes computacionales. Por esta razón, a menudo SQL se considera como un *sublenguaje* de datos porque se utiliza con frecuencia en asociación con la aplicación de lenguajes de programación como C y Java, lenguajes que no fueron diseñados para la manipulación de datos almacenados en una base de datos.

Como resultado, SQL se utiliza en conjunto con la aplicación del lenguaje para proporcionar un medio eficaz de acceder a los datos, razón por la cual se considera a SQL como un sublenguaje.

2.2.10 Pruebas funcionales

En [17] el objetivo de la prueba funcional es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones. La prueba funcional toma el punto de vista del usuario. En la prueba funcional las funciones son probadas ingresando las entradas y examinando las salidas.
La estructura interna del programa raramente es considerada.

2.2.11 Pruebas de usuario

En [17], es la prueba realizada por el tipo de persona que usará el producto. Puede ser realizado en cualquier momento durante el desarrollo, en el lugar del cliente o en el de desarrollo, en ejercicios completamente dirigidos o a la discreción del usuario.

2.3 Bases Conceptuales

2.3.1 Junta administradora de servicio de saneamiento

En [18], es una Asociación que se encarga de la prestación de los servicios de saneamiento en los centros poblados y comunidades rurales. Se llama servicios de saneamiento a los servicios de agua potable, disposición de excretas (letrinas) y eliminación de basura (pozo relleno). Ahora presentamos, gráficamente, las instancias que la ley reconoce cuando se organiza la JASS (Junta Administradora de servicio de saneamiento).

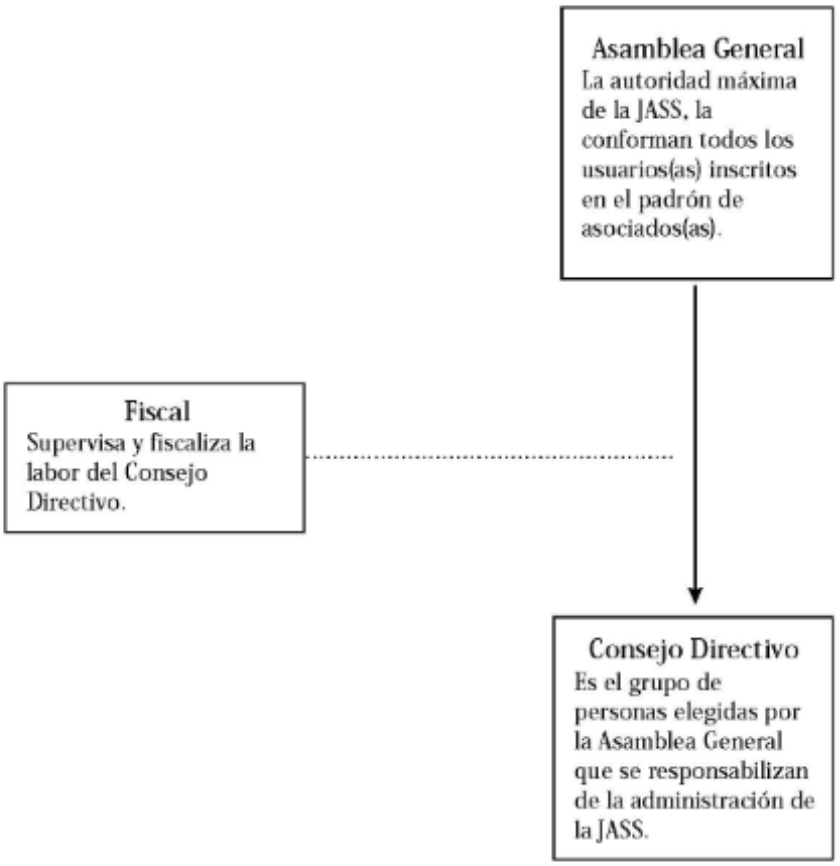


Fig. 2.9 Organización de la JASS

La Fig. 2.9 muestra como está organizada jerárquicamente la JASS (Junta Administradora de servicios).

2.3.2 Asociados

En [18], los asociados y las asociadas son los comuneros y comuneras que participaron en las faenas de construcción del sistema. También son aquellas personas que, sin haber participado en las faenas de construcción, han pagado su cuota de inscripción. Todo asociado(as) está inscrito(a) en el padrón de Asociados de la JASS. Cada vivienda tiene derecho a un punto de agua y solo puede inscribir a un usuario(a) en la JASS.

2.3.3 Consejo Directivo

En [18], es la instancia responsable de la administración de la JASS. Tiene la finalidad de asegurar la calidad del servicio y una buena gestión y administración. Sus miembros son elegidos por la Asamblea General por un período de 2 años y son responsables de manera conjunta de las decisiones que toman. ¿Quiénes lo integran? Los miembros del Consejo Directivo son cinco:

- El presidente(a)
- El secretario(a).
- El tesorero(a).
- Dos vocales.

2.3.4 Asamblea general

En [18], la Asamblea general está integrada por todos los asociados(as) inscritos(as) en el padrón de asociados(as) de la JASS.

Funciones de la asamblea general.

- Aprobar el estatuto, reglamento interno y sus modificaciones.
- Aprobar el plan de trabajo, el presupuesto anual y la cuota familiar.
- Supervisar y evaluar las actividades realizadas por el Consejo Directivo.
- Designar al comité electoral.
- Resolver y sancionar casos de denuncias a miembros del Consejo Directivo y/o asociados(as).
- Confirmar o revocar las sanciones impuestas por el Consejo Directivo.
- Elegir a los miembros del Consejo Directivo.
- Otras funciones que por su naturaleza le corresponden como máxima autoridad de la JASS.

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1 Tipo de investigación

El tipo de investigación determina la manera cómo el investigador aborda el evento de estudio, de acuerdo a las técnicas, métodos, instrumentos y procedimientos propios de cada uno. Para el presente trabajo de investigación se determinó que el tipo de investigación es aplicada, debido a que tiende a la resolución de problemas, dirigidas a conseguir innovaciones, mejoras de procesos o productos, en este caso mediante el uso de la tecnología.

3.2 Nivel de la investigación

El nivel de investigación está relacionado con el grado de profundidad y alcance que se pretende con la misma. De acuerdo a los propósitos del presente trabajo de investigación se determinó que el nivel de la investigación es correlacional debido a que se pretende probar las hipótesis formuladas y medir así, la relación de las variables dependiente e independiente.

3.3 Diseño de la investigación

El tipo de diseño para el trabajo de investigación es experimental, en este tipo de diseño, el investigador espera comprobar los efectos de una intervención específica, para esto tiene un papel activo para llevar a cabo la intervención. En el presente proyecto, se emplea este diseño de investigación, ya que se tuvo un papel activo realizando una intervención con el desarrollo del sistema de control de asistencia para comprobar el efecto que causaba en la Junta Administradora de Servicios de Saneamiento Quilcas.

3.4 Hipótesis

3.4.1 Hipótesis general

La implementación de un sistema de código de barras sí optimiza el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas.

3.4.2 Hipótesis específicas

a) Mediante el diseño de una interfaz, sí mejorará la administración del padrón general de asociados.
b) El desarrollo de un procedimiento usando el lector de códigos de barras, sí agilizará el registro de asistencia.
c) Al generar la lista de inasistencia de forma automática, sí garantizará que el cobro de las multas sea confiable.

3.5 Variables

3.5.1 Variable independiente

El desarrollo e implementación de un sistema de código de barras.

3.5.2 Variable dependiente

Control de asistencia en la Junta Administradora de Servicios de Saneamiento Quilcas.

3.5.3 Definición Conceptual

3.5.3.1 Desarrollo e implementación de un sistema

Aplicación de principios, técnicas, herramientas y métodos para la construcción, implementación, instalación, mantenimiento y gestión de sistemas de información.

3.5.3.2 Control de asistencia

Es un procedimiento administrativo, que consiste en la puesta en práctica de una serie de instrumentos, con la finalidad de registrar y controlar al personal de una determinada empresa o institución.

3.5.4 Definición operacional

3.5.4.1 Integridad de la información

De la variable dependiente. Corresponde al grado de no corrupción de la información. El sistema distribuye el manejo de información de los asociados a través de perfiles de manera óptima.

3.5.4.2 Velocidad

De la variable independiente. Corresponde al entorno del sistema que simplificará el tiempo de registro de asistencia de los asociados, esto impactará en el tiempo de entrega de la información final de los mismos.

3.5.4.3 Confiabilidad

De la variable dependiente. Corresponde al grado de aceptación de los indicadores finales. El registro de asistencia a través del sistema aumenta la confianza al tratarse de un sistema automatizado y el manejo de una base de datos.

3.6 Población y muestra

3.6.1 Población

La población son los 1162 asociados inscritos en el padrón general conformados por 7 sectores del distrito de Quilcas, los cuales son: Pampa, Santa Cruz, 27 de mayo, Patac, Mancoculi, Unuimarca y Centenario.

3.6.2 Muestra

La muestra es no aleatoria o dirigida, en este caso corresponde a los 362 asociados del sector Pampa inscritos en el padrón general de la entidad.

3.7 Matriz de consistencia

Es un instrumento fundamental de un trabajo de investigación, consta de varios cuadros formados por filas y columnas, permite al investigador evaluar el grado de conexión lógica y coherencia entre el título, el problema, los objetivos, las hipótesis, las variables, el tipo, método, diseño e instrumentos de investigación; de mismo modo la población y la muestra correspondiente de estudio. En la tabla 6 se muestra la matriz de consistencia

Tabla 6

Matriz de consistencia

PROBLEMA	OBJETIVOS	HIPÓTESIS	VARIABLES	METODOLOGÍA
Problema general: ¿Cómo optimizar el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas? **Problemas específicos:** **a)** ¿Cómo mejorar la administración del padrón general de asociados? **b)** ¿De qué manera se puede agilizar el registro de asistencia? **c)** ¿Cómo se garantiza que la lista de inasistencia para el cobro de las multas sea confiable?	**Objetivo general:** Implementar un sistema de código de barras para optimizar el control de asistencia de los asociados en la Junta Administradora Quilcas **Objetivos específicos:** **a)** Diseñar una interfaz para administrar el padrón general de asociados. **b)** Desarrollar un procedimiento para el registro de asistencias mediante el uso del lector de código de barras. **c)** Generar la lista de inasistencia de manera automática para el cobro de las multas.	**Hipótesis general:** La implementación de un sistema de código de barras sí optimiza el control de asistencia de los asociados en la Junta Administradora de Servicios de Saneamiento Quilcas. **Hipótesis específicas:** **a)** Mediante el diseño de una interfaz si mejorará la administración del padrón general de asociados. **b)** El desarrollo de un procedimiento usando el lector de código de barras, si agilizará el registro de asistencia. **c)** Al generar la lista de inasistencia de forma automática, sí garantizará que el cobro de las multas sea confiable.	**Variable Independiente:** El desarrollo e implementación de un sistema de código de barras. **Dimensión:** - Velocidad **Variable Dependiente:** Control de asistencia en la Junta Administradora de Servicios de Saneamiento Quilcas. **Dimensiones:** **-** Integridad de la información. - Confiabilidad.	**Tipo de investigación:** Aplicada. **Nivel de investigación:** Correlacional. **Diseño de investigación:** Experimental. **Población:** 1162 asociados divididos en 7 sectores, inscritos en el padrón general de la JASSQ. **Muestra:** Es no aleatoria o dirigida y corresponde a los 362 asociados del sector Pampa. **Técnica:** - Entrevistas.

3.8 Descripción de la metodología seleccionada

La investigación aplicada busca la generación de conocimiento con aplicación directa a los problemas de la sociedad o el sector productivo. Esta se basa fundamentalmente en los hallazgos tecnológicos de la investigación básica, ocupándose del proceso de enlace entre la teoría y el producto

La investigación aplicada tiene por objetivo la generación de conocimiento con aplicación directa y a mediano plazo en la sociedad o en el sector productivo. Este tipo de estudios presenta un gran valor agregado por la utilización del conocimiento que proviene de la investigación básica. De esta manera, se genera riqueza por la diversificación y progreso del sector productivo. Así, la investigación aplicada impacta indirectamente en el aumento del nivel de vida de la población y en la creación de plazas de trabajo. La Figura 1 presenta el desarrollo del proceso investigativo desde la concepción de la idea hasta la elaboración del producto.

Propiedad intelectual: fundamentos y utilidad Para que los resultados de la investigación rindan beneficios económicos por el mayor tiempo posible, se debe proteger adecuadamente el conocimiento. Los resultados de la investigación aplicada se dividen en dos grandes categorías

- Núcleo tecnológico: corresponde a la base de conocimiento genérico necesario para el desarrollo de prototipos, que no depende del sector productivo de la industria con la que la universidad colabora. Este conocimiento puede ser aplicado a varios campos productivos. La universidad debe mantener la propiedad de este conocimiento para salvaguardar la libertad de explotación industrial.

- Tecnologías específicas: dependen principalmente del sector productivo y de las necesidades de la industria con la que se colabora. Los resultados del núcleo tecnológico no dependen del sector de actividad de la industria al contrario de las tecnologías específicas. conocimiento para la universidad. Puesto que la colaboración genera beneficios económicos para la industria, esta debe percibir el acuerdo como una inversión necesaria. S e requiere un apoyo financiero para la etapa de maduración y transferencia tecnológica. E n contraparte, la universidad debe invertir en la creación y el mantenimiento de grupos de investigación altamente calificados, en la creación de laboratorios especializados de alto nivel y en la etapa de investigación inicial para permitir el nacimiento de nuevos conceptos.

Estructura de una investigación aplicada:

- Para el problema general específico incluye preguntas clave de la investigación. Deben abordarse sin ambigüedad y explicitarse de manera clara, concisa y breve.

- Deben establecer en número limitado y ordenados, al final del trabajo ha de proporcionarse alguna respuesta a las preguntas formuladas.

- Los procedimientos de análisis deben hacerse explícitos, justificando la razón de su uso (sobre todo ante distintas alternativas), el alcance y las limitaciones de la técnica empleada en cada caso.

- Si se utilizan procedimientos informáticos, se recomienda indicar el programa utilizado.

Por otro lado, la Investigación Científica Aplicada puede también orientarse a la producción de conocimientos y métodos que mejorar o hacer mucho más eficiente el sector vengan a productivo de bienes o servicios, buscando imprimir en la vida del humano promedio bienestar, a través de un medio productivo mucho más eficiente. Igualmente, puede tener una fase teórica e investigativa, así como otra experimental, que lleve incluso a la elaboración de prototipos.

CAPÍTULO IV

DESARROLLO DE LA INVESTIGACIÓN

4.1 Requerimientos del sistema

Se establece una comunicación continua con el cliente, para obtener los requerimientos principales del sistema, tomando en cuenta la prioridad y tiempo estimado para el desarrollo.

4.1.1 Identificación de requerimientos

Para identificar los requerimientos de la entidad, se aplicó la técnica de entrevista, coordinando hora y fechas con: el presidente, secretario, tesorera y vocales de la Junta Administradora de Servicios de Saneamiento Quilcas, donde se detallan la realidad y las necesidades principales. En las tablas 7 – 12 se muestran el formato de las entrevistas.

Tabla 7

Entrevista 01

Entrevista 01 – E01			
a) Identificación de reunión			
Tema	Situación actual.	**Cliente**	JASSQ
Alcance	Relevamiento de información.		
Fecha	10/05/217	**Horario**	13:30 – 17:30
Lugar	Jr. Ramón Castilla - JASSQ		

b) Participantes			
#	**Nombre**	**ID**	**Cargo**
1	Félix Cárdenas Apolinario	FC	Presidente
2	Iván Diego Rivera Meza	IR	Tesista

c) Temas tratados			
Tema	**#**	**Plant. Por**	**Detalles**
Explicación de la situación actual del control de asistencia a las asambleas	1	FC/IR	Se explicó cuáles son las actividades para el control de asistencia y como es el manejo del padrón general de asociados.

Tabla 8

Entrevista 02

Entrevista 02 – E02			
a) Identificación de reunión			
Tema	Asociados.	**Cliente**	JASSQ
Alcance	Relevamiento de información.		
Fecha	17/05/2107	**Horario**	13:30 – 17:30
Lugar	Jr. Ramón Castilla - JASSQ		

b) Participantes			
#	**Nombre**	**ID**	**Cargo**
1	Santos Apolinario Núñez	SA	Secretario
2	Iván Diego Rivera Meza	IR	Tesista

c) Temas tratados			
Tema	**#**	**Plant. Por**	**Detalles**
Explicación cómo se gestiona a los asociados.	1	SA	Se explicó como gestiona a los asociados que necesitan cambios, a los nuevos y los que dan de baja su servicio.
Registro búsqueda actualización o eliminación del asociado.	2	SA/IR	Se explicó que dichos actividades se lleva a cabo manualmente.

Tabla 9

Entrevista 03

Entrevista 03 – E03					
a) Identificación de reunión					
Tema	Asambleas.		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	24/07/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Santos Apolinario Núñez	SA	Secretario		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
Explicación como se crea las asambleas.	1	SA	Se explicó como es la creación de las agendas y la fecha para las asambleas que el presidente ordena.		
Cambio de fecha o de agenda de las asambleas.	2	SA/IR	Se realiza manualmente y en ocasiones se extravía el papel donde se apuntó.		

Tabla 10

Entrevista 04

Entrevista 04 – E04					
a) Identificación de reunión					
Tema	Asistencia.		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	31/05/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Santos Apolinario Núñez	SA	Secretario		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
Explicación de cómo contabiliza las asistencias.	1	SA	Se explicó que el conteo de las asistencias se hace de manera manual.		
Cuanto demora en contabilizar las asistencias	2	SA/IR	Se detalló que si lo hace ese mismo día demora de una a dos horas, por lo general lo deja para la semana siguiente.		

Tabla 11

Entrevista 05

Entrevista 05 – E05					
a) Identificación de reunión					
Tema	Control de asistencia.		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	07/06/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Elvira Córdova Quispe	EC	Vocal		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
Explicación como es el control de asistencia.	1	EC	Se explicó las actividades que realiza para el control de asistencia durante el ingreso, dentro y a la salida de la asamblea.		
De qué manera registra la asistencia.	2	EC/IR	Se detalló que cuentan con un padrón por cada sector, en el cual los asociados firman su entrada y salida, lo cual genera desorden.		

Tabla 12

Entrevista 06

Entrevista 06 – E06					
a) Identificación de reunión					
Tema	Inasistencias.		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	14/07/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Paola Roca Inga	PR	Tesorera		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
Explicación como realiza las cobranzas de las multas por las inasistencias.	1	PR	Se explicó que espera al secretario a que termine de contabilizar las asistencias.		
De qué manera obtiene la listas de inasistencias.	2	PR/IR	Se detalló que se realiza manualmente comparando la lista de asistencia con el padrón general y así obtiene la lista de inasistencias.		

4.1.2 Especificación de requerimientos

Las entrevistas dan como resultado las historias de usuario, las cuales deben ser descritas en un lenguaje común, para que puedan ser entendidas por todos (Cliente, Desarrollador y Usuarios), representando los requerimientos que debe cumplir el sistema.

El nombre de las historias de usuario que se obtuvieron son:

- Acceso al sistema
- Gestión de usuarios
- Registro de asociados
- Gestión de asociados
- Gestión de asambleas

- Control de asistencia
- Consulta de asistencia
- Consulta de inasistencia
- Exportar de datos a Excel
- Generación de reporte

A continuación, en las tablas 13 – 22 se detallan las historias de usuario, las cuales se usarán para llevar a cabo el desarrollo del sistema.

Tabla 13

Historia de usuario – Acceso al sistema

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Administrador, usuarios del sistema.
Nombre Historia: Acceso al sistema	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Media **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 1**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: Los usuarios del sistema tendrán un nombre de usuario y clave única con la que podrán ingresar. De acuerdo a su perfil.	
Observaciones: El presidente tendrá el perfil de administrador, y los usuarios del sistema tendrán perfil de secretario, tesorera y vocal.	

Tabla 14

Historia de usuario – Gestión de usuario

HISTORIA DE USUARIO	
Número: 2	**Usuario:** Presidente.
Nombre Historia: Gestión de usuario	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 1**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El sistema permitirá al presidente el acceso a todas las funcionalidades del sistema. Así mismo podrá realizar las operaciones de crear, cambiar el perfil, edición y eliminación de los usuarios del sistema.	
Observaciones: El secretario tendrá acceso al registro y gestión de asociados así mismo gestión de asambleas y consulta de asistencias, la tesorera a consulta de inasistencia y el vocal al control de asistencia.	

Tabla 15

Historia de usuario – Registro de asociados

HISTORIA DE USUARIO	
Número: 3	**Usuario:** Secretario
Nombre Historia: Registro de asociados	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 3	**Iteración Asignada: 1**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El sistema debe permitir, guardar y mostrar la siguiente información: código, apellidos, nombre, DNI, dirección, sector, estado de cada asociado así mismo el sistema debe permitir generar, mostrar y guardar el código de barras de cada asociado, para que luego sea impreso en una ficha respectiva.	
Observaciones: El sector es la zona donde viven los asociados y el estado se divide en 2 que son los siguientes: activo o exonerado, el primero paga las multas y el segundo no.	

Tabla 16

Historia de usuario – Gestión de asociados

HISTORIA DE USUARIO	
Número: 4	**Usuario:** Secretario
Nombre Historia: Gestión de asociados	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 1**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El secretario podrá editar, eliminar y realizar la búsqueda de los asociados por apellido, código, sector o estado también podrá visualizar el total de asociados.	
Observaciones: El secretario podrá realizar dichos procesos previa documentación firmada por el presidente. El código es único, por lo tanto no se podrá modificar, si se elimina el código pasará a otro asociado activo.	

Tabla 17

Historia de usuario – Gestión de asambleas

HISTORIA DE USUARIO	
Número: 5	**Usuario:** Secretario
Nombre Historia: Gestión de asambleas	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Media **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 1**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: Se tendrá un registro de asambleas por fecha con su respectiva agenda debido a que el presidente las toma como referencia, además el secretario podrá realizar la búsqueda, creación y edición de dichas asambleas y podrá visualizar el total de las mismas ya que en ocasiones por motivos especiales se postergan a otra fecha.	
Observaciones: La búsqueda de las asambleas debe realizarse entre rango de fechas para que la búsqueda sea más sencilla.	

Tabla 18

Historia de usuario – Control de asistencia

HISTORIA DE USUARIO	
Número: 6	**Usuario:** Vocal
Nombre Historia: Control de asistencia	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 2**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El vocal asignado tendrá una opción para iniciar el registro de asistencia, así el lector de código de barras podrá leer los código de barra de la ficha de cada asociado y el sistema debe mostrar en pantalla el código, apellidos, nombre, sector, hora de entrada y fecha por cada asociado, una vez finalice, el vocal tendrá la opción de cerrar el registro de asistencia.	
Observaciones: La asistencia de los asociados exonerados no es obligatoria, pero el sistema los registrará normalmente él además debe permitir ingresar manualmente el código si el asociado no cuenta con su ficha al momento del registro.	

Tabla 19

Historia de usuario – Consulta de asistencia

HISTORIA DE USUARIO	
Número: 7	**Usuario:** Secretario
Nombre Historia: Consulta de asistencia	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 2**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El sistema le permitirá al secretario poder visualizar el total de y la lista de asistencia el cual tendrá lo siguiente, código, apellidos, nombre, sector, hora de entrada y fecha de cada asociado que asistió a la asamblea mensual, así mismo podrán hacer una búsqueda por código y apellido, posteriormente podrán eliminar la lista de asistencia para poder liberar espacio en la base de datos.	
Observaciones: Para eliminar la lista de asistencia, el presidente y el secretario tendrán la opción de seleccionar toda lista caso contrario otra opción para deseleccionar.	

Tabla 20

Historia de usuario – Consulta de inasistencia

HISTORIA DE USUARIO	
Número: 8	**Usuario:** Tesorera
Nombre Historia: Consulta de inasistencia	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Alta **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 2**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: La tesorera podrá visualizar la lista de inasistencia.	
Observaciones: La tesorera solo tendrá acceso a dicho formulario de acuerdo a su perfil.	

Tabla 21

Historia de usuario – Exportar datos Excel

HISTORIA DE USUARIO	
Número: 9	**Usuario:** Tesorera
Nombre Historia: Exportar datos a Excel.	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Media **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 3**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: La tesorera podrá exportar a Excel la lista de inasistencias, para que realice el cobro respectivo de las multas a todos los asociados activos, excepto a los exonerados.	
Observaciones: El archivo generado en Excel la tesorera podrá llevarse en forma digital o impreso, ya que todos los cobros de la Junta Administradora de Servicios de Saneamiento Quilcas lo realiza en su oficina.	

Tabla 22

Historia de usuario – Generar reporte

HISTORIA DE USUARIO	
Número: 10	**Usuario:** Presidente
Nombre Historia: Generar reporte	
Prioridad en Negocio: Alta **(Alta,Media,Baja)**	**Riesgo en Desarrollo:** Media **(Alta,Media,Baja)**
Puntos Estimados: 2	**Iteración Asignada: 3**
Programador Responsable: Iván Diego Rivera Meza	
Descripción: El sistema permitirá al presidente poder generar un reporte del padrón general de asociados de la Junta Administradora de Servicios de Saneamiento Quilcas, el cual puede ser impreso de manera directa o exportado a Excel y/o a PDF.	
Observaciones: El reporte generado por el sistema no contendrá el código de barras.	

4.1.3 Validación de requerimientos

En la tabla 23 se muestra la validación de los requerimientos.

Tabla 23

Matriz – Validación de requerimientos

Nº de Iteración	Nº de Historia de Usuario	Nombre Historia de Usuario	Resumen	Validación
1	1	Acceso al sistema	Los usuarios del sistema tendrán una clave única de acceso.	CONFORME
	2	Gestión de usuario	Crear, editar, eliminar y asignar perfil.	CONFORME
	3	Registro de asociados	Registrar datos y generar código de barras.	CONFORME
	4	Gestión de asociados	Crear, editar, buscar y eliminar.	CONFORME
	5	Gestión de asambleas	Crear, editar, buscar y eliminar.	CONFORME
2	6	Control de asistencia	Marcar asistencia por medio del lector de código de barras.	CONFORME
	7	Consulta de asistencia	Visualizar, consultar y eliminar la lista de asistencia.	CONFORME
	8	Consulta de inasistencia	Visualizar lista de inasistencia.	CONFORME
3	9	Exportar datos a Excel.	Exportar la lista de inasistentes.	CONFORME
	10	Generar reporte	Generar reporte de los asociados.	CONFORME

4.2 Análisis y diseño del sistema

Iniciamos con las fases de la metodología XP, las cuales son: Planificación, diseño, codificación y pruebas.

4.2.1 Planificación

Se quiere que el SISCA (Sistema Control de Asistencia) mediante el lector de código de barras permita registrar las asistencias de los asociados a las asambleas. Esto facilitará tanto el control como la obtención de la lista de inasistencia. Para le entrega de este proyecto, el sistema contará con los siguientes módulos:

- Usuario: Los usuarios tendrán un perfil definido en el sistema para poder acceder a las funcionalidades del mismo.
- Administración: Permitirá gestionar y registrar a los asociados y las asambleas.
- Control: Registrará todas las asistencias de los asociados.
- Consulta: Estarán los reportes tanto de las asistencias como de las inasistencias.
- Información: Muestra los datos del sistema como versión y nombre.

Los módulos mencionados anteriormente se han recopilado en reuniones con el presidente de la JASSQ, Félix Cárdenas Apolinario.

En la tabla 24 se muestra la asignación de los roles para el presente proyecto.

Tabla 24

Asignación de roles del proyecto

Roles	Asignado a:
Programador	Iván Diego Rivera Meza
Cliente	Félix Cárdenas Apolinario
Encargado de pruebas (Tester)	Iván Diego Rivera Meza
Encargado de seguimiento (Tracker)	Iván Diego Rivera Meza
Entrenador (Coach)	Iván Diego Rivera Meza
Consultor	Magno Baldeón Tovar, Jessica Vilchez Gutarra.
Gestor (Big Boss)	Iván Diego Rivera Meza

Basándonos en las historias de usuario definidas para el desarrollo del sistema de escritorio, se ha elaborado el siguiente plan de entrega, el cual muestra las historias de usuario que se llevarán a cabo en cada iteración. Para este plan de entrega se ha tomado en cuenta la prioridad y el esfuerzo de cada historia de usuario. En la tabla 25 se muestra el plan de entrega del proyecto.

Tabla 25

Plan de entrega del proyecto

Historias	Iteración	Prioridad	Esfuerzo	Fecha Inicio	Fecha Final
Historia 1	1	Alta	1	14/06/17	21/06/17
Historia 2	1	Alta	2	21/06/17	28/06/17
Historia 3	1	Alta	3	28/06/17	12/07/17
Historia 4	1	Alta	2	12/07/17	19/07/17
Historia 5	1	Alta	2	19/07/17	26/07/17
Historia 6	2	Alta	2	26/07/17	09/08/17
Historia 7	2	Alta	2	09/08/17	16/08/17
Historia 8	2	Alta	2	16/08/17	23/08/17
Historia 9	3	Alta	2	23/08/17	30/08/17
Historia 10	3	Alta	2	30/08/17	06/09/17

4.2.2 Primera iteración

En la primera iteración se han desarrollado los módulos Usuario y Administración, los cuales para lograr su progreso se ha utilizado y aplicado cada una de las herramientas que se destacan en la metodología XP.

En la tabla 26 se muestra de forma general las historias de usuario y sus módulos respectivos para la primera iteración.

Tabla 26

Historias de usuario

Número	Nombre	Módulo
1	Acceso al sistema	
2	Gestión de usuario	Usuario
3	Registro de asociados	Usuario
4	Gestión de asociados	Administración
5	Gestión de asambleas	Administración

En la tabla 27 se muestra de manera general las tareas de ingeniería de la primera iteración.

Tabla 27

Tareas de ingeniería

Número de Tarea	Historia	Nombre de la Tarea
1	1	Diseño de interfaz acceso al sistema.
2	2	Diseño de interfaz de usuario para la gestión de usuario.
3	2	Creación de la base de datos para la gestión de usuario.
4	2	Guardar datos de los usuarios del sistema en la base de datos.
5	3	Diseño de interfaz para el registro de asociados.

6	3	Creación de la base de datos para la el registro de asociados.
7	3	Guardar datos de los asociados en la base de datos.
8	4	Creación de las opciones para la gestión de asociados.
9	5	Diseño de interfaz de gestión de asambleas.
10	5	Creación de la base de datos para la gestión de asambleas.
11	5	Guardar datos de las asambleas en la base de datos.

En las tablas 28- 38 se detallan cada una de las tareas

Tabla 28

Tarea de ingeniería 1 para historia de usuario 1

Tarea de Ingeniería	
Número de tarea:**1**	Número de historia:**1**
Nombre de tarea: Diseño de interfaz accesos al sistema	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 14/06/17	Fecha fin: 21/06/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se realizará el diseño de la interfaz, en el cual los usuarios del sistema pondrán su usuario y contraseña.	

Tabla 29

Tarea de ingeniería 2 para historia de usuario 2

Tarea de Ingeniería	
Número de tarea:**2**	Número de historia:**2**
Nombre de tarea: Diseño de interfaz de usuario para la gestión de usuario.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 21/06/17	Fecha fin: 23/06/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se realizará el diseño de la interfaz de usuario para registrar a los usuarios del sistema.	

Tabla 30

Tarea de ingeniería 3 para historia de usuario 2

Tarea de Ingeniería	
Número de tarea:**3**	Número de historia:**2**
Nombre de tarea: Creación de la base de datos para la gestión de usuario.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 23/06/17	Fecha fin: 27/06/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se creará la base de datos para guardar la información de los usuarios que tendrán un perfil de sistema.	

Tabla 31

Tarea de ingeniería 4 para historia de usuario 2

Tarea de Ingeniería	
Número de tarea:**4**	Número de historia:**2**
Nombre de tarea: Guardar datos de los usuarios del sistema en la base de datos.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 27/06/17	Fecha fin: 28/06/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se necesita guardar la información en la base de datos para mantener registro de cada usuario que tendrá acceso al sistema.	

Tabla 32

Tarea de ingeniería 5 para historia de usuario 3

Tarea de Ingeniería	
Número de tarea:**5**	Número de historia:**3**
Nombre de tarea: Diseño de interfaz para el registro de asociados.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 28/06/17	Fecha fin: 04/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se diseñará el diseño de la interfaz para el registro de asociados, en donde se generará el código de barras de cada uno.	

Tabla 33

Tareas de ingeniería 6 para historia de usuario 3

Tarea de Ingeniería	
Número de tarea:**6**	Número de historia:**3**
Nombre de tarea: Creación de la base de datos para la el registro de asociados.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 04/07/17	Fecha fin: 11/07/017
Programador responsable: Iván Diego Rivera Meza	
Descripción: Crear la base de datos de asociados para almacenar la información requerida de cada asociado.	

Tabla 34

Tareas de ingeniería 7 para historia de usuario 3

Tarea de Ingeniería	
Número de tarea:**7**	Número de historia:**3**
Nombre de tarea: Guardar datos de los asociados en la base de datos.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 11/07/17	Fecha fin: 12/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se guardar en la base de datos la información requerida de cada asociado.	

Tabla 35

Tareas de ingeniería 8 para historia de usuario 4

Tarea de Ingeniería	
Número de tarea:**8**	Número de historia:**4**
Nombre de tarea: Creación de las opciones para la gestión de asociados.	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha inicio: 12/07/17	Fecha fin: 19/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se creará las opciones dentro de la interfaz de registro de asociado para editar, eliminar y buscar a los asociados.	

Tabla 36

Tareas de ingeniería 9 para historia de usuario 5

Tarea de Ingeniería	
Número de tarea:**9**	Número de historia:**5**
Nombre de tarea: Diseño de interfaz de gestión de asambleas.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 19/07/17	Fecha fin: 21/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se diseñará la interfaz de gestión de asambleas para registrar, eliminar, editar y buscar.	

Tabla 37

Tareas de ingeniería 10 para historia de usuario 5

Tarea de Ingeniería	
Número de tarea:**10**	Número de historia:**5**
Nombre de tarea: Creación de la base de datos para la gestión de asambleas..	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 21/07/17	Fecha fin: 25/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se creará la base de datos para almacenar información requerida de cada asamblea.	

Tabla 38

Tareas de ingeniería 11 para historia de usuario 5

Tarea de Ingeniería	
Número de tarea:**11**	Número de historia:**5**
Nombre de tarea: Guardar datos de las asambleas en la base de datos.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio:25/07/17	Fecha fin: 26/07/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se guarda en la base de datos la información requerida por cada asamblea.	

4.2.2.1 Diseño

Se diseñaron los prototipos usando el software Balsamiq, como se muestran en las figuras 4.1 - 4.4.

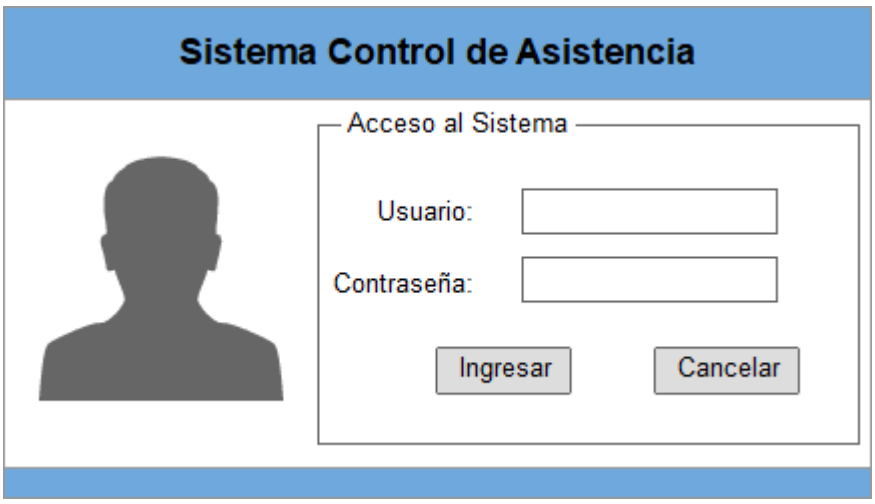


Fig. 4.1 Diseño - Acceso al Sistema

La Fig. 4.1 muestra el diseño de la interfaz para el acceso al sistema.

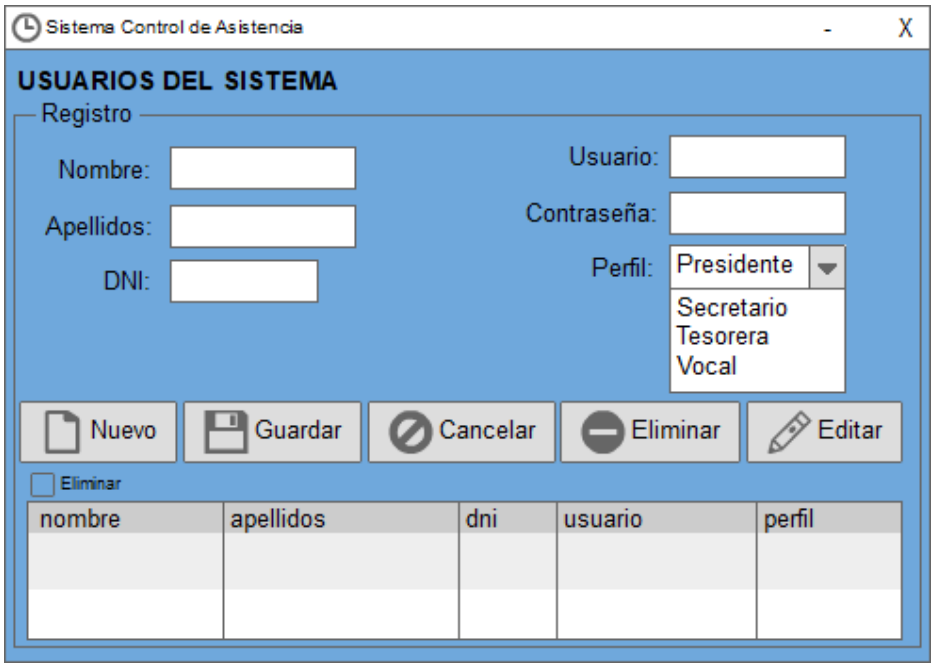


Fig. 4.2 Diseño – Gestión de usuarios.

La Fig. 4.2 muestra el diseño de la interfaz para la gestión de usuarios del sistema.

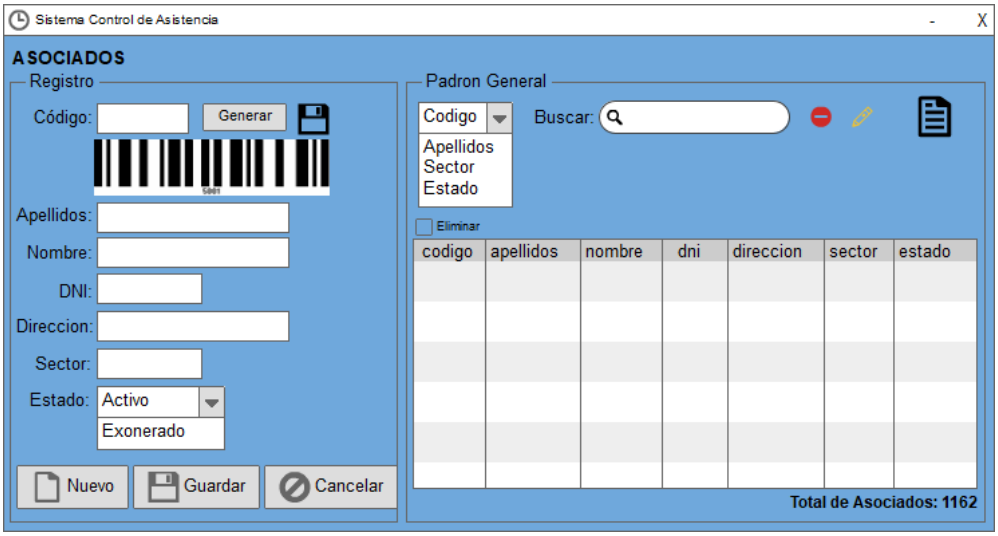


Fig. 4.3 Diseño – Registro de asociados

La Fig. 4.3 muestra el diseño de la interfaz para el registro y gestión de los asociados.

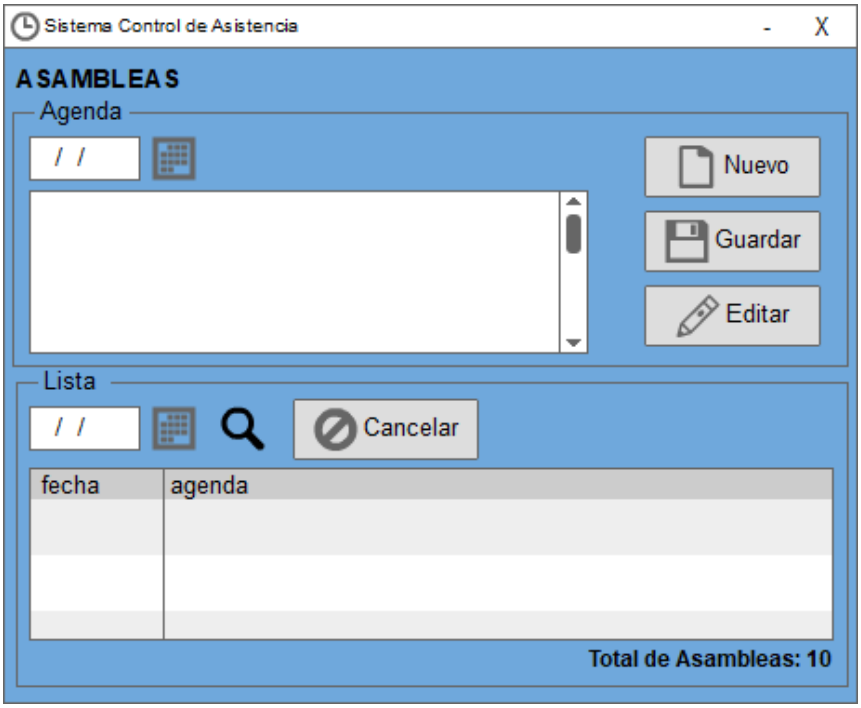


Fig. 4.4 Diseño – Gestión de asambleas

La Fig. 4.4 muestra el diseño de la interfaz para la gestión de asambleas.

4.2.2.2 Codificación

Una vez aceptados los prototipos por el cliente, se procede a la codificación (ver anexos 11-26), a continuación, se muestra los resultados en los siguientes pantallazos (Fig. 4.5 – 4.8).

Fig. 4.5 Interfaz para el acceso al sistema

La Fig. 4.5 muestra la interfaz de acceso al sistema, permite identificar al usuario mediante el nombre de usuario y su contraseña para hacer uso del sistema.

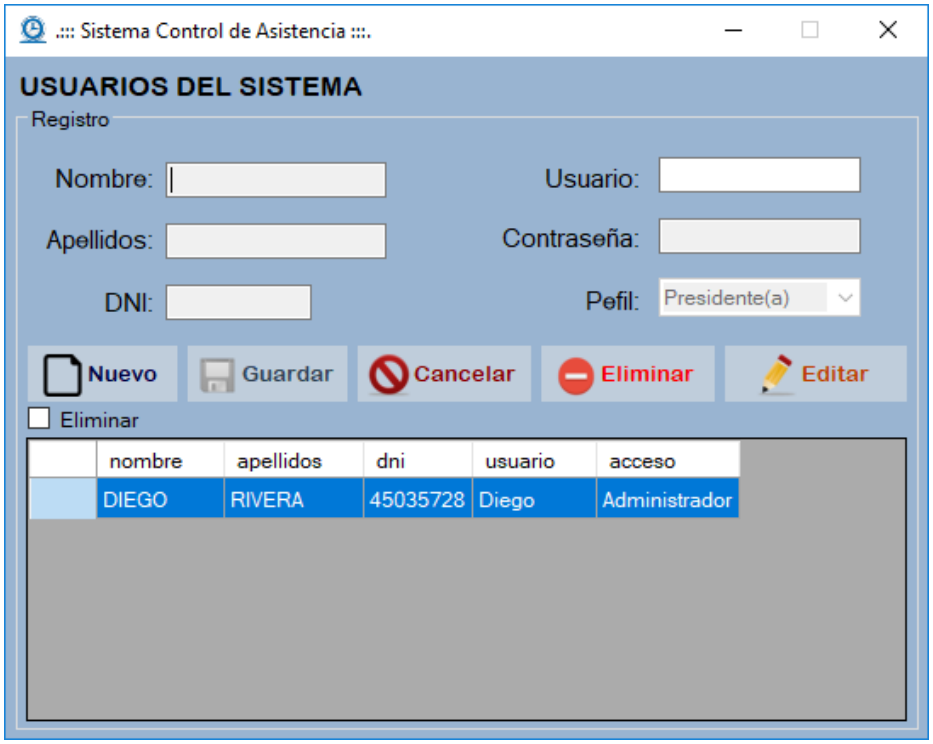


Fig. 4.6 Interfaz para usuarios del sistema

La Fig. 4.6 muestra la interfaz del formulario de usuarios del sistema, donde se crea, edita, da perfil y elimina a dichos usuarios.

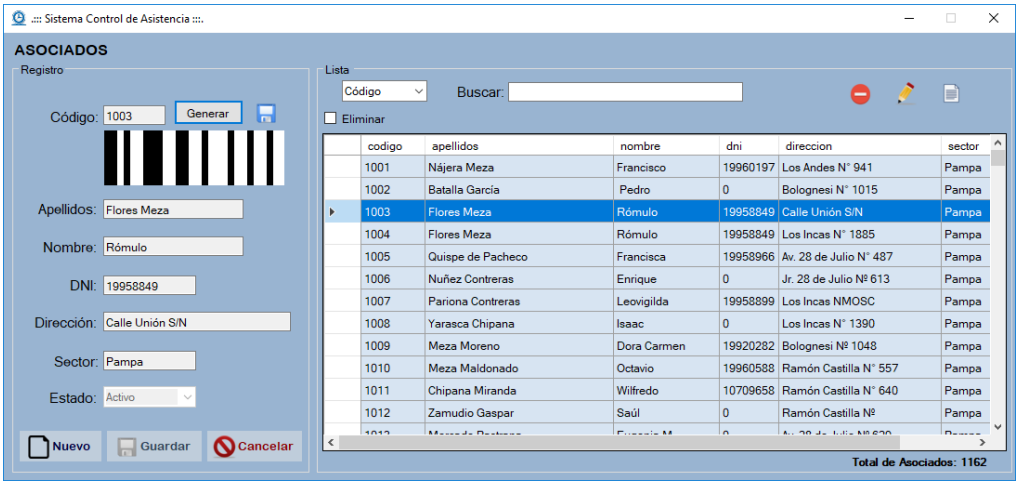


Fig. 4.7 Interfaz para los asociados

La Fig. 4.7 muestra la interfaz del formulario de los asociados, donde se registra, genera código de barras, edita, se busca y elimina a los asociados.

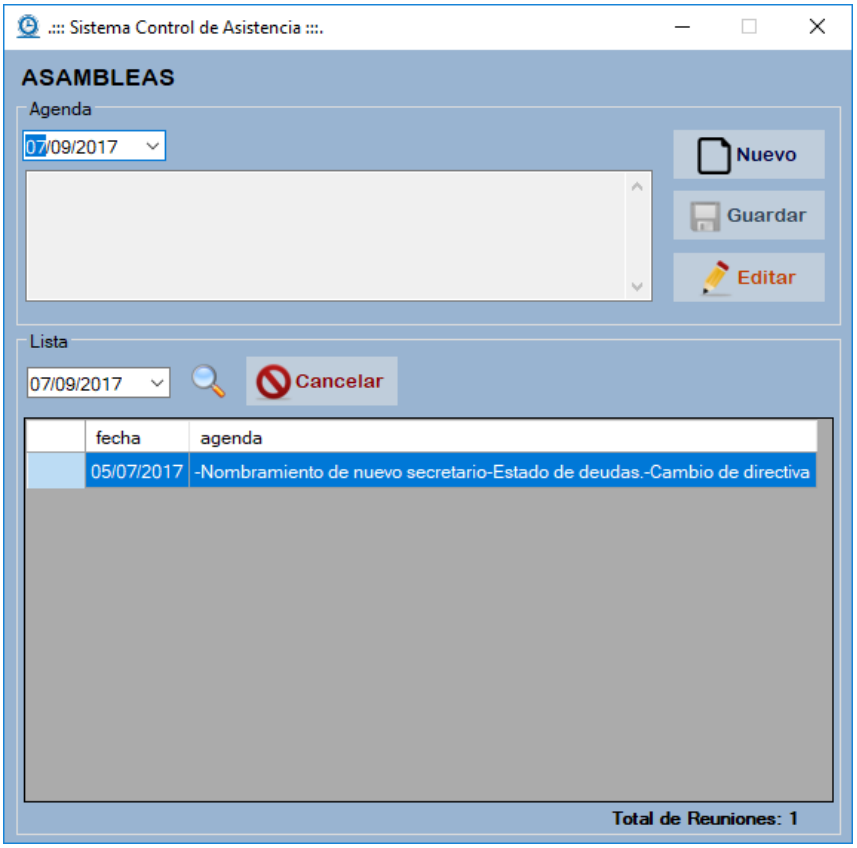


Fig. 4.8 Interfaz para asambleas

La Fig. 4.8 muestra la interfaz del formulario de usuarios del sistema, donde se crea y edita tanto la agenda como la fecha de las asambleas.

4.2.2.3 Pruebas

Para validar la primera iteración, se aplicará las pruebas de aceptación, las cuales verifican que el sistema funcione de acuerdo a las historias de usuario. En las tablas 39 – 43 se muestran las siguientes pruebas realizadas.

Tabla 39

Caso de prueba – Acceso al sistema

Caso de prueba	
Código: 1	**Nº Historia de usuario: 1**
Historia de usuario: Acceso al sistema.	
Condiciones de ejecución: Cada usuario del sistema debe contar con un perfil de usuario, nombre de usuario y contraseña para poder ingresar al sistema.	
Entradas/Pasos de ejecución: 1. Dar clic en el ícono del sistema 2. Ingresar nombre de usuario y contraseña. 3. Clic en el botón ACEPTAR. 4. Visualizar el panel principal del sistema.	
Resultado esperado: Acceso a las funcionalidades del sistema dependiendo del perfil de usuario.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 40

Caso de prueba – Gestión de usuario

Caso de prueba	
Código: 2	**Nº Historia de usuario: 2**
Historia de usuario: Gestión de usuario.	
Condiciones de ejecución: El administrador del sistema, en este caso el presidente debe autenticarse para poder gestionar a los usuarios del sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña USUARIOS 2. Ingresar a la opción MANTENIMIENTO. 3. Crear, editar, asignar o eliminar a cualquier usuario del sistema. 4. Clic en los botones respectivos: NUEVO, GUARDAR, EDITAR o ELIMINAR.	
Resultado esperado: Cambios actualizados correctamente.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 41

Caso de prueba – Registro de asociados

Caso de prueba	
Código: 3	**Nº Historia de usuario: 3**
Historia de usuario: Registro de asociados.	
Condiciones de ejecución: El secretario deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña ADMINISTRACIÓN. 2. Ingresar a la opción ASOCIADOS. 3. Ingresar los datos del asociado en los campos respectivos. 4. Generar y guardar el código de barras. 5. Clic en el botón GUARDAR.	
Resultado esperado: Se guardaron los datos del asociado.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 42

Caso de prueba – Gestión de asociados

Caso de prueba	
Código: 4	**Nº Historia de usuario: 4**
Historia de usuario: Gestión de asociados.	
Condiciones de ejecución: El secretario deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña ADMINISTRACIÓN. 2. Ingresar a la opción ASOCIADOS. 3. Buscar, editar, eliminar los datos del asociado. 4. Clic en el botón GUARDAR.	
Resultado esperado: Se actualizaron los datos del asociado.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 43

Caso de prueba – Gestión de asambleas

Caso de prueba	
Código: 5	**Nº Historia de usuario: 5**
Historia de usuario: Gestión de asambleas.	
Condiciones de ejecución: El secretario deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña ADMINISTRACIÓN. 2. Ingresar a la opción ASAMBLEAS. 3. Agregar una fecha y agenda de la asamblea. 4. Clic en el botón GUARDAR. 5. Buscar alguna asamblea. 6. Doble clic sobre la asamblea encontrada. 7. Clic en el botón EDITAR. 8. Modificar la fecha o la agenda de la asamblea. 9. Clic en el botón GUARDAR.	
Resultado esperado: Se guardó y actualizó correctamente la asamblea.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

4.2.3 Segunda iteración

En la tabla 44 se muestran de forma general las historias de usuario para la primera iteración y sus módulos respectivos.

Tabla 44

Historias de usuario

Número	Nombre	Módulo
6	Control de asistencia	Control
7	Consulta de asistencia	Consulta
8	Consulta de inasistencia	Consulta

En la tabla 45 se muestra de forma general las tareas de ingeniería para la segunda iteración.

Tabla 45

Tareas de ingeniería

Número de Tarea	Historia	Nombre de la Tarea
1	6	Diseño interfaz para el control de asistencia.
2	6	Creación de la base de datos para el control de asistencia.
3	7	Diseño interfaz para la gestión de asistencia.
4	8	Diseño interfaz para la consulta de inasistencia.

En las tablas 46- 49 se detallan cada una de las tareas

Tabla 46

Tareas de ingeniería 1 para historia de usuario 6

Tarea de Ingeniería	
Número de tarea:**1**	Número de historia:**6**
Nombre de tarea: Diseño interfaz para el control de asistencia.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 26/07/17	Fecha fin: 01/08/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se diseña la interfaz para el registro de asistencia de cada asociado a la asamblea.	

Tabla 47

Tareas de ingeniería 2 para historia de usuario 6

Tarea de Ingeniería	
Número de tarea:**13**	Número de historia:**6**
Nombre de tarea: Creación de la base de datos para el control de asistencia.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 01/08/17	Fecha fin: 09/08/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se creará la base de datos para almacenar información requerida de cada asamblea.	

Tabla 48

Tareas de ingeniería 3 para historia de usuario 7

Tarea de Ingeniería	
Número de tarea:**14**	Número de historia:**7**
Nombre de tarea: Diseño interfaz para la gestión de asistencia.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 09/08/17	Fecha fin: 16/08/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se diseña la interfaz para la gestión de asistencia de los asociados.	

Tabla 49

Tareas de ingeniería 4 para historia de usuario 8

Tarea de Ingeniería	
Número de tarea:**15**	Número de historia:**8**
Nombre de tarea: Diseño interfaz para la consulta de inasistencia.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio:16/08/07	Fecha fin: 23/08/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se diseña la interfaz para la consulta de las inasistencias de los asociados a las asambleas.	

4.2.3.1 Diseño

Se diseñaron los prototipos usando el software Balsamiq, como se muestran en las figuras 4.9 - 4.11.

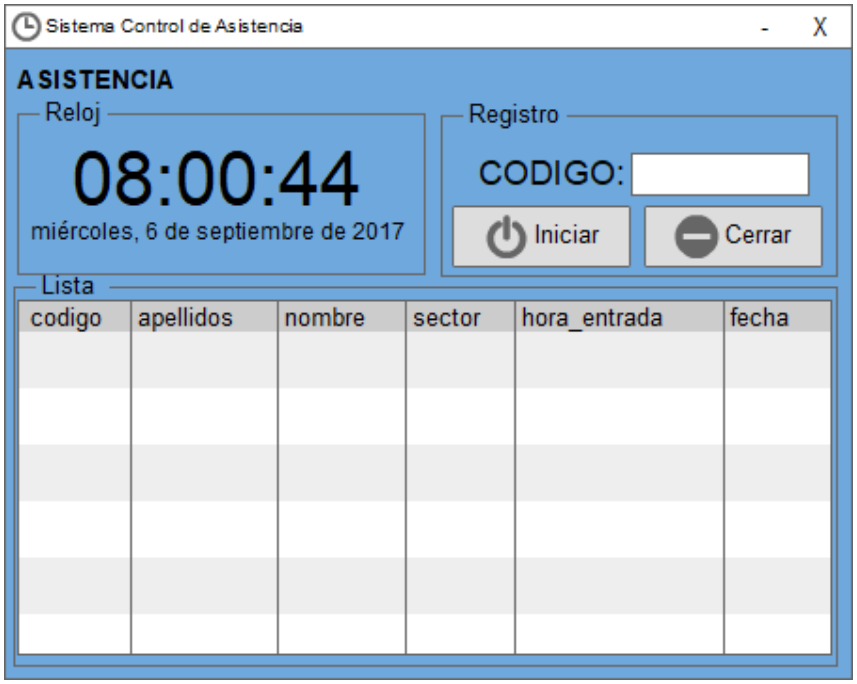


Fig. 4.9 Diseño – Control de asistencia

La Fig. 4.9 muestra el diseño de la interfaz para el control de asistencia de los asociados a las asambleas.

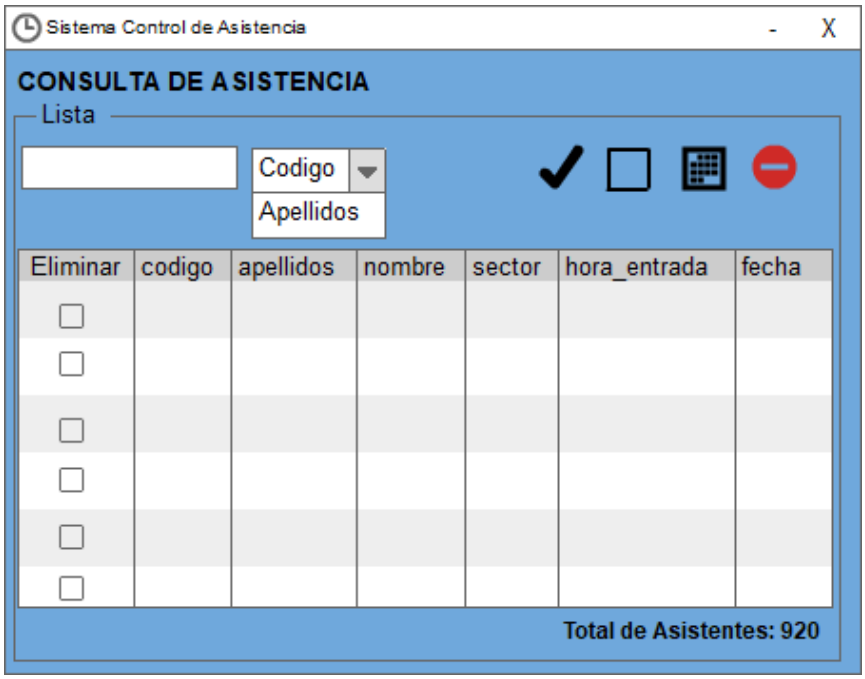


Fig. 4.10 Diseño – Consulta de asistencia

La Fig. 4.10 muestra el diseño de la interfaz para la consulta de asistencia de los asociados a las asambleas.

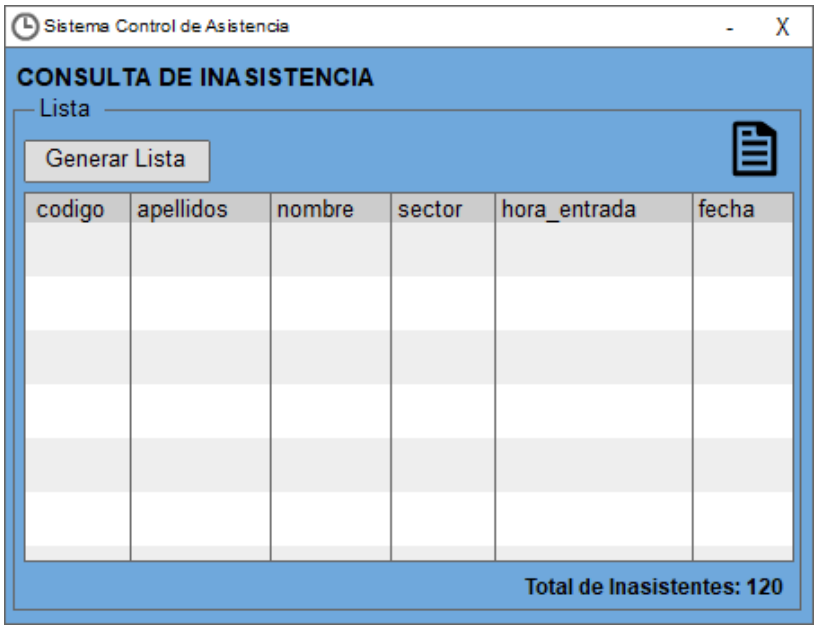


Fig. 4.11 Diseño – Consulta de inasistencia

La Fig. 4.11 muestra el diseño de la interfaz para generar y exportar la lista de inasistencia de los asociados a las asambleas.

4.2.3.2 Codificación

Una vez aceptados los prototipos por el cliente, se procede a la codificación (ver anexos 11-26), a continuación, se muestra los resultados en los siguientes pantallazos (Fig. 4.12 – 4.14).

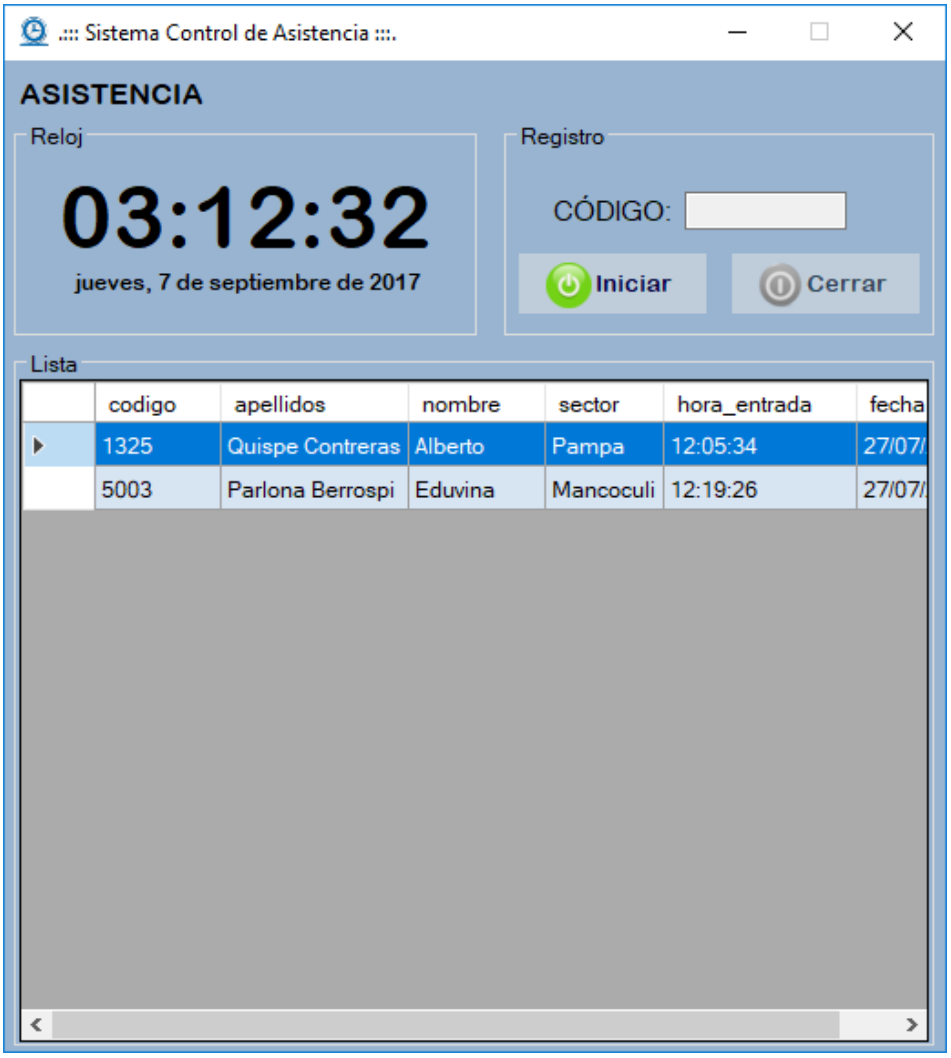


Fig. 4.12 Interfaz para asistencias

La Fig. 4.12 muestra la interfaz del formulario de asistencia del sistema, donde se registra las asistencias del asociado a la asamblea, cuando pasa su ficha por el lector de código de barras, caso contrario es ingresado manualmente al sistema por el vocal encargado.

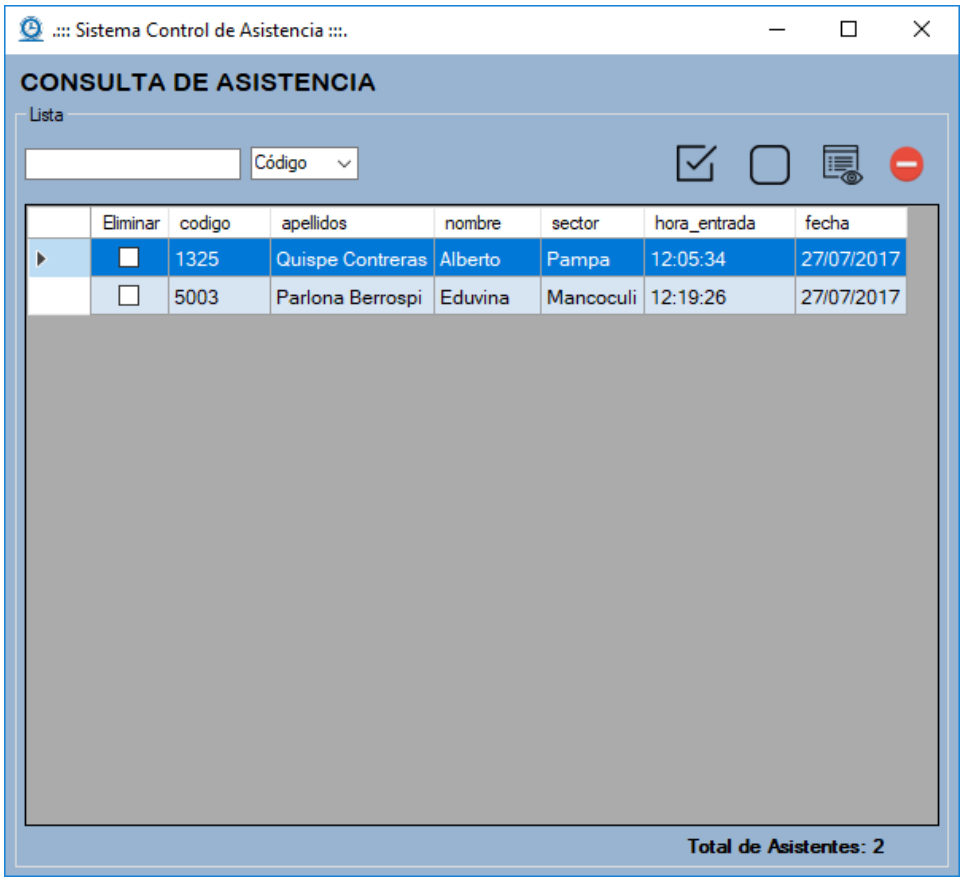


Fig. 4.13 Interfaz para consulta de asistencia

La Fig. 4.13 muestra la interfaz del formulario consulta de asistencia, donde se visualiza el total de asistentes así mismo se puede buscar por código o apellido la asistencia del asociado y también eliminar dicha lista.

	codigo	apellidos	nombre	direccion	sector	es
	1001	Nájera Meza	Francisco	Los Andes N° 941	Pampa	Ac
	1002	Batalla García	Pedro	Bolognesi N° 1015	Pampa	Ac
	1003	Flores Meza	Rómulo	Calle Unión S/N	Pampa	Ac
	1004	Flores Meza	Rómulo	Los Incas N° 1885	Pampa	Ac
	1005	Quispe de Pache...	Francisca	Av. 28 de Julio N...	Pampa	Ac
	1006	Nuñez Contreras	Enrique	Jr. 28 de Julio Nº ...	Pampa	Ac
	1007	Pariona Contreras	Leovigilda	Los Incas NMOSC	Pampa	Ac
	1008	Yarasca Chipana	Isaac	Los Incas N° 1390	Pampa	Ac
	1009	Meza Moreno	Dora Carmen	Bolognesi Nº 1048	Pampa	Ac
	1010	Meza Maldonado	Octavio	Ramón Castilla N...	Pampa	Ac
	1011	Chipana Miranda	Wilfredo	Ramón Castilla N...	Pampa	Ac
	1012	Zamudio Gaspar	Saúl	Ramón Castilla Nº	Pampa	Ac
	1013	Mercado Pastrana	Eugenia M.	Av. 28 de Julio N...	Pampa	Ac
	1014	García Pastrana	Eulalia	Los Andes Nº 1007	Pampa	Ac
	1015	Lavado Palacios	Clenio	Bolognesi N ° 139	Pampa	Ex
	1016	Meza Moreno	Celia Haydee	S. Chipana Nº 379	Pampa	Ac

Fig. 4.14 Interfaz para la consulta de inasistencia

La Fig. 4.14 muestra la interfaz del formulario consulta de inasistencia, donde se puede visualizar a los asociados que faltaron a las reuniones y luego extraer dicha lista a un archivo de Excel

4.2.3.3 Pruebas

Para validar la primera iteración, se aplicará las pruebas de aceptación, las cuales verifican que el sistema funcione de acuerdo a las historias de usuario. En las tablas 50 – 52 se muestran las siguientes pruebas realizadas:

Tabla 50

Caso de prueba – Control de asistencia

Caso de prueba	
Código: 6	**Nº Historia de usuario: 6**
Historia de usuario: Control de asistencia.	
Condiciones de ejecución: El vocal deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña CONTROL. 2. Ingresar a la opción ASISTENCIA. 3. Clic en el botón INICIAR. 4. Visualizar registro en el sistema. 5. Clic en el botón CERRAR.	
Resultado esperado: Se registró la asistencia en tiempo óptimo.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 51

Caso de prueba – Consulta de asistencia

Caso de prueba	
Código: 7	**Nº Historia de usuario: 7**
Historia de usuario: Consulta de asistencia.	
Condiciones de ejecución: Debe haber registros de asistencia y el secretario deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña CONSULTA 2. Ingresar a la opción ASISTENCIA. 3. Visualizar la lista de asistencia y el total. 4. Buscar por código o apellido la asistencia de algún asociado. 5. Eliminar la lista de asistencia.	
Resultado esperado: Se consultó y eliminó la lista de asistencia.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 52

Caso de prueba – Consulta de inasistencia

Caso de prueba	
Código: 8	**Nº Historia de usuario: 8**
Historia de usuario: Consulta de inasistencia.	
Condiciones de ejecución: Debe haber registros de asistencia y la tesorera deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña CONSULTA 2. Ingresar a la opción INASISTENCIA. 3. Clic en el botón GENERAR LISTA. 4. Visualizar lista de inasistencia.	
Resultado esperado: Se generó y visualizó la lista de inasistencia.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

4.2.4 Tercera iteración

En la tabla 53 se muestran de forma general las historias de usuario para la primera iteración.

Tabla 53

Historias de usuario

Número	Nombre	Módulo
9	Exportar datos a Excel.	Consulta
10	Generar reporte	Administración

En la tabla 54 se muestran de forma general las historias de usuario para la segunda iteración.

Tabla 54

Tareas de ingeniería

Número de Tarea	Historia	Nombre de la Tarea
1	9	Agregar la opción para exportar a Excel.
2	10	Agregar la opción para generar reporte.

En las tablas 55- 56 se detallan cada una de las tareas

Tabla 55

Tareas de ingeniería 1 para historia de usuario 9

Tarea de Ingeniería	
Número de tarea:**1**	Número de historia:**9**
Nombre de tarea: Agregar la opción para exportar a Excel.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 23/08/17	Fecha fin: 30/08/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se agrega la opción para poder exportar la lista de inasistencia a Excel.	

Tabla 56

Tareas de ingeniería 2 para historia de usuario 10

Tarea de Ingeniería	
Número de tarea:**2**	Número de historia:**10**
Nombre de tarea: Agregar la opción para generar reporte.	
Tipo de tarea: Desarrollo	Puntos Estimados: 0.5
Fecha inicio: 30/08/17	Fecha fin: 06/09/17
Programador responsable: Iván Diego Rivera Meza	
Descripción: Se agrega la opción para poder generar el reporte del padrón de asociados.	

4.2.4.1 Diseño

Se diseñaron los prototipos usando el software Balsamiq, como se muestran en la fig. 15.

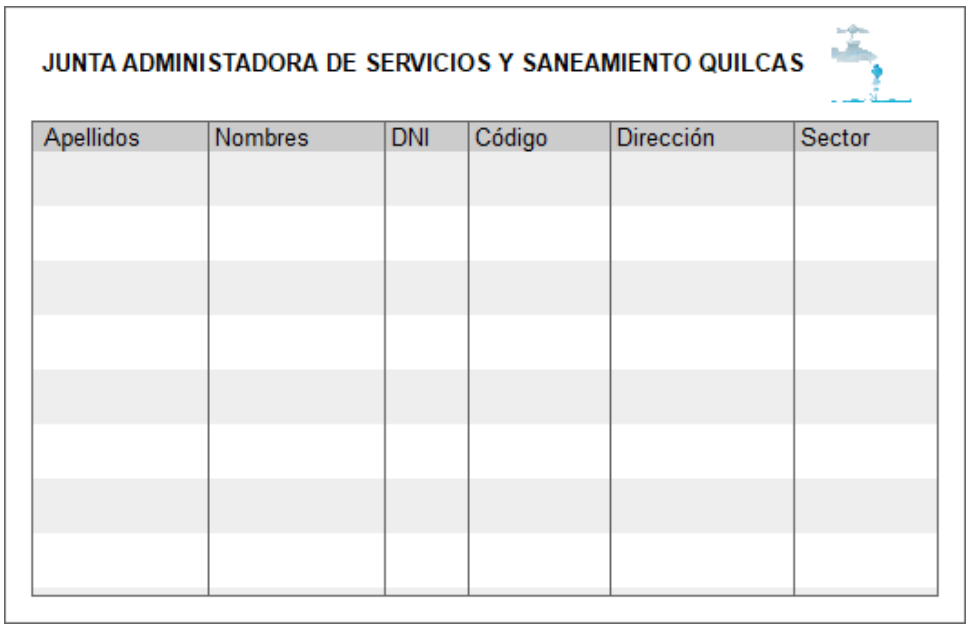


Fig. 4.15 Diseño – Reporte

La Fig. 4.15 muestra el diseño de la interfaz para el reporte del padrón general de asociados.

4.2.4.2 Codificación

Una vez aceptados los prototipos por el cliente, se procede a la codificación (ver anexos 11-26), a continuación, se muestra los resultados en el siguiente pantallazo (Fig. 4.16)

Código	DNI	Apellidos	Nombre	Dirección	Sector
1001	19960197	Nájera Meza	Francisco	Los Andes N° 941	Pampa
1002	0	Batalla García	Pedro	Bolognesi N° 1015	Pampa
1003	19958849	Flores Meza	Rómulo	Calle Unión S/N	Pampa
1004	19958849	Flores Meza	Rómulo	Los Incas N° 1885	Pampa
1005	19958966	Quispe de Pacheco	Francisca	Av. 28 de Julio N° 487	Pampa
1006	0	Nuñez Contreras	Enrique	Jr. 28 de Julio N° 613	Pampa
1007	19958899	Pariona Contreras	Leovigilda	Los Incas NMOSC	Pampa
1008	0	Yarasca Chipana	Isaac	Los Incas N° 1390	Pampa
1009	19920282	Meza Moreno	Dora Carmen	Bolognesi N° 1048	Pampa
1010	19960588	Meza Maldonado	Octavio	Ramón Castilla N° 557	Pampa
1011	10709658	Chipana Miranda	Wilfredo	Ramón Castilla N° 640	Pampa
1012	0	Zamudio Gaspar	Saúl	Ramón Castilla N°	Pampa
1013	0	Mercado Pastrana	Eugenia M.	Av. 28 de Julio N° 629	Pampa
1014	0	García Pastrana	Eulalia	Los Andes N° 1007	Pampa
1015	20544251	Lavado Palacios	Clenio	Bolognesi N ° 139	Pampa
1016	0	Meza Moreno	Celia Haydee	S. Chipana N° 379	Pampa
1017	19959827	Pariona Malpica	Román	Los Andes N° 1061	Pampa
1018	0	Pariona de Bendezú	Elcira	Calle Sucre N°	Pampa
1019	0	Comunidad Campesina		PITI	Pampa
1020	19960019	Avila Zamudio Vda. de Gavilán	Julia	Los Incas N° 1440	Pampa
1021	0	Daviran Pacheco	Alejandro	Los Incas N° 1603	Pampa
1022	19959335	Huaman de Pariona	Teresa	Los Andes N° 925	Pampa
1023	0	Calderón Chipana	Amadeo	Calle Unión S/N°	Pampa
1024	19959799	Avila Salvador	Vicente Anastacio	Daniel A. Carrión S/N°	Pampa

Fig. 4.16 Interfaz del reporte

La Fig. 4.16 muestra la interfaz para el reporte del padrón general de asociados de la Junta Administradora de Servicios de Saneamiento Quilcas, el cual luego se podrá imprimir o exportar a los archivos, PDF, Excel o Word.

4.2.4.3 Pruebas

Para validar la primera iteración, se aplicará las pruebas de aceptación, las cuales verifican que el sistema funcione de acuerdo a las historias de usuario. En las tablas 57 – 58 se muestran las siguientes pruebas realizadas.

Tabla 57

Caso de prueba – Exportar datos a Excel

Caso de prueba	
Código: 9	**Nº Historia de usuario: 9**
Historia de usuario: Exportar datos a Excel.	
Condiciones de ejecución: Debe haber registros de inasistencia y la tesorera deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña CONSULTA 2. Ingresar a la opción INASISTENCIA. 3. Clic en el botón GENERAR LISTA. 4. Visualizar lista de inasistencia. 5. Clic en el ícono de Excel para exportar.	
Resultado esperado: Se exportó la lista de inasistencia.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

Tabla 58

Caso de prueba – Generar reporte

Caso de prueba	
Código: 10	**Nº Historia de usuario: 10**
Historia de usuario: Generar reporte.	
Condiciones de ejecución: Debe haber datos de los asociados y el presidente deberá estar con la sesión iniciada en el sistema.	
Entradas/Pasos de ejecución: 1. Clic en la pestaña ADMINISTRACIÓN 2. Ingresar a la opción ASOCIADOS. 3. Clic en el ícono definido para generar reporte. 4. Visualizar reporte generado. 5. Imprimir o exportar el reporte generado.	
Resultado esperado: Se generó el padrón general de asociados.	
Evaluación de la prueba: La prueba concluyó satisfactoriamente.	

4.3 Construcción del sistema

Para la construcción del sistema se empleó la arquitectura de 3 capas en lenguaje C# con el entorno de trabajo Visual Studio 2013 y la base datos en SQL Server 2012 ambos en su versión gratuita. El sistema tiene la siguiente estructura, como se visualiza en la fig. 17, la interfaz del menú principal (Fig. 18) y la información (Fig. 19).

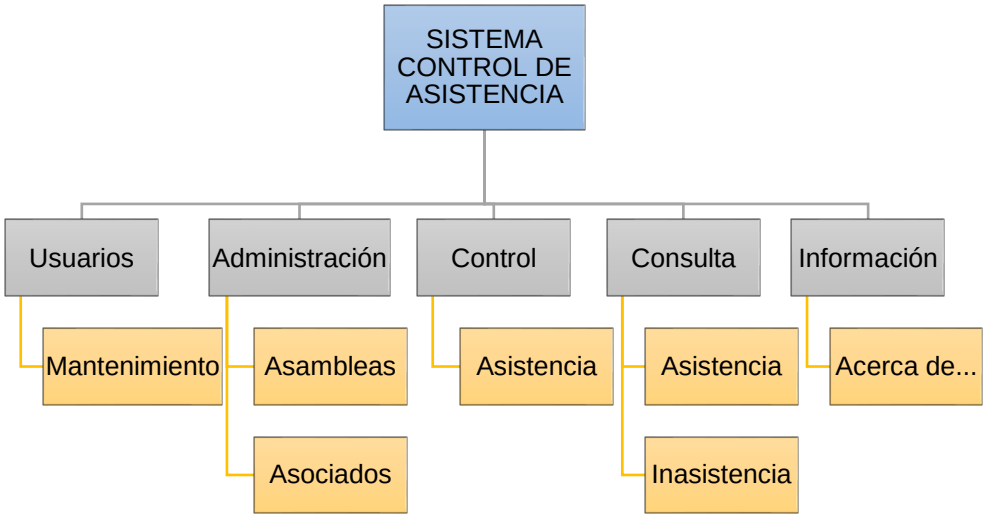


Fig. 4.17 Diagrama de navegación del sistema control de asistencia

La Fig. 4.17 muestra el diagrama de navegación del sistema control de asistencia que permite organizar los formularios.

Fig. 4.18 Interfaz del menú principal

La Fig. 4.18 muestra el menú principal del sistema control de asistencia, dentro de los cuales estarán los formularios desarrollados en cada módulo o pestaña.

Fig. 4.19 Interfaz de la información del sistema

La Fig. 4.19 muestra la interfaz de la información acerca del sistema.

4.3.1 Arquitectura del sistema

La arquitectura en 3 capas es separa la capa de datos de la capa de negocios y ésta a su vez de la capa presentación. En la figura 4.13 se muestra la arquitectura.

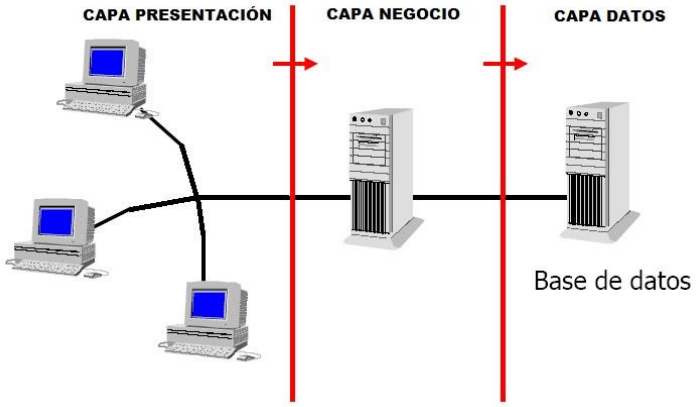


Fig. 4.20 Arquitectura en tres capas

La Fig. La Fig. 4.20 muestra es la estructura de la arquitectura en tres y capas.

- Capa Presentación: Es la responsable de la presentación visual de la aplicación. Podemos decir es la que se presenta al usuario, llamada también formulario o interfaz.

- Capa Negocio: Es la responsable de procesamiento que tiene lugar en la aplicación. Esta capa intermedia es la que conlleva capacidad de mantenimiento y reutilización.

- Datos: Esta capa se encarga de acceder a los datos, se debe usar la capa datos para almacenar y recuperar toda la información de sincronización del sistema.

4.3.2 Diseño de la base de datos

Para el diseño de la base de datos se utilizó el programa DBDesigner 4, como se muestra en la fig. 4.21.

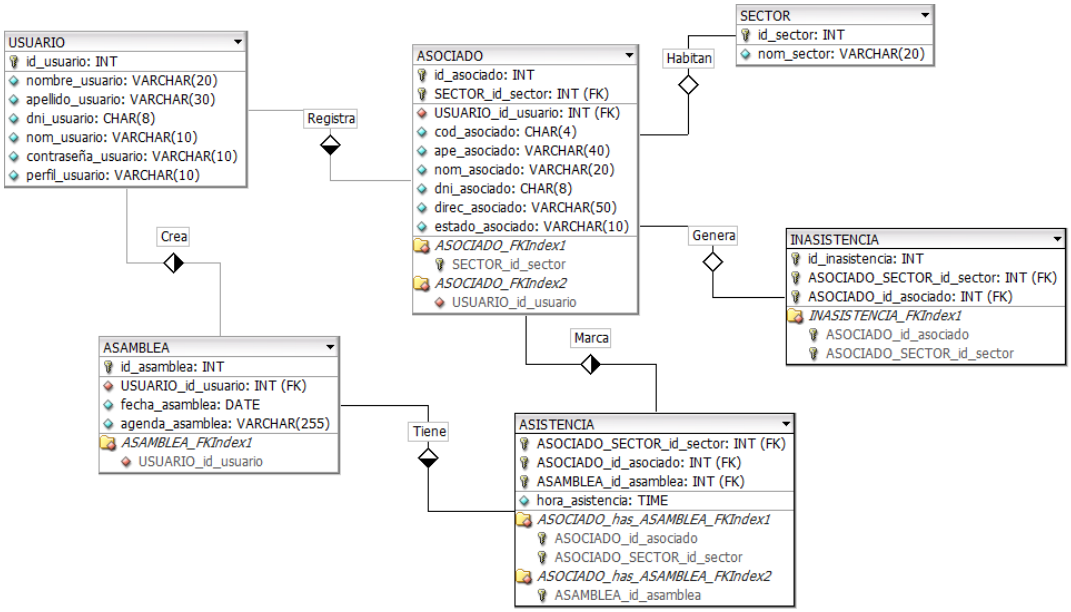


Fig. 4.21 Diseño de la base de datos

La Fig. La Fig. 4.21 muestra el diseño lógico de la base de datos

CAPÍTULO V
RESULTADOS Y DISCUSIÓN

5.1 Discusión de resultados

Usando la técnica de las entrevistas como medio de recolección de datos, se pudo obtener los requerimientos plasmado en las historias de usuario, el cual ayudó para el desarrollo del sistema, luego se procedió a validar la implementación de acuerdo a los requerimientos del sistema mediante la misma técnica con la directiva de la Junta Administradora de Servicios de Saneamiento Quilcas, en la tabla 59 se visualiza el resultado.

Tabla 59

Entrevista Grupal

Día	30 de agosto del 2017
Horario	13:00 – 17:00
Lugar de encuentro	Jr. Ramón Castilla - JASSQ
Actividades realizadas	Validarla implementación del sistema
Objetivo	Comprobar que cumpla con los requerimientos establecidos para poder implementar.
Resultado	El sistema se implementó con éxito
Participantes	Iván Diego Rivera Meza, Félix Cárdenas Apolinario (presidente), Paola Roca Inga (tesorera), Pedro Apolinario Astucuri (secretario), Francisca Quispe de Pacheco (vocal).

Se aplicó la misma técnica a diez asociados, luego de implementar el sistema, en las tablas 60 – 69 se muestra el resultado

Tabla 60

Entrevista 01

Entrevista 01 – E01			
a) Identificación de reunión			
Tema	Tiempo de registro de asistencia	**Cliente**	JASSQ
Alcance	Relevamiento de información.		
Fecha	13/09/2017	**Horario**	13:30 – 17:30
Lugar	Jr. Ramón Castilla - JASSQ		

b) Participantes			
#	**Nombre**	**ID**	**Cargo**
1	Isaac Yarasca Chipana	EC	Asociado
2	Iván Diego Rivera Meza	IR	Tesista

c) Temas tratados			
Tema	**#**	**Plant. Por**	**Detalles**
¿El sistema disminuyó el tiempo de registro de asistencia?	1	IY/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema.

Tabla 61

Entrevista 02

Entrevista 02 – E02					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Freddy E. Porras Rodríguez	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	FP/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema y propone más lectores de código de barras.		

Tabla 62

Entrevista 03

Entrevista 03 – E03					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Francisco Nájera Meza	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	FN/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido, a la vez sugiere que haya más control en la puerta de ingreso.		

Tabla 63

Entrevista 04

Entrevista 04 – E04					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Aparicia Rodríguez Suazo	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	AR/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido y que debería implementarse mejoras en cuanto a la infraestructura tecnológica de la entidad.		

Tabla 64

Entrevista 05

Entrevista 05 – E05					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Nicolás Huamán Salinas	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	NHC/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema, a la vez sugiere que compre nuevos equipos de cómputo.		

Tabla 65

Entrevista 06

Entrevista 06 – E06					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Oswaldo Chipana Rodríguez	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	OC/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema.		

Tabla 66

Entrevista 07

Entrevista 07 – E07					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Dora Meza Moreno	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	DM/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema, plantea que se implemente un sistema de caja.		

Tabla 67

Entrevista 08

Entrevista 08 – E08					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Miguel C. Mayorga Nájera	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	MM/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema.		

Tabla 68

Entrevista 09

Entrevista 09 – E09					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Elvira Córdova Quispe	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	EC/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema, a la vez indica que la gente debería adaptarse a los cambios.		

Tabla 69

Entrevista 10

Entrevista 10 – E10					
a) Identificación de reunión					
Tema	Tiempo de registro de asistencia		**Cliente**	JASSQ	
Alcance	Relevamiento de información.				
Fecha	13/09/2017		**Horario**	13:30 – 17:30	
Lugar	Jr. Ramón Castilla - JASSQ				
b) Participantes					
#	**Nombre**	**ID**	**Cargo**		
1	Severíano Centraras Osores	EC	Asociado		
2	Iván Diego Rivera Meza	IR	Tesista		
c) Temas tratados					
Tema	**#**	**Plant. Por**	**Detalles**		
¿El sistema disminuyó el tiempo de registro de asistencia?	1	SC/IR	El asociado afirma que el tiempo de registro de asistencia ha disminuido con la implementación del sistema.		

Se observó que los asociados están de acuerdo que la implementación del sistema disminuyó el tiempo de registro de asistencia, en la tabla 70 se muestra el resumen:

Tabla 70

Resumen de las entrevistas

Asociado	Sector	¿El sistema disminuyó el tiempo de registro de asistencia?
Isaac Yarasca Chipana	Pampa	Sí
Freddy E. Porras Rodríguez	27 de mayo	Sí
Francisco Nájera Meza	Unuimarca	Sí
Aparicia Rodríguez Suazo	Santa Cruz	Sí
Nicolás Huamán Salinas	Patac	Sí
Oswaldo Chipana Rodríguez	Llacta	Sí
Dora Meza Moreno	Centenario	Sí
Miguel C. Mayorga Nájera	Pampa	Sí
Elvira Córdova Quispe	27 de mayo	Sí
Severíano Centraras Osores	Patac	Sí

5.2 Pruebas del sistema

Para realizar las pruebas del sistema según los requerimientos de los usuarios finales, se aplicó la herramienta cheklist, en las tablas 71-77 se muestra los resultados.

Tabla 71

Cheklist 01

Módulo	Usuario/Mantenimiento	Usuario	Félix Cárdenas	
Perfil	Presidente/Administrador	Versión	1.1	
Procesos		SI	NO	Observación
Ingresar al sistema con el usuario y contraseña creado.		X		Su cumplió correctamente
Ingresar al módulo Usuario/Mantenimiento		X		Su cumplió correctamente
Agregar Usuario		X		Su cumplió correctamente
Asignar Perfil		X		Su cumplió correctamente
Editar Usuario		X		Su cumplió correctamente
Eliminar Usuario		X		Su cumplió correctamente

Tabla 72

Cheklist 02

Módulo	Administración/Asociados	Usuario	Pedro Astucuri	
Perfil	Secretario	Versión	1.1	
Procesos		SI	NO	Observación
Ingresar al sistema con el usuario y contraseña creado.		X		Su cumplió correctamente
Ingresar al módulo Administración/Asociados		X		Su cumplió correctamente
Agregar Asociado		X		Su cumplió correctamente
Generar Código de barras del Asociado		X		Su cumplió correctamente
Editar Asociado		X		Su cumplió correctamente
Eliminar Asociado		X		Su cumplió correctamente

Tabla 73

Cheklist 03

Módulo	Administración/Asambleas	Usuario	Pedro Astucuri	
Perfil	Secretario	Versión	1.1	
Procesos		SI	NO	Observación
Agregar Asamblea		X		Su cumplió correctamente
Editar Asamblea		X		Su cumplió correctamente
Buscar Asamblea		X		Su cumplió correctamente

Tabla 74

Cheklist 04

Módulo	Consulta/Asistencia	Usuario	Pedro Astucuri	
Perfil	Secretario	Versión	1.1	
Procesos		SI	NO	Observación
Visualizar la lista de Asistencia		X		Su cumplió correctamente
Realizar la búsqueda por código y apellidos de los Asistentes		X		Su cumplió correctamente
Eliminar lista de Asistencia		X		Su cumplió correctamente

Tabla 75

Cheklist 05

Módulo	Control/Asistencia	Usuario	Dora Meza	
Perfil	Vocal	Versión	1.1	
Procesos		SI	NO	Observación
Ingresar al sistema con el usuario y contraseña creado.		X		Su cumplió correctamente
Ingresar al módulo Control/Asistencia		X		Su cumplió correctamente
Apertura de Asistencia		X		Su cumplió correctamente
Registrar Asistencia con el lector de código de barras		X		Su cumplió correctamente
Registrar Asistencia con el teclado		X		Su cumplió correctamente
Visualizar el registro de Asistencia		X		Su cumplió correctamente
Finalizar el registro de Asistencia		X		Su cumplió correctamente

Tabla 76

Cheklist 06

Módulo	Consulta/Inasistencia	Usuario	Paola Inga	
Perfil	Tesorera	Versión	1.1	
Procesos		SI	NO	Observación
Ingresar al sistema con el usuario y contraseña creado.		X		Su cumplió correctamente
Ingresar al módulo Consulta/Inasistencia		X		Su cumplió correctamente
Generar la lista de Inasistencia		X		Su cumplió correctamente
Visualizar la lista de Inasistencia		X		Su cumplió correctamente
Exportar a Excel la lista de Inasistencia		X		Su cumplió correctamente

Tabla 77

Cheklist 07

Módulo	Administración/Asociados	Usuario	Félix Apolinario	
Perfil	Presidente	Versión	1.1	
Procesos		SI	NO	Observación
Ingresar al sistema con el usuario y contraseña creado.		X		Su cumplió correctamente
Ingresar al módulo Administración/Asociados		X		Su cumplió correctamente
Generar Reporte de Asociados		X		Su cumplió correctamente
Visualizar Reporte de Asociados		X		Su cumplió correctamente
Exportar a un archivo en Excel, Word y PDF.		X		Su cumplió correctamente

CONCLUSIONES

1. Se desarrolló e implementó un sistema de código de barras, el cual optimizó el control de asistencia de los asociados a las asambleas usando el código de barras y aplicando la metodología ágil XP (eXtreme Programming).

2. La interfaz diseñada para el padrón general, mejoró la administración del mismo, garantizando la confidencialidad, integridad y disponibilidad de la información de todos los asociados de la Junta Administradora de Servicios de Saneamiento Quilcas.

3. Se logró desarrollar un procedimiento para el registro de asistencia, con el uso del lector de código de barras, ahorrando tiempo tanto de los colaboradores que supervisan las asistencias y de los asociados que asistente a las asambleas.

4. Al generar la lista de inasistencia de forma automática, se logró que el cobro de las multas sea confiable y en tiempo propicio, de esta manera se asegura el ingreso económico para la entidad por dicho concepto.

RECOMENDACIONES

1. Hacer un mantenimiento preventivo al equipo de cómputo donde está instalado el sistema, con fechas establecidas, del mismo modo se recomienda implementar un servidor y estructura de red respectiva, para garantizar la escalabilidad y usabilidad del sistema.

2. Mantener actualizado los datos del padrón general, asegurando así una correcta administración de los asociados, asimismo realizar un backup de la base de datos, con el fin de tener un plan de contingencia ante cualquier eventualidad.

3. Se recomienda que haya más de una lectora de código de barras para el registro de asistencia, de esa manera se reducirá aún más el tiempo de ingreso a las asambleas.

4. La lista de inasistencia generada debe ser exportada a un archivo en Excel el mismo día de la asamblea por la tesorera y que solo ella tenga acceso a dicha información para que el cobro de las multas sea de manera transparente.

REFERENCIAS BIBLIOGRÁFICAS

[1] Hoyo Cordero Luis Rodrigo y Arteaga Rico Edgar Antonio: “El Código de Barras”. Centro Universitario Anglo Mexicano. México. Preparatoria Categoría Científica Físico Matemáticas Investigación Bibliográfica.

[2] L. Hernández y A. Martín: “Codificación de información mediante código de barras”. Departamento de Tratamiento de la Información y Codificación, Instituto de Física Aplicada, Consejo Superior de Investigaciones Científicas, Departamento de Matemáticas Aplicada, Universidad de Salamanca. Bol. Soc. Esp. Mat. Apl. nº 27(2004),28-48.

[3] Ing. Bernal Picado Arguello, Consultor/Asociado CICAP: “Memoria: Proyecto Implementación del Sistema de Código de Barras en el Sector Público”. Universidad de Costa Rica, Centro de Investigación y Capacitación en Administración Pública (CICAP), GS1 Costa Rica.

[4] Olga Marina Morales García: “El Código de Barras y su Aplicación en Bibliotecas Universitarias de la Ciudad de Guatemala”. Universidad de San Carlos de Guatemala, Facultad de Humanidades, Escuela de Bibliotecología. Guatemala noviembre de 2001.

[5] Adela Sandoval Sánchez: “Propuesta de Diseño de Implementación del Sistema de Código de Barras en el Departamento de Registros Médicos y Servicios de Apoyo al Diagnóstico en el Hospital San Juan de Dios”. Especialidad en Administración de Servicios de Salud, Programa de Gerencia Moderna y Gestión del Cambio en Salud. San José, Costa Rica, junio, 2008.

[6] Amaro Calderón Sarah Dámaris y Valverde Rebaza, Jorge Carlos:” Metodologías Ágiles”. Universidad Nacional de Trujillo, Facultad de Ciencias Físicas y Matemáticas, Escuela de Informática. Trujillo – Perú 2007.

[7] David Eduardo Rojas Ventura y Miguel Ángel Vidal Gonzales: “Aplicativo móvil para la asistencia de pacientes con Alzheimer en su Fase Inicial”. Escuela Profesional de Ingeniería de Computación y Sistemas. Universidad San Martín de Porres. Lima 2015.

[8] Villafuerte Huincho, Franklin “Eficiencia operativa de la gestión de planillas mediante el software Praxis – GL en la Municipalidad Provincial de Concepción”. Facultad de Ingeniería de Sistemas. Universidad del Centro del Perú. Huancayo –Perú, 2014.

[9] Mendoza Ricaldi, Luis Ángel: “Implementación de software para el registro y procesamiento de atenciones de salud en las actividades de responsabilidad social Caso – Mina Corihuarmi.” Facultad de Ingeniería de Sistemas. Universidad del Centro del Perú. Huancayo – Perú, 2014.

[10] José Carlos Acosta Bravo: “Sistema de Información basado en recomendaciones para mejorar el rendimiento del cultivo de papa en la región Puno aplicando metodología Ágil XP”. Universidad Nacional Mayor de San Marcos. Facultad de Ingeniería de Sistemas e Informática. EAP de Ingeniería de Sistemas. Lima- Perú, 2016.

[11] A. Arraz Ramonet. Administración de datos y archivos por computadora, 2a. Edición, 1994

[12] A. Orjuela Duarte, M. Rojas C. Las metodologías de desarrollo ágil como una oportunidad para la Ingeniería del software educativo. Vol. 5 No.2. junio, 2008.

[13] J. Conesa Caralt, A. Rius Gavidia, J. Ceballos Villach, D. Gañán Jiménez. Introducción a .NET, 1era Edición, Barcelona, España, Editorial UOC, 2010.

[14] S.D. Moquillaza Henriquez, H.V. Huerta, L. Guerra Grados. Programación en N capas. Revista de investigación de sistemas e informática, vol. 7, N.º 2, julio - diciembre 2010.

[15] V. Nevado Cabello, Introducción a las bases de datos relacionales, Madrid, España: Editorial Visión Libros.

[16] A. J. Oppel, R. Sheldon, Fundamentos de SQL, 3ra. Edición, México, México: The McGraw – Educación, 2010.

[17] B. Pérez Lamancha, Proceso de testing funcional independiente, Maestría, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 2006.

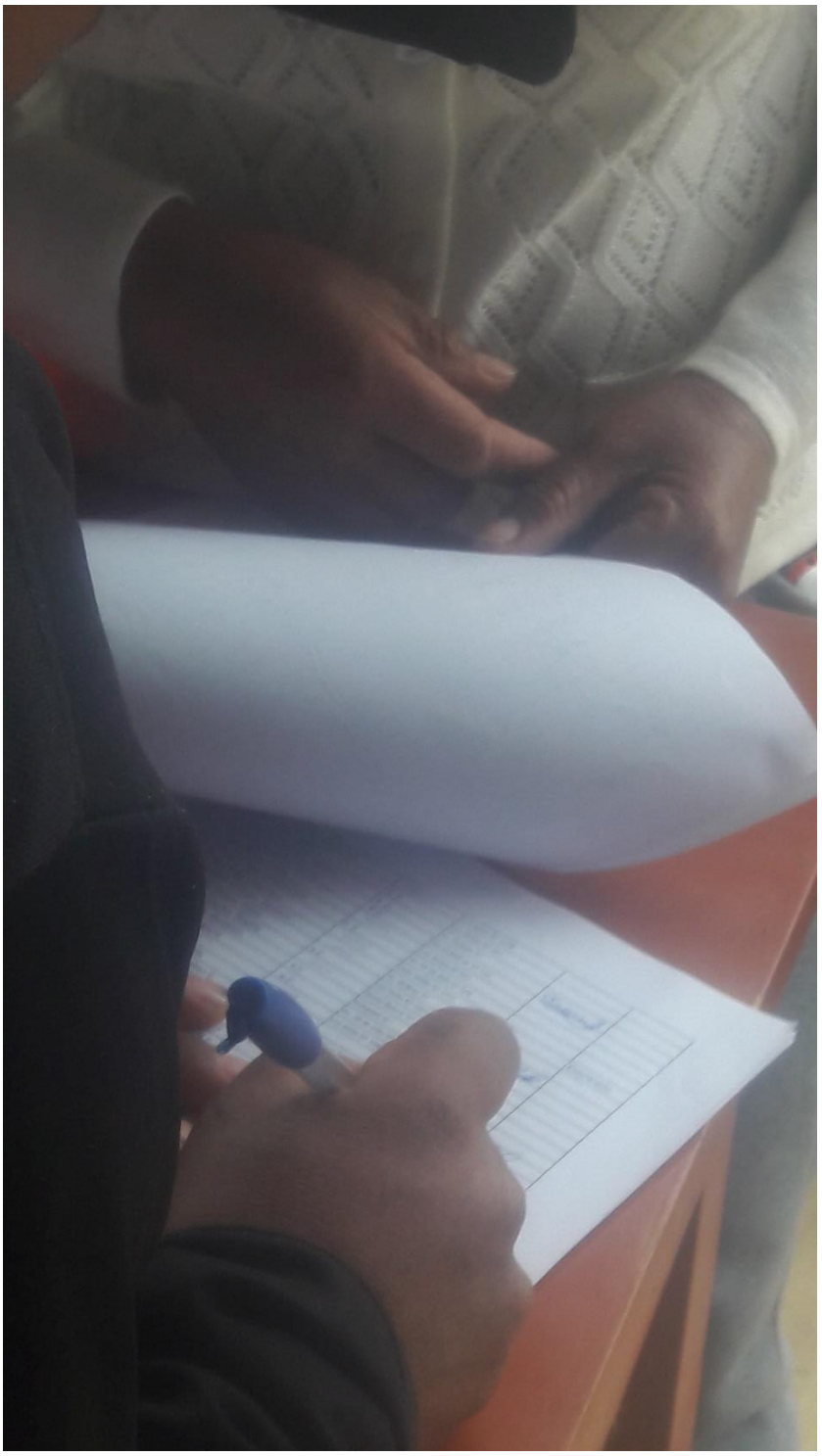
[18] R. Agüero, R. Montalvo, M. Montes, R. M. Valle, N. Vidalón, Manual de organización y gestión de las juntas administradoras de servicios de saneamiento, 2da Edición, Perú, Asociación SER, 2005.

[19] C. de la Cruz Casaño, Metodología de la investigación tecnológica en ingeniería, Revista Ingenium Vol.1 (1), pp 43-46, 2016.

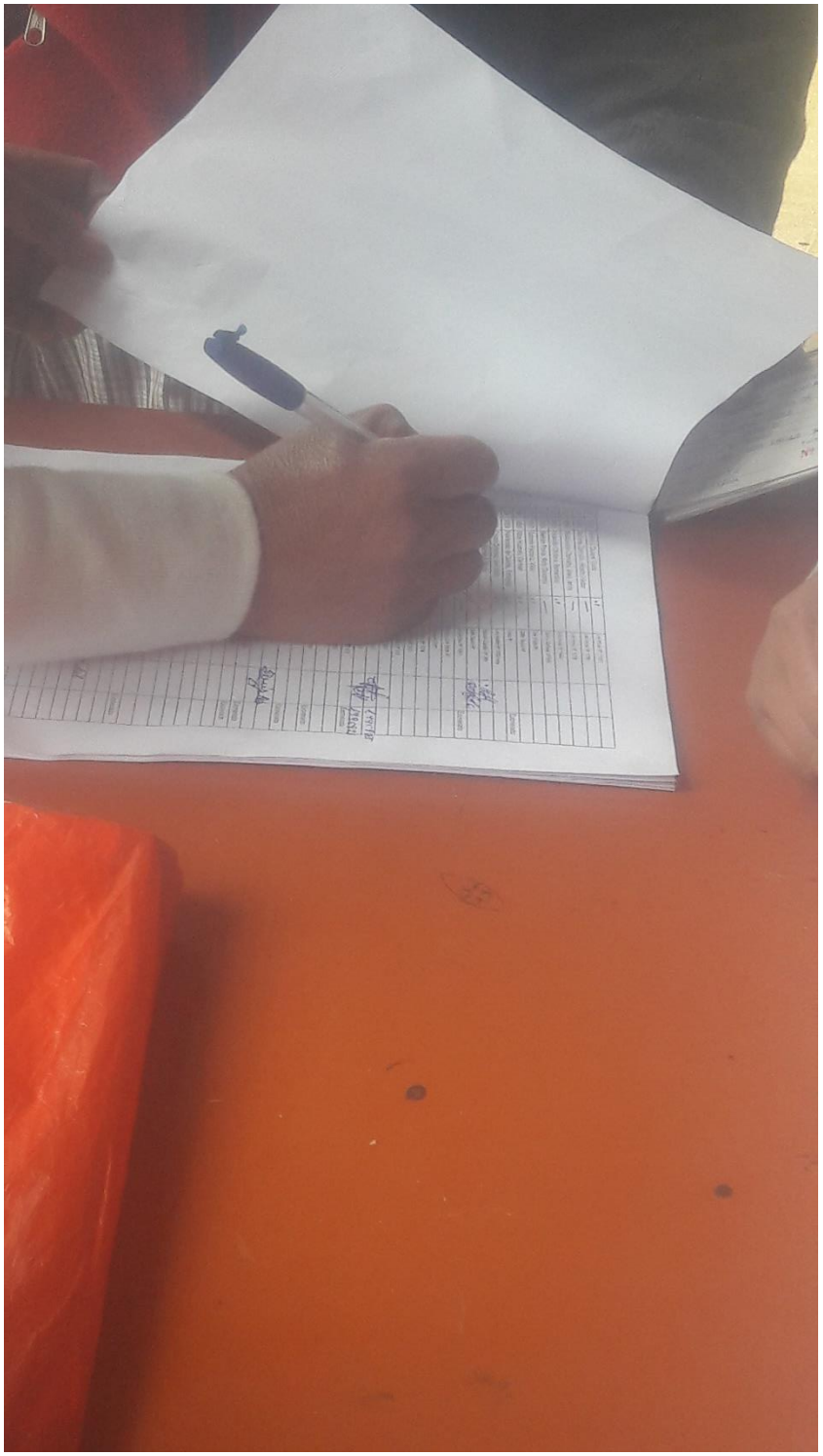
[20] S. Meléndez, M. Gaitán, N. Pérez. Metodología ágil de desarrollo de software programación extrema, Sistema web de evaluación al desempeño docente UNAN – Managua, empleando la metdología ágil programación extrema, Título, Facultad de Ciencias e Ingeniería, Universidad Nacional Autonoma de Nicaragua, Managua, 2016.

ANEXOS

Anexo 1 – Cola para ingresar a la asamblea.



Anexo 2 Padrón general de asociados



Anexo 3 Firma del padrón al ingresar y salir de la asamblea.

Tabla	Columna	Tipo	Longitud	Descripción
Asociado	Id_asociado	Int		Identificador del asociado
	cod_asociado	Char	4	Código de inscripción del asociado
	ape_asociado	Varchar	40	Apellido del asociado
	nom_asociado	Varchar	20	Nombre del asociado
	dni_asociado	Char	8	DNI del asociado
	direc_asociado	Vachar	50	Dirección del asociado
	estado_asociado	Varchar	10	Estado activo o exonerado del asociado
Asistencia	Id_asociado	Int		Llave foránea asociado
	Id_asamblea	Int		Llave foránea de la asamblea
	Hora_asistencia	Time		Hora de entrada para la asistencia
Sector	Id_sector	Int		Identificador del sector
	nom_sector	Varchar	20	Nombre del sector
Asamblea	Id_asamblea	Int		Identificador de la asamblea
	fecha_asamblea	Date		Fecha que se realizara la asamblea
	agenda_asamblea	Varchar	255	Agenda de la asamblea
Inasistencia	Id_inasistencia	Int		Identificador de la inasistencia
	Id_asociado	Int		Llave foránea asociado
	Id_sector	Int		Llave foránea sector
Usuario	Id_usuario	Int		Identificador del usuario
	nombre_usuario	Varchar	20	Nombre del usuario del sistema.
	apellido_usuario	Varchar	30	Apellido del usuario del sistema
	dni_usuario	Char	8	DNI de usuario del sistema
	nom_usuario	Varchar	10	Nombre de usuario para acceder al sistema.
	contraseña_usuario	Varchar	10	Contraseña de usuario para acceder al sistema.
	perfil_usuario	Varchar	10	Privilegio dentro del sistema.

Anexo 4 Diccionario de datos

Código	Apellidos	Nombre	DNI	Sector	Estado
1001	Nájera Meza	Francisco	19960197	Pampa	Activo
1002	Batalla Quispe	Jorge		Pampa	Activo
1003	Flores Meza	Rómulo	19958849	Pampa	Activo
1004	Flores Meza	Rómulo	19958849	Pampa	Activo
1005	Quispe de Pacheco	Francisca	19958966	Pampa	Activo
1006	Nuñez Contreras	Enrique	41594781	Pampa	Activo
1007	Pariona Contreras	Leovigilda	19958899	Pampa	Activo
1008	Yarasca Chipana	Isaac	19958845	Pampa	Activo
1009	Meza Moreno	Dora Carmen		Pampa	Activo
1010	Meza Maldonado	Octavio	19960588	Pampa	Activo
1011	Chipana Miranda	Wilfredo	10709658	Pampa	Activo
1012	Zamudio Gaspar	Saúl	42413933	Pampa	Activo
1013	Mercado Pastrana	Eugenia M.	19960856	Pampa	Activo
1014	García Pastrana	Eulalia		Pampa	Activo
1015	Lavado Palacios	Clenio	20544251	Pampa	Exonerado
1016	Meza Moreno	Celia Haydee	19960856	Pampa	Activo
1017	Pariona Malpica	Román	19959827	Pampa	Activo
1018	Pariona de Bendezú	Elcira	19960815	Pampa	Exonerado
1019	Comunidad Campesina			Pampa	Activo
1020	Avila Zamudio Vda. de Gavilán	Julia	19960019	Pampa	Activo
1021	Daviran Pacheco	Alejandro	19960460	Pampa	Activo
1022	Huaman de Pariona	Teresa	19959335	Pampa	Exonerado
1023	Calderón Chipana	Amadeo		Pampa	Exonerado
1024	Avila Salvador	Vicente Anastacio	19959799	Pampa	Activo
1025	Huamán Quispe	Mario Lorenzo	20048653	Pampa	Activo
1026	Gamarra Chipana	Victoria	19959692	Pampa	Activo
1027	Pariona Vda. de Vivanco	Digna	19959576	Pampa	Exonerado
1028	Bernabé Apolinario	Pedro	19958852	Pampa	Activo
1029	Bernabé Apolinario	Pedro	19958852	Pampa	Activo
1030	CETI 27 DE MAYO			Pampa	Activo
1031	Zarate Carhuachin	Velia Victoria	19958988	Pampa	Exonerado
1032	Tiza Quispe	Leónidas	19959797	Pampa	Activo
1033	Wincho Contratas	Gertrudis	19961199	Pampa	Activo
1034	Casallo Dávila	Margarita		Pampa	Exonerado
1035	Llacua Pomayay	Carlos	20071864	Pampa	Activo
1036	Laurente de Chipana	Elena	19960348	Pampa	Activo
1037	Palacios de Berrospi	Brigida	19959238	Pampa	Activo
1038	Rodríguez Quispe	Basilia		Pampa	Exonerado
1039	Palacios Vda. de Miranda	Eulalia	19959787	Pampa	Exonerado
1040	Zarate Malpica	Luis	19959663	Pampa	Activo
1041	Suazo Chávez	Eustaquia	19959086	Pampa	Activo
1042	Huacachi Quispe	Maria	23213886	Pampa	Activo

1043	Mantari Inga	Narciso		Pampa	Exonerado
1044	Pacheco Calderon	Yuliana	19977368	Pampa	Activo
1045	Arroyo Aguilar	Norma	40591755	Pampa	Activo
1046	Avila Sánchez	Carlos	19959318	Pampa	Exonerado
1047	Avila Chuquimantari	Dionicia	42976768	Pampa	Activo
1048	Martines Amau	Evy Yanina	47511129	Pampa	Activo
1049	Ramos Cordova	Almenda	19958820	Pampa	Exonerado
1050	Meza Quispe	Angélica	47459332	Pampa	Activo
1051	Roqueña Vda. de Pariona	Viviana		Pampa	Activo
1052	Miranda Quispe	Dario	19960042	Pampa	Activo
1053	Salomé Zamudio	Alberto Héctor	19961008	Pampa	Activo
1054	Maldonado Zamudio	Jossy Jenny	40797149	Pampa	Activo
1055	Huamán Córdova	Bernardino	19960328	Pampa	Activo
1056	Beltrán Ponce	Nelly Giovanna	20084523	Pampa	Activo
1057	Salomé Pacheco	Vidal	19960346	Pampa	Activo
1058	Allca Huamán	Carmen	19960333	Pampa	Activo
1059	Fernández de Ccente	Fortunata	19960489	Pampa	Exonerado
1060	Crispin Ordoñez	Jeni Edith	43822524	Pampa	Activo
1061	Huamán Bernabé	Martin	19960207	Pampa	Activo
1062	Gamarra Pariona	Carmen	19958858	Pampa	Activo
1063	Quispe de Contreras	Victoria		Pampa	Exonerado
1064	Arellano de Inga	Juana	19959286	Pampa	Activo
1065	Suazo Rodriguez	Benjamina	19959144	Pampa	Activo
1066	Contreras Galindo	Jorge	20084768	Pampa	Activo
1067	Contreras Antonio	Alicia	41398705	Pampa	Activo
1068	Inocente de Davirán	Fortunata	19960357	Pampa	Activo
1069	Chuquimantari López	Gloría	19960857	Pampa	Activo
1070	Avila Miranda	Miriam		Pampa	Activo
1071	Avila Calderón	Roque	10656682	Pampa	Activo
1072	Contreras de Núñez	Áurea	19959997	Pampa	Exonerado
1073	Miranda Tiza	Alfredo Vidal	19958968	Pampa	Activo
1074	Ramos Hurtado	Lourdes	19960017	Pampa	Activo
1075	Veliz Suazo	Pedro	19960199	Pampa	Exonerado
1076	Gamarra Vda. de Avila	Celina	19960530	Pampa	Activo
1077	Ponce Laurente	Paúl	6578324	Pampa	Exonerado
1078	Pariona Contreras	Jesús Bárbara	19918915	Pampa	Activo
1079	Chipana Colonio	Domitila		Pampa	Activo
1080	Machuca Cahuana	Filomena	19960510	Pampa	Exonerado
1081	Veliz Suazo	Eutropia Cristina	20412170	Pampa	Exonerado
1082	Avila Galindo	Luz Abencia	19960887	Pampa	Activo
1083	Contreras Cárdenas	Francisco	19959773	Pampa	Activo
1084	Dávila de Chuquimantari	Cleria	19960184	Pampa	Activo
1085	Contreras Colonio	Herminia	19980198	Pampa	Activo

1086	Medina de Vargas	Cecilia	19960229	Pampa	Activo
1087	Zamudio Chuquimantari	Enma J.	19959548	Pampa	Activo
1088	Chipana Sánchez	Ermogenes G.	19961042	Pampa	Activo
1089	Rodríguez Martínez	Saturnino	19960023	Pampa	Exonerado
1090	Meza Yarasca	Alicia	19960547	Pampa	Activo
1091	Quispe Vda. de Calderón	Sabina		Pampa	Activo
1092	Tiza Pariona	Elizabeth Jesica	41892442	Pampa	Activo
1093	Contreras Santa	Cruz Magno	19959548	Pampa	Activo
1094	Zarate Contreras	Epifanía	19959366	Pampa	Activo
1095	Pariona Zamudio	Victoria		Pampa	Activo
1096	Quispe Rojas	Fortunata		Pampa	Activo
1097	Samaniego Zacarías	Terecita	41325540	Pampa	Activo
1098	Dávila de Palacios	Elisa		Pampa	Exonerado
1099	Tiza Dávila	Juan Hildebrando	19959204	Pampa	Activo
1100	Palomino Vda. de Pariona	Marina	19959004	Pampa	Activo
1101	Avila Salvador	Marcelino	19959970	Pampa	Activo
1102	Pacheco de Oré	Rebeca	19959503	Pampa	Activo
1103	Pacheco Daviran	Sabina	19960249	Pampa	Activo
1104	Pacheco Davirán	Prudencio	19959343	Pampa	Activo
1105	Davirán Osares	Francisco Ricardo	19959595	Pampa	Activo
1106	Ponce Quispe	Teófilo Cayo	19959854	Pampa	Activo
1107	Alcoser de Quispe	María I.		Pampa	Activo
1108	Soto Meza	Hilario	19958803	Pampa	Activo
1109	Sedaño Pacheco	Teodoro	19959577	Pampa	Activo
1110	Loroña Papuico	Gavina	19958523	Pampa	Activo
1111	Soto Meza	Hilario	19958803	Pampa	Activo
1112	Chipara Inga	Alfredo Cirilo	19958864	Pampa	Activo
1113	Suazo Avila	Francisco	19959155	Pampa	Exonerado
1114	Granados Tenicela	Teodoro	20661667	Pampa	Activo
1115	Zamudio Quispe	Rossi	19961020	Pampa	Activo
1116	Quispe de Ospina	Rogata	20401369	Pampa	Activo
1117	Ruelas Coalla	Benjamín	19959507	Pampa	Activo
1118	Gago Huamán	María	19954870	Pampa	Activo
1119	C.T.E. M N° 30234			Pampa	Activo
1120	Ramos Cordova	Iraida		Pampa	Exonerado
1121	Cordova Davirán	Macabeo E.	19958836	Pampa	Activo
1122	Salinas Davila	Maximo	19960709	Pampa	Activo
1123	Zamata Rosales	Feliciana	40732552	Pampa	Activo
1124	Avila Cordova	Pablo	19960477	Pampa	Activo
1125	Zamudio Meza	Paulina	19960723	Pampa	Activo
1126	Zarate Malpica	Eugenia	19959303	Pampa	Activo
1127	Zarate Malpica	Paulino	19959302	Pampa	Activo

1128	Zarate Malpica	Edy Nicasia	19965597	Pampa	Activo
1129	Zarate Malpica	Edy Nicasia	19965597	Pampa	Activo
1130	Chipana Colonio	Valeriano	19960336	Pampa	Activo
1131	Davíran Gelindo	Carmen	20415474	Pampa	Activo
1132	Cordova de Sánchez	Epifanía		Pampa	Activo
1133	Rodríguez Sedaño	José	19959637	Pampa	Activo
1134	Sánchez Huamán	Pedro	19961168	Pampa	Activo
1135	Sánchez Sebastian	Alfredo Gil	41425380	Pampa	Activo
1136	Gaspar Astucuri	Teodora	19960275	Pampa	Exonerado
1137	Ceras Meléndez	Nicanor	19959240	Pampa	Exonerado
1138	Galindo Sánchez	Pedro	19959098	Pampa	Exonerado
1139	Quispe Meza Irene	Sofía	40728419	Pampa	Exonerado
1140	Tiza Dávila Margarita	Luz	19961095	Pampa	Activo
1141	Sánchez Zamudio	Natalia	19960265	Pampa	Exonerado
1142	Malpica Meza	Indalecia	42141900	Pampa	Activo
1143	Zarate Malpica	Bartolomé	19958947	Pampa	Activo
1144	Palacios de Castro	Nery	19960188	Pampa	Activo
1145	Rojas Rodríguez	Elisa	16124173	Pampa	Activo
1146	Salinas Nauta	Antonio	19959011	Pampa	Activo
1147	Suazo de Pariona	Reynalda	19959555	Pampa	Activo
1148	Rodríguez Villanas	Ernesto	20404553	Pampa	Activo
1149	Rodríguez Sedaño	Gregorio	19959374	Pampa	Activo
1150	Najera Travezaño	Marcela Estela		Pampa	Activo
1151	Suazo Sánchez	Maranao	19959300	Pampa	Activo
1152	Suazo Sánchez	Elsa	19960266	Pampa	Activo
1153	Suarez Ramos Vd. de Ramos	Florencia	19960101	Pampa	Activo
1154	De La Cruz Helario	Máximo	23228869	Pampa	Activo
1155	Villegas Piuca	Dionicia	20985419	Pampa	Activo
1156	Córdova Bernabé	Antonia		Pampa	Exonerado
1157	Miranda Zamudio	Alvino	19976519	Pampa	Exonerado
1158	Quispe Pariona	Irene		Pampa	Activo
1159	Astocuri Alanya	Victoria		Pampa	Activo
1160	Zamudio Meza	Jesús	19960326	Pampa	Activo
1161	Ciperiano Cantorín	Urbano	20417813	Pampa	Activo
1162	Chipana Colonio	Antonio		Pampa	Activo
1163	Miranda Chávez	Jesús Eugenio		Pampa	Activo
1164	Mantari Buendía	José L.		Pampa	Activo
1165	Chuquimantari Pariona	Nely	20011433	Pampa	Activo
1166	TorpJteo Contreras	Elizabeth	48461329	Pampa	Activo
1167	Sánchez Huamán	Vilma	19960292	Pampa	Activo
1168	Zarate Malpica	Eugenia	19959303	Pampa	Activo
1169	Quispe Vela de Soto	Leoncia	19959926	Pampa	Activo
1170	Cóndor Carbajal	Reyna	20078707	Pampa	Activo
1171	Miranda Salome	Herlinda	41244413	Pampa	Activo

1172	Ruiz Villegas	Francisco	20680165	Pampa	Activo
1173	Aquino Meza	Justina	19859056	Pampa	Activo
1174	Avila Ramos	Edita Maria	19960678	Pampa	Activo
1175	Zacarías León	Agustín	19960025	Pampa	Activo
1176	Davirán Salinas	Saturnino	19960938	Pampa	Activo
1177	Davirán Pacheco	Sabina	19960808	Pampa	Activo
1178	Yarasca Chipana	Teódulo	19958956	Pampa	Activo
1179	J.N.E. Nº 529			Pampa	Activo
1180	Miranda Zamudio	David		Pampa	Activo
1181	Gaspar Astucuri	Teodora	19960275	Pampa	Activo
1182	Avila Malpica	Eugenia	19959174	Pampa	Activo
1183	Cuba Chávez	Juana Juliana	19976707	Pampa	Exonerado
1184	Beltrán Ponce	Nelly Giovanna	20084523	Pampa	Activo
1185	Sánchez Huamán	Olimpída	19960209	Pampa	Activo
1186	Meza Chipana	Nemesio	19959214	Pampa	Activo
1187	Morcado Pastrana	Isabel	19960620	Pampa	Activo
1188	Suarez Hinojosa	Máxima	20983677	Pampa	Activo
1189	Zarate Carhuachín	Velia Victoria	19958988	Pampa	Exonerado
1190	Huayta Crispin	Valeriano	20435502	Pampa	Activo
1191	Pariona Coila	Liliana Ancieta		Pampa	Activo
1192	Gamarra Chipana	Gregoria	19959495	Pampa	Activo
1193	Soto de Pérez	Honorata	19845118	Pampa	Activo
1194	Roca Uantoy	Paulino		Pampa	Activo
1195	Huamán Meza	Benjamín F.	20425102	Pampa	Exonerado
1196	Pacheco Calderón	Yuliana	19977368	Pampa	Activo
1197	Palacios Dávila	María	19960057	Pampa	Activo
1198	Contreras Meza	Mencía Albina	19959573	Pampa	Activo
1199	Rivera Amanso	Bertila	19960221	Pampa	Activo
1200	Meza Moreno	Wilfredo	19958912	Pampa	Activo
1201	Quispe Barzola	Gertrudes		Pampa	Activo
1202	Ordoñez Contreras	Antonio	19959367	Pampa	Activo
1203	Contreras Batalla	Froilán	19959101	Pampa	Activo
1204	Miguel Mayta	Huberta		Pampa	Activo
1205	Quispe Condonay	María	46242489	Pampa	Activo
1206	Rivera Segura	Esteban	4016849	Pampa	Activo
1207	Benito Suazo	Rosario	70279272	Pampa	Activo
1208	Contreras Galindo	Gregoria	19961145	Pampa	Activo
1209	Oscanoa Vda. de Palacin	Paulina	21061414	Pampa	Exonerado
1210	Quispe Pariona	Armanda Rosa	19960360	Pampa	Activo
1211	Prudencio Pocomucha	Maruja	23259041	Pampa	Activo
1212	Mayorga Nájera	María	19949173	Pampa	Activo
1213	Bernabé Chuquimantari	Silvana	41661290	Pampa	Activo
1214	Contreras Barja	Cristel	19961114	Pampa	Activo
1215	Rodríguez Sedaño	Elias	19959585	Pampa	Activo

1216	Davirán Inocente	Luis	19959149	Pampa	Activo
1217	Avila Ramos	Zenaida		Pampa	Activo
1218	Quispe Allca	Jhon Antonio	42722507	Pampa	Activo
1219	Colonio Pacheco	Eddy Ines		Pampa	Exonerado
1220	Yarasca Chipana	Gudofreda	19960263	Pampa	Activo
1221	Marin Cuba Genoveva	Victoria	19959658	Pampa	Activo
1222	Sullca Cauchos	Felicita	23521916	Pampa	Activo
1223	Panduro Vargas	José		Pampa	Activo
1224	Inga Fabián de Ramos	Sandy Iris	40878016	Pampa	Activo
1225	Zamudio Pomalaza	Maura		Pampa	Activo
1226	Simón Maldonado	Lincol	19959294	Pampa	Activo
1227	Contreras Ramos	Celinda	19960576	Pampa	Activo
1228	Hinostroza Apolinario	Raúl	20973485	Pampa	Activo
1229	Pacheco Davirau	Sabina	19960249	Pampa	Activo
1230	Pariona de Bendezu	Ercira	20098735	Pampa	Activo
1231	Quispe Daviran	Juan		Pampa	Activo
1232	Pacheco Contreras	Justina	41080153	Pampa	Activo
1233	Quispe Machuca	Julia Lucila	19960870	Pampa	Activo
1234	Hurtado Gago	Paulina	19960464	Pampa	Activo
1235	Salinas Dávila	Narciso	19958804	Pampa	Activo
1236	Salinas Sánchez	Melecio	19960251	Pampa	Exonerado
1237	Nájera Paytampoma	Yonila F.	19960466	Pampa	Activo
1238	Chuquimantari Gonzales	Samuel	19960641	Pampa	Activo
1239	Chipana Sánchez	Margarita Maria	19960072	Pampa	Activo
1240	Ponce Pariona	Lincol	19960097	Pampa	Activo
1241	Ayala de Zamudio	Justina		Pampa	Activo
1242	Salomé Huayta	Federico		Pampa	Exonerado
1243	Rivera Inga	Judith	41018794	Pampa	Activo
1244	Sotacuro Pacheco	Jesús	71481508	Pampa	Activo
1245	Chipana Ro#an de Lozano	Victoria		Pampa	Exonerado
1246	Arroyo de Martínez	Nancy	20895039	Pampa	Activo
1247	Salomé Pacheco	Sabino	19960615	Pampa	Activo
1248	Nuñez Contreras	Arturo Cayo	19961057	Pampa	Activo
1249	Gaspar Barja	Herlinda	19977406	Pampa	Activo
1250	Miranda Quispe	Dario	19960042	Pampa	Activo
1251	Pacheco Contreras	Felicia	20098739	Pampa	Activo
1252	Paquiegauri Noa	Estela	23466227	Pampa	Activo
1253	Rodríguez Suazo	Nicanor	19958850	Pampa	Activo
1254	Curo Ricra	Ercelia	41684299	Pampa	Activo
1255	Curo de Vílchez	Juliana	37713684	Pampa	Activo
1256	Tiza Dávila	Margarita	19961095	Pampa	Activo
1257	Avila Avila Manuel	Virgilio	19959798	Pampa	Activo
1258	Alanya Mercado	Irma R.	21011825	Pampa	Activo
1259	Zarate Malpica	Paulino	19959302	Pampa	Activo

1260	Chipartá Veliz	Ronaldo Moisés		Pampa	Activo
1261	Contreras Urbano	Rodolfo		Pampa	Exonerado
1262	Salomé Zamudio	Andrés Domingo	19977550	Pampa	Activo
1263	De la Cruz Romani	Lucia	23468108	Pampa	Activo
1264	Buendía Salomé	Ricardo	40122699	Pampa	Activo
1265	Chuquimantari Avila	William H.	19960889	Pampa	Activo
1266	Velásquez Salinas	Jacinto	20098609	Pampa	Activo
1267	Batalla Corzo	Marta del Carmen		Pampa	Activo
1268	Castillo Tafur	Rosa Eduviges		Pampa	Activo
1269	Zamudio Miranda	Máxima	19959044	Pampa	Activo
1270	Gaspar Alcoser	Florentino	20993689	Pampa	Activo
1271	Vilcas Quispe	Gregoria	40520869	Pampa	Activo
1272	Huamani Carrillo	Heráclides	23267231	Pampa	Activo
1273	Roque Javier David Eleaud	David Eleaud	16294636	Pampa	Activo
1274	Yalli Quispe	Angélica Alicia	23270607	Pampa	Activo
1275	Salinas Dávila	Ernán	19958906	Pampa	Activo
1276	Rivas Ñaupari	Rodrigo Daniel	19837028	Pampa	Activo
1277	Cárdenas Gamarra	Jhym Jhover	40769055	Pampa	Activo
1278	Crispin Zacarías	Elizabeth	45009672	Pampa	Activo
1279	Rodríguez Machuca	Lucia	23549738	Pampa	Activo
1280	Ponce de Sanches	Marta	19960258	Pampa	Activo
1281	Cipriano Chavarria	Juan	20414168	Pampa	Activo
1282	Huamani Capcha	Yonny		Pampa	Activo
1283	Almonacid Soto	Rosario Manuela	20663007	Pampa	Activo
1284	Ordoñez Contreras	Liberata	19960779	Pampa	Activo
1285	Huamán Quispe	María Isabel	45700152	Pampa	Activo
1286	Contreras Allca	Gaudencia Ilda	19960736	Pampa	Activo
1287	Cuba Rivas	Lily Elva		Pampa	Activo
1288	Córdova Garayar	Maria Esther		Pampa	Activo
1289	Hinostroza Rios	Ivan Victor	23276444	Pampa	Activo
1290	Cielito Quilqueño (MUNICIP)			Pampa	Activo
1291	Davila Veliz	Aida Luz	19960815	Pampa	Activo
1292	Sociedad Carnav Cruz de Espinas			Pampa	Exonerado
1293	Sánchez Córdova	Francisca	19959152	Pampa	Activo
1294	Yarasca Galindo	Marina	20592441	Pampa	Activo
1295	Miranda Salome	Elva	44636269	Pampa	Activo
1296	Victorio Silvestre	Jina	44331451	Pampa	Activo
1297	Lazaro Landeon	Elizabeth	42307892	Pampa	Activo
1298	Cordova Alarcon	Nory	42744630	Pampa	Activo
1299	Melo Ureta	Heli Janai	43518594	Pampa	Activo
1300	Zarate Malpica	Juan Luis	19959664	Pampa	Activo

1301	Morales Sánchez	Elva Baselia	19961055	Pampa	Activo
1302	Curo Ricra	Víctor	40778121	Pampa	Activo
1303	Punce Carrasco	William		Pampa	Activo
1304	Rodríguez Ponce	Nélida	43863584	Pampa	Activo
1305	Camargo Pizarro	Laura Juana	42279384	Pampa	Activo
1306	Rodriguez Loroña	Teresa	20078809	Pampa	Exonerado
1307	Zarate Ordóñez	Mariluz	41771265	Pampa	Activo
1308	Gamarra Pariona	Freddy	19960769	Pampa	Activo
1309	Fernández Ccente	Gloria R.	80431980	Pampa	Activo
1310	Feliciana Zamata	Rosales	40732550	Pampa	Activo
1311	Bruno Palacios	Edy Mavel	20098645	Pampa	Activo
1312	Alcocer Contreras	Catalina Juana	19961012	Pampa	Activo
1313	Alcocer Cuba	Ana	45672490	Pampa	Activo
1314	Huayta Chavez	Filverto Eugenio	19959740	Pampa	Activo
1315	Bruno Palacio	Yenny Lucy	20098614	Pampa	Activo
1316	Soto Chipana	Juana Segundina	19959645	Pampa	Activo
1317	Rivera Chipana	Gladys		Pampa	Exonerado
1318	Zarate Monago	Héctor Alfredo	20098702	Pampa	Activo
1319	Aguilar Quispe	Cesar R.	19960924	Pampa	Activo
1320	Inga Espinosa	Rosmeri Luz	42404510	Pampa	Activo
1321	Samaniego Zacarías	Juvencia	80007018	Pampa	Activo
1322	Salinas Pariona	María	45039740	Pampa	Activo
1323	Contreras Batalla	Froilan	19959101	Pampa	Activo
1324	Huincho Choccelahua	Elmer Efrain	71645450	Pampa	Activo
1325	Quispe Contreras	Alberto	70420331	Pampa	Activo
1326	Meza Maldonado	Octavio	19960588	Pampa	Activo
1327	Suazo Chávez	Eustaquia	19959086	Pampa	Activo
1328	Quiza Ramos	Mery	40684317	Pampa	Activo
1329	E.P. Evangélica			Pampa	Activo
1330	Zamudio Ventura	Percy Alex	42780388	Pampa	Activo
1331	Zavala Sanchez	Luis Ivan		Pampa	Exonerado
1332	Contreras Aguilar	Jhaneth Rosa	43853158	Pampa	Activo
1333	Gamarra Chipana	Doris E.	19959837	Pampa	Activo
1334	Inga Damian	Helmer	42106597	Pampa	Exonerado
1335	Oré Lázaro	Rosario María	44821816	Pampa	Activo
1336	Hurtado Gago	Jovina	19960556	Pampa	Activo
1337	Morales Laura	Cesar Alberto	40830641	Pampa	Activo
1338	Contreras Avila	Angélica	40358732	Pampa	Activo
1339	Zamudio Quispe	Justina	19986680	Pampa	Activo
1340	Yarasca Tiza	Luz	45908012	Pampa	Activo
1341	Cabello Sanchas	Sulema	46572100	Pampa	Activo
1342	Venegas Zapata	Maria Irene		Pampa	Activo
1343	Rodríguez Quispe	Bibina		Pampa	Activo

1344	Meza Moreno	Wilfredo Santiago	19958912	Pampa	Activo
1345	Quispe Pariona	Fredy Pablo	19958988	Pampa	Activo
1346	Huaranga Asto	Olga Lidia	20103450	Pampa	Activo
1347	Huayra Quispe	Florinda	45392656	Pampa	Activo
1348	Pariona de Rodríguez	Alejandra		Pampa	Activo
1349	Surichaqui Gomez	Felicita	20107398	Pampa	Activo
1350	Yarasca Galindo	Carlos A.	43510273	Pampa	Activo
1351	Zarate Malpica	Juan Luis	19959664	Pampa	Activo
1352	Zarate Malpica	Paulino	19959302	Pampa	Activo
1353	Yarasca Galindo	Héctor	20098617	Pampa	Activo
1354	Daviran Inocente	Juan de Dios		Pampa	Exonerado
1355	Davila Astocuri	Mary Luz	40122441	Pampa	Activo
1356	Velasquez Salinas	Jacinto	20098609	Pampa	Activo
1357	Hurtado Ore	Rosa Gudelia	19961090	Pampa	Activo
1358	Borja Gaspar	Marleni	40728404	Pampa	Activo
1359	Martinez Porras	Jesús		Pampa	Activo
1360	Bruno Palacios	Annie A.	47250561	Pampa	Activo
1361	Sanchez Gaspar	Almagro	43190252	Pampa	Activo
1362	Martinez Porras	Jesús		Pampa	Activo

Anexo 5 Asociados del sector Pampa - Quilcas

```
public class DUsuarios
   {
      private int _Id;
      private string _Dni;
      private string _Apellidos;
      private string _Nombre;
      private string _Usuario;
      private string _Contraseña;
      private string _Acceso;
      public int Id
      {
         get { return _Id; }
         set { _Id = value; }
      }
      public string Dni
      {
         get { return _Dni; }
         set { _Dni = value; }
      }
      public string Apellidos
      {
         get { return _Apellidos; }
         set { _Apellidos = value; }
      }
      public string Nombre
      {
         get { return _Nombre; }
         set { _Nombre = value; }
      }
      public string Usuario
      {
         get { return _Usuario; }
         set { _Usuario = value; }
      }
      public string Contraseña
      {
         get { return _Contraseña; }
         set { _Contraseña = value; }
      }
      public string Acceso
      {
         get { return _Acceso; }
         set { _Acceso = value; }
      }

      public DUsuarios() { }

      public DUsuarios (int id, string dni, string apellidos, string nombre, string usuario,string
contraseña,string acceso)

      {
         this.Id = id;
         this.Dni = dni;
         this.Apellidos = apellidos;
         this.Nombre = nombre;
         this.Usuario = usuario;
         this.Contraseña = contraseña;
         this.Acceso = acceso;
      }
      public string Insertar(DUsuarios Usuario)
      {
         string rpta = "";
         SqlConnection SqlCon = new SqlConnection();
         try
         {
            SqlCon.ConnectionString = Conexion.Cn;
```

```
            SqlCon.Open();
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "insertar_usuario";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParId = new SqlParameter();
            ParId.ParameterName = "@id";
            ParId.SqlDbType = SqlDbType.Int;
            ParId.Direction = ParameterDirection.Output;
            SqlCmd.Parameters.Add(ParId);

            SqlParameter ParDni = new SqlParameter();
            ParDni.ParameterName = "@dni";
            ParDni.SqlDbType = SqlDbType.VarChar;
            ParDni.Size = 8;
            ParDni.Value = Usuario.Dni;
            SqlCmd.Parameters.Add(ParDni);

            SqlParameter ParApellidos = new SqlParameter();
            ParApellidos.ParameterName = "@apellidos";
            ParApellidos.SqlDbType = SqlDbType.VarChar;
            ParApellidos.Size = 50;
            ParApellidos.Value = Usuario.Apellidos;
            SqlCmd.Parameters.Add(ParApellidos);
            SqlParameter ParNombre = new SqlParameter();
            ParNombre.ParameterName = "@nombre";
            ParNombre.SqlDbType = SqlDbType.VarChar;
            ParNombre.Size = 50;
            ParNombre.Value = Usuario.Nombre;
            SqlCmd.Parameters.Add(ParNombre);

            SqlParameter ParUsuario = new SqlParameter();
            ParUsuario.ParameterName = "@usuario";
            ParUsuario.SqlDbType = SqlDbType.VarChar;
            ParUsuario.Size = 20;
            ParUsuario.Value = Usuario.Usuario;
            SqlCmd.Parameters.Add(ParUsuario);

            SqlParameter ParContraseña = new SqlParameter();
            ParContraseña.ParameterName = "@contraseña";
            ParContraseña.SqlDbType = SqlDbType.VarChar;
            ParContraseña.Size = 20;
            ParContraseña.Value = Usuario.Contraseña;
            SqlCmd.Parameters.Add(ParContraseña);

            SqlParameter ParAcceso = new SqlParameter();
            ParAcceso.ParameterName = "@acceso";
            ParAcceso.SqlDbType = SqlDbType.VarChar;
            ParAcceso.Size = 30;
            ParAcceso.Value = Usuario.Acceso;
            SqlCmd.Parameters.Add(ParAcceso);

            rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se Ingreso el Registro";
        }
        catch (Exception ex)
        {
            rpta = ex.Message;
        }
        finally
        {
            if (SqlCon.State == ConnectionState.Open) SqlCon.Close();
        }
        return rpta;
    }
    public string Editar(DUsuarios Usuario)
```

```
{
    string rpta = "";
    SqlConnection SqlCon = new SqlConnection();
    try
    {
        SqlCon.ConnectionString = Conexion.Cn;
        SqlCon.Open();
        SqlCommand SqlCmd = new SqlCommand();
        SqlCmd.Connection = SqlCon;
        SqlCmd.CommandText = "editar_usuario";
        SqlCmd.CommandType =        CommandType.StoredProcedure;

        SqlParameter ParId = new SqlParameter();
        ParId.ParameterName = "@id";
        ParId.SqlDbType = SqlDbType.Int;
        ParId.Value = Usuario.Id;
        SqlCmd.Parameters.Add(ParId);

        SqlParameter ParDni = new SqlParameter();
        ParDni.ParameterName = "@dni";
        ParDni.SqlDbType = SqlDbType.VarChar;
        ParDni.Size = 8;
        ParDni.Value = Usuario.Dni;
        SqlCmd.Parameters.Add(ParDni);

        SqlParameter ParApellidos = new SqlParameter();
        ParApellidos.ParameterName = "@apellidos";
        ParApellidos.SqlDbType = SqlDbType.VarChar;
        ParApellidos.Size = 50;
        ParApellidos.Value = Usuario.Apellidos;
        SqlCmd.Parameters.Add(ParApellidos);

        SqlParameter ParNombre = new SqlParameter();
        ParNombre.ParameterName = "@nombre";
        ParNombre.SqlDbType = SqlDbType.VarChar;
        ParNombre.Size = 50;
        ParNombre.Value = Usuario.Nombre;
        SqlCmd.Parameters.Add(ParNombre);

        SqlParameter ParUsuario = new SqlParameter();
        ParUsuario.ParameterName = "@usuario";
        ParUsuario.SqlDbType = SqlDbType.VarChar;
        ParUsuario.Size = 20;
        ParUsuario.Value = Usuario.Usuario;
        SqlCmd.Parameters.Add(ParUsuario);

        SqlParameter ParContraseña = new SqlParameter();
        ParContraseña.ParameterName = "@contraseña";
        ParContraseña.SqlDbType = SqlDbType.VarChar;
        ParContraseña.Size = 20;
        ParContraseña.Value = Usuario.Contraseña;
        SqlCmd.Parameters.Add(ParContraseña);

        SqlParameter ParAcceso = new SqlParameter();
        ParAcceso.ParameterName = "@acceso";
        ParAcceso.SqlDbType = SqlDbType.VarChar;
        ParAcceso.Size = 30;
        ParAcceso.Value = Usuario.Acceso;
        SqlCmd.Parameters.Add(ParAcceso);

        rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se Actualizó el Registro";
    }
    catch (Exception ex)
    {
        rpta = ex.Message;
    }
```

```
        finally
        {
            if (SqlCon.State == ConnectionState.Open) SqlCon.Close();
        }
        return rpta;
    }
    public string Eliminar(DUsuarios Usuario)
    {
        string rpta = "";
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
            SqlCon.Open();

            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "eliminar_usuario";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParId = new SqlParameter();
            ParId.ParameterName = "@id";
            ParId.SqlDbType = SqlDbType.Int;
            ParId.Value = Usuario.Id;
            SqlCmd.Parameters.Add(ParId);

            rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se se Eliminó el Registro";
        }
        catch (Exception ex)
        {
            rpta = ex.Message;
        }
        finally
        {
            if (SqlCon.State == ConnectionState.Open) SqlCon.Close();
        }
        return rpta;
    }
    public DataTable Mostrar()
    {
        DataTable DtResultado = new DataTable("usuarios");
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "mostrar_usuarios";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);
        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;
    }
    public DataTable Login(DUsuarios Usuario)
    {
        DataTable DtResultado = new DataTable("usuarios");
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
```

```
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "login_usuario";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParUsuario = new SqlParameter();
            ParUsuario.ParameterName = "@usuario";
            ParUsuario.SqlDbType = SqlDbType.VarChar;
            ParUsuario.Size = 20;
            ParUsuario.Value = Usuario.Usuario;
            SqlCmd.Parameters.Add(ParUsuario);

            SqlParameter ParContraseña= new SqlParameter();
            ParContraseña.ParameterName = "@contraseña";
            ParContraseña.SqlDbType = SqlDbType.VarChar;
            ParContraseña.Size = 20;
            ParContraseña.Value = Usuario.Contraseña;
            SqlCmd.Parameters.Add(ParContraseña);

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);
        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;
    }

  }
}
```

Anexo 6 Capa datos – Usuario

```
using CapaDatos;
namespace CapaNegocio
{
    public class NUsuarios
    {
        public static string Insertar(string nombre, string apellidos,
string dni,string usuario,
            string contraseña, string acceso)
        {
            DUsuarios Obj = new DUsuarios();
            Obj.Dni = dni;
            Obj.Apellidos = apellidos;
            Obj.Nombre = nombre;
            Obj.Usuario = usuario;
            Obj.Contraseña = contraseña;
            Obj.Acceso = acceso;
            return Obj.Insertar(Obj);
        }
        public static string Editar(int id, string nombre, string apellidos,
string dni, string usuario,
            string contraseña, string acceso)
        {
            DUsuarios Obj = new DUsuarios();
            Obj.Id = id;
            Obj.Dni = dni;
            Obj.Apellidos = apellidos;
            Obj.Nombre = nombre;
            Obj.Usuario = usuario;
            Obj.Contraseña = contraseña;
```

```
            Obj.Acceso = acceso;
            return Obj.Editar(Obj);
        }
        public static string Eliminar(int id)
        {
            DUsuarios Obj = new DUsuarios();
            Obj.Id = id;
            return Obj.Eliminar(Obj);
        }
        public static DataTable Mostrar()
        {
            return new DUsuarios().Mostrar();
        }
        public static DataTable Login(string usuario, string contraseña)
        {
            DUsuarios Obj = new DUsuarios();
            Obj.Usuario= usuario;
            Obj.Contraseña = contraseña;
            return Obj.Login(Obj);
        }
    }
}
```

Anexo 7 Capa negocio – Usuario

```
using CapaNegocio;

namespace CapaPresentacion
{
    public partial class frmUsuarios : Form
    {
        private bool IsNuevo = false;
        private bool IsEditar = false;
        public frmUsuarios()
        {
            InitializeComponent();
            this.ttMensaje.SetToolTip(this.txtNombre, "Ingrese el
nombre");
            this.ttMensaje.SetToolTip(this.txtApellidos, "Ingrese los
apellidos");
            this.ttMensaje.SetToolTip(this.txtDni, "Ingrese el
documento");
            this.ttMensaje.SetToolTip(this.txtUsuario, "Ingrese un nombre
de usuario");
            this.ttMensaje.SetToolTip(this.txtContraseña, "Ingrese la
contraseña del usuario");
        }
        private void MensajeOK(string Mensaje)
        {
            MessageBox.Show(Mensaje, "Sistema Control de Asistencias",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void MensajeError(string Mensaje)
        {
            MessageBox.Show(Mensaje, "Sistema Control de Asistencias",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void Limpiar()
        {
            this.txtIdUsuario.Text = string.Empty;
            this.txtNombre.Text = string.Empty;
            this.txtApellidos.Text = string.Empty;
```

```
        this.txtDni.Text = string.Empty;

        this.txtContraseña.Text = string.Empty;
    }
    private void Habilitar(bool Valor)
    {
        this.txtIdUsuario.ReadOnly = !Valor;
        this.txtNombre.ReadOnly = !Valor;
        this.txtApellidos.ReadOnly = !Valor;
        this.txtDni.ReadOnly = !Valor;
        this.cbAcceso.Enabled = Valor;

        this.txtContraseña.ReadOnly = !Valor;
    }
    private void Botones()
    {
        if (this.IsNuevo || this.IsEditar)
        {
            this.Habilitar(true);
            this.btnNuevo.Enabled = false;
            this.btnInsertar.Enabled = true;
            this.btnEditar.Enabled = false;
            this.btnEliminar.Enabled = false;
        }
        else
        {
            this.Habilitar(false);
            this.btnNuevo.Enabled = true;
            this.btnInsertar.Enabled = false;
            this.btnEditar.Enabled = true;
            this.btnEliminar.Enabled = true;
        }

    }
    private void OcultarColumnas()
    {
        this.dataListado.Columns[0].Visible = false;
        this.dataListado.Columns[1].Visible = false;
        this.dataListado.Columns[6].Visible = false;
    }
    private void Mostrar()
    {
        this.dataListado.DataSource = NUsuarios.Mostrar();
        this.OcultarColumnas();
    }

    private void frmUsuarios_Load(object sender, EventArgs e)
    {
        this.Mostrar();
        this.Habilitar(false);
        this.Botones();
    }

    private void btnNuevo_Click(object sender, EventArgs e)
    {
        this.IsNuevo = true;
        this.IsEditar = false;
        this.Botones();
        this.Limpiar();
        this.Habilitar(true);
        this.txtNombre.Focus();
    }
    private void btnInsertar_Click(object sender, EventArgs e)
    {
        try
```

```
            {
                string Rpta = "";
                if (this.txtNombre.Text == string.Empty ||
this.txtApellidos.Text == string.Empty || txtDni.Text == string.Empty ||
this.txtDni.TextLength < 8 ||txtUsuario.Text == string.Empty ||
txtContraseña.Text == string.Empty)
                {
                    MensajeError("Falta ingresar algunos datos, serán
remarcados");
                    errorIcono.SetError(txtNombre, "Ingrese un Valor");
                    errorIcono.SetError(txtApellidos, "Ingrese un Valor");
                    errorIcono.SetError(txtDni, "Ingrese un Valor");
                    errorIcono.SetError(txtUsuario, "Ingrese un Valor");
                    errorIcono.SetError(txtContraseña, "Ingrese un
Valor");
                }
                else
                {
                    if (this.IsNuevo)
                    {
                        Rpta =
NUsuarios.Insertar(this.txtNombre.Text.Trim(),

this.txtApellidos.Text.Trim(),this.txtUsuario.Text.Trim(),
                        txtDni.Text,txtContraseña.Text,cbAcceso.Text);

                    }
                    else
                    {

                        Rpta =
NUsuarios.Editar(Convert.ToInt32(this.txtIdUsuario.Text),
this.txtNombre.Text.Trim(),
                        this.txtApellidos.Text.Trim(),
                        txtDni.Text, this.txtUsuario.Text.Trim(),
txtContraseña.Text, cbAcceso.Text);
                    }

                    if (Rpta.Equals("OK"))
                    {
                        if (this.IsNuevo)
                        {
                            this.MensajeOK("Se creo correctamente el
usuario");
                        }
                        else
                        {
                            this.MensajeOK("Se actualizó correctamente el
usuario");
                        }

                    }
                    else
                    {

                        this.MensajeError(Rpta);
                    }
                    this.IsNuevo = false;
                    this.IsEditar = false;
                    this.Botones();
                    this.Limpiar();
                    this.Mostrar();
                }
            }
```

```
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message + ex.StackTrace);
            }
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.IsNuevo = false;
            this.IsEditar = false;
            this.Botones();
            this.Limpiar();
            this.Habilitar(false);
        }

        private void btnEliminar_Click(object sender, EventArgs e)
        {
            try
            {
                DialogResult Opcion;
                Opcion = MessageBox.Show("¿Desea eliminar el usuario
marcado?", "Sistema Control de Asistencias", MessageBoxButtons.OKCancel,
MessageBoxIcon.Question);

                if (Opcion == DialogResult.OK)
                {
                    string Codigo;
                    string Rpta = "";

                    foreach (DataGridViewRow row in dataListado.Rows)
                    {
                        if (Convert.ToBoolean(row.Cells[0].Value))
                        {
                            Codigo = Convert.ToString(row.Cells[1].Value);
                            Rpta =
NUsuarios.Eliminar(Convert.ToInt32(Codigo));

                            if (Rpta.Equals("OK"))
                            {
                                this.MensajeOK("Se Eliminó Correctamente
el registro");
                            }
                            else
                            {
                                this.MensajeError(Rpta);
                            }

                        }
                    }
                    this.Mostrar();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message + ex.StackTrace);
            }
        }

        private void btnEditar_Click(object sender, EventArgs e)
        {
            if (!this.txtIdUsuario.Text.Equals(""))
            {
                this.IsEditar = true;
                this.Botones();
```

```
                this.Habilitar(true);
            }
            else
            {
                this.MensajeError("Debe de seleccionar primero el usuario
a modificar");
            }
        }

        private void dataListado_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.ColumnIndex == dataListado.Columns["Eliminar"].Index)
            {
                DataGridViewCheckBoxCell ChkEliminar =
(DataGridViewCheckBoxCell)dataListado.Rows[e.RowIndex].Cells["Eliminar"];
                ChkEliminar.Value = !Convert.ToBoolean(ChkEliminar.Value);
            }
        }
        private void chkEliminar_CheckedChanged(object sender, EventArgs
e)
        {
            if (chkEliminar.Checked)
            {
                this.dataListado.Columns[0].Visible = true;
            }
            else
            {
                this.dataListado.Columns[0].Visible = false;
            }
        }

        private void dataListado_DoubleClick(object sender, EventArgs
e)
        {
            this.txtIdUsuario.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["id"].Value);
            this.txtNombre.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["nombre"].Value);
            this.txtApellidos.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["apellidos"].Value)
;
            this.txtDni.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["dni"].Value);
            this.txtUsuario.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["usuario"].Value);
            this.txtContraseña.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["contraseña"].Value
);
            this.cbAcceso.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["acceso"].Value);
        }

        private void txtDni_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }

            if (txtDni.TextLength > 7 && e.KeyChar != '\b')
            {
```

```
                e.Handled = true;
            }
        }

        private void txtNombre_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }
        }

        private void txtApellidos_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
           }
   }


    }
}
```

Anexo 8 Capa presentación – Usuario

```
namespace CapaDatos
{
    public class DAsociados
    {
        private int _Id_asoc;
        private string _Codigo;
        private string _Dni;
        private string _Apellidos;
        private string _Nombre;
        private string _Direccion;
        private string _Sector;
        private string _Estado;
        private string _TextoBuscar;

        public int Id_asoc
        {
            get { return _Id_asoc; }
            set { _Id_asoc = value; }
        }
        public string Codigo
        {
            get { return _Codigo; }
            set { _Codigo = value; }
        }
        public string Dni
```

```
        {
            get { return _Dni; }
            set { _Dni = value; }
        }
        public string Apellidos
        {
            get { return _Apellidos; }
            set { _Apellidos = value; }
        }
        public string Nombre
        {
            get { return _Nombre; }
            set { _Nombre = value; }
        }
        public string Direccion
        {
            get { return _Direccion; }
            set { _Direccion = value; }
        }
        public string Sector
        {
            get { return _Sector; }
            set { _Sector = value; }
        }
        public string Estado
        {
            get { return _Estado; }
            set { _Estado = value; }
        }

        public DAsociados() { }

        public string TextoBuscar
        {
            get { return _TextoBuscar; }
            set { _TextoBuscar = value; }
        }

        public DAsociados(int id_asoc,string codigo, string dni,
string apellidos, string nombre,
            string direccion, string sector, string estado, string
textobuscar)
        {
            this.Id_asoc = id_asoc;
            this.Codigo = codigo;
            this.Dni = dni;
            this.Apellidos = apellidos;
            this.Nombre = nombre;
            this.Direccion = direccion;
            this.Sector = sector;
            this.Estado = estado;
            this.TextoBuscar = textobuscar;
        }

        public string Insertar(DAsociados Asociados)
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {
```

```
SqlCon.ConnectionString = Conexion.Cn;
SqlCon.Open();

SqlCommand SqlCmd = new SqlCommand();
SqlCmd.Connection = SqlCon;
SqlCmd.CommandText = "insertar_asociado";
SqlCmd.CommandType = CommandType.StoredProcedure;

SqlParameter ParId_asoc = new SqlParameter();
ParId_asoc.ParameterName = "@id_asoc";
ParId_asoc.SqlDbType = SqlDbType.Int;
ParId_asoc.Direction = ParameterDirection.Output;
SqlCmd.Parameters.Add(ParId_asoc);

SqlParameter ParCodigo = new SqlParameter();
ParCodigo.ParameterName = "@codigo";
ParCodigo.SqlDbType = SqlDbType.VarChar;
ParCodigo.Size = 4;
ParCodigo.Value = Asociados.Codigo;
SqlCmd.Parameters.Add(ParCodigo);

SqlParameter ParDni= new SqlParameter();
ParDni.ParameterName = "@dni";
ParDni.SqlDbType = SqlDbType.VarChar;
ParDni.Size = 8;
ParDni.Value = Asociados.Dni;
SqlCmd.Parameters.Add(ParDni);

SqlParameter ParApellidos= new SqlParameter();
ParApellidos.ParameterName = "@apellidos";
ParApellidos.SqlDbType = SqlDbType.VarChar;
ParApellidos.Size = 50;
ParApellidos.Value = Asociados.Apellidos;
SqlCmd.Parameters.Add(ParApellidos);

SqlParameter ParNombre = new SqlParameter();
ParNombre.ParameterName = "@nombre";
ParNombre.SqlDbType = SqlDbType.VarChar;
ParNombre.Size = 50;
ParNombre.Value = Asociados.Nombre;
SqlCmd.Parameters.Add(ParNombre);

SqlParameter ParDireccion = new SqlParameter();
ParDireccion.ParameterName = "@direccion";
ParDireccion.SqlDbType = SqlDbType.VarChar;
ParDireccion.Size = 50;
ParDireccion.Value = Asociados.Direccion;
SqlCmd.Parameters.Add(ParDireccion);

SqlParameter ParSector = new SqlParameter();
ParSector.ParameterName = "@sector";
ParSector.SqlDbType = SqlDbType.VarChar;
ParSector.Size = 50;
ParSector.Value = Asociados.Sector;
SqlCmd.Parameters.Add(ParSector);

SqlParameter ParEstado = new SqlParameter();
ParEstado.ParameterName = "@estado";
ParEstado.SqlDbType = SqlDbType.VarChar;
ParEstado.Size = 30;
ParEstado.Value = Asociados.Estado;
```

```
                SqlCmd.Parameters.Add(ParEstado);

                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
Ingreso el Registro";
            }
            catch (Exception ex)
            {
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }

        public string Editar(DAsociados Asociados)
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {

                SqlCon.ConnectionString = Conexion.Cn;
                SqlCon.Open();

                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "editar_asociado";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParId_asoc = new SqlParameter();
                ParId_asoc.ParameterName = "@id_asoc";
                ParId_asoc.SqlDbType = SqlDbType.Int;
                ParId_asoc.Value = Asociados.Id_asoc;
                SqlCmd.Parameters.Add(ParId_asoc);

                SqlParameter ParCodigo = new SqlParameter();
                ParCodigo.ParameterName = "@codigo";
                ParCodigo.SqlDbType = SqlDbType.VarChar;
                ParCodigo.Size = 4;
                ParCodigo.Value = Asociados.Codigo;
                SqlCmd.Parameters.Add(ParCodigo);

                SqlParameter ParDni = new SqlParameter();
                ParDni.ParameterName = "@dni";
                ParDni.SqlDbType = SqlDbType.VarChar;
                ParDni.Size = 8;
                ParDni.Value = Asociados.Dni;
                SqlCmd.Parameters.Add(ParDni);

                SqlParameter ParApellidos = new SqlParameter();
                ParApellidos.ParameterName = "@apellidos";
                ParApellidos.SqlDbType = SqlDbType.VarChar;
                ParApellidos.Size = 50;
                ParApellidos.Value = Asociados.Apellidos;
                SqlCmd.Parameters.Add(ParApellidos);

                SqlParameter ParNombre = new SqlParameter();
                ParNombre.ParameterName = "@nombre";
```

```
                ParNombre.SqlDbType = SqlDbType.VarChar;
                ParNombre.Size = 50;
                ParNombre.Value = Asociados.Nombre;
                SqlCmd.Parameters.Add(ParNombre);

                SqlParameter ParDireccion = new SqlParameter();
                ParDireccion.ParameterName = "@direccion";
                ParDireccion.SqlDbType = SqlDbType.VarChar;
                ParDireccion.Size = 50;
                ParDireccion.Value = Asociados.Direccion;
                SqlCmd.Parameters.Add(ParDireccion);

                SqlParameter ParSector = new SqlParameter();
                ParSector.ParameterName = "@sector";
                ParSector.SqlDbType = SqlDbType.VarChar;
                ParSector.Size = 50;
                ParSector.Value = Asociados.Sector;
                SqlCmd.Parameters.Add(ParSector);

                SqlParameter ParEstado = new SqlParameter();
                ParEstado.ParameterName = "@estado";
                ParEstado.SqlDbType = SqlDbType.VarChar;
                ParEstado.Size = 30;
                ParEstado.Value = Asociados.Estado;
                SqlCmd.Parameters.Add(ParEstado);

                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
Actualizó el Registro";
            }
            catch (Exception ex)
            {
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }

        public string Eliminar(DAsociados Asociados)
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {

                SqlCon.ConnectionString = Conexion.Cn;
                SqlCon.Open();

                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "eliminar_asociado";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParId_asoc = new SqlParameter();
                ParId_asoc.ParameterName = "@id_asoc";
                ParId_asoc.SqlDbType = SqlDbType.Int;
                ParId_asoc.Value = Asociados.Id_asoc;
                SqlCmd.Parameters.Add(ParId_asoc);
```

```
                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
se Eliminó el Registro";
            }
            catch (Exception ex)
            {
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }

        public DataTable Mostrar()
        {
            DataTable DtResultado = new DataTable("asociados");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "mostrar_asociado";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                SqlDat.Fill(DtResultado);

            }
            catch (Exception)
            {
                DtResultado = null;
            }
            return DtResultado;

        }
        public DataTable BuscarApellidos(DAsociados Asociados)
        {
            DataTable DtResultado = new DataTable("asociados");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "buscar_asociado_apellidos";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParTextoBuscar = new SqlParameter();
                ParTextoBuscar.ParameterName = "@textobuscar";
                ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
                ParTextoBuscar.Size = 50;
                ParTextoBuscar.Value = Asociados.TextoBuscar;
                SqlCmd.Parameters.Add(ParTextoBuscar);

                SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                SqlDat.Fill(DtResultado);
```

```
        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;

    }
    public DataTable BuscarEstado(DAsociados Asociados)
    {
        DataTable DtResultado = new DataTable("asociados");
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "buscar_asociado_estado";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParTextoBuscar = new SqlParameter();
            ParTextoBuscar.ParameterName = "@textobuscar";
            ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
            ParTextoBuscar.Size = 50;
            ParTextoBuscar.Value = Asociados.TextoBuscar;
            SqlCmd.Parameters.Add(ParTextoBuscar);

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);

        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;

    }
    public DataTable BuscarCodigo(DAsociados Asociados)
    {
        DataTable DtResultado = new DataTable("asociados");
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "buscar_asociado_codigo";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParTextoBuscar = new SqlParameter();
            ParTextoBuscar.ParameterName = "@textobuscar";
            ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
            ParTextoBuscar.Size = 50;
            ParTextoBuscar.Value = Asociados.TextoBuscar;
            SqlCmd.Parameters.Add(ParTextoBuscar);

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);

        }
```

```
                catch (Exception)
                {
                    DtResultado = null;
                }
                return DtResultado;

            }
            public DataTable BuscarSector(DAsociados Asociados)
            {
                DataTable DtResultado = new DataTable("asociados");
                SqlConnection SqlCon = new SqlConnection();
                try
                {
                    SqlCon.ConnectionString = Conexion.Cn;
                    SqlCommand SqlCmd = new SqlCommand();
                    SqlCmd.Connection = SqlCon;
                    SqlCmd.CommandText = "buscar_asociado_sector";
                    SqlCmd.CommandType = CommandType.StoredProcedure;

                    SqlParameter ParTextoBuscar = new SqlParameter();
                    ParTextoBuscar.ParameterName = "@textobuscar";
                    ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
                    ParTextoBuscar.Size = 50;
                    ParTextoBuscar.Value = Asociados.TextoBuscar;
                    SqlCmd.Parameters.Add(ParTextoBuscar);

                    SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                    SqlDat.Fill(DtResultado);

                }
                catch (Exception)
                {
                    DtResultado = null;
                }
                return DtResultado;

            }
        }
    }
```

Anexo 9 Capa datos - Asociado

```
using CapaDatos;
namespace CapaNegocio
{
    public class NAsociados
    {
        public static string Insertar(string codigo, string dni, string
apellidos, string nombre, string direccion,
            string sector,string estado)
        {
            DAsociados Obj = new DAsociados();
            Obj.Codigo = codigo;
            Obj.Dni = dni;
            Obj.Apellidos = apellidos;
            Obj.Nombre = nombre;
            Obj.Direccion = direccion;
            Obj.Sector = sector;
            Obj.Estado = estado;

            return Obj.Insertar(Obj);
        }

        public static string Editar(int id_asoc, string codigo, string
dni, string apellidos, string nombre, string direccion,
            string sector,string estado)
        {
            DAsociados Obj = new DAsociados();
            Obj.Id_asoc = id_asoc;
            Obj.Codigo = codigo;
            Obj.Dni = dni;
            Obj.Apellidos = apellidos;
            Obj.Nombre = nombre;
            Obj.Direccion = direccion;
            Obj.Sector = sector;
            Obj.Estado = estado;

            return Obj.Editar(Obj);
        }

        public static string Eliminar(int id_asoc)
        {
            DAsociados Obj = new DAsociados();
            Obj.Id_asoc = id_asoc;

            return Obj.Eliminar(Obj);
        }

        public static DataTable Mostrar()
        {
            return new DAsociados().Mostrar();
        }
        public static DataTable BuscarApellidos(string textobuscar)
        {
            DAsociados Obj = new DAsociados();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarApellidos(Obj);
        }
        public static DataTable BuscarEstado(string textobuscar)
        {
            DAsociados Obj = new DAsociados();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarEstado(Obj);
```

```
        }
        public static DataTable BuscarCodigo(string textobuscar)
        {
            DAsociados Obj = new DAsociados();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarCodigo(Obj);
        }
        public static DataTable BuscarSector(string textobuscar)
        {
            DAsociados Obj = new DAsociados();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarSector(Obj);
        }
    }
}
```

Anexo 10 Capa negocio – Asociado

```
using CapaNegocio;
namespace CapaPresentacion
{
    public partial class frmAsociados : Form
    {
        private bool IsNuevo = false;

        private bool IsEditar = false;
        public frmAsociados()
        {
            InitializeComponent();
            this.ttMensaje.SetToolTip(this.txtCodigo, "Ingrese el
código del asociado");
            this.ttMensaje.SetToolTip(this.txtDni, "Ingrese el DNI del
asociado");
            this.ttMensaje.SetToolTip(this.txtNombre, "Ingrese el
nombre del asociado");
            this.ttMensaje.SetToolTip(this.txtApellidos, "Ingrese los
apellidos del asociado");
            this.ttMensaje.SetToolTip(this.txtSector, "Ingrese el
sector del asociado");
            this.ttMensaje.SetToolTip(this.txtDireccion, "Ingrese la
dirección del asociado");
        }

        private void frmAsociados_Load(object sender, EventArgs e)
        {
            this.Mostrar();
            this.Habilitar(false);
            this.Botones();
        }
        private void MensajeOk(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        private void MensajeError(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        }
        private void Limpiar()
        {
            this.txtCodigo.Text = string.Empty;
            this.txtDni.Text = string.Empty;
            this.txtNombre.Text = string.Empty;
            this.txtApellidos.Text = string.Empty;
            this.txtDireccion.Text = string.Empty;
            this.txtSector.Text = string.Empty;
        }
        private void Habilitar(bool valor)
        {
            this.txtCodigo.ReadOnly = !valor;
            this.txtDni.ReadOnly = !valor;
            this.txtApellidos.ReadOnly = !valor;
            this.txtNombre.ReadOnly = !valor;
            this.txtSector.ReadOnly = !valor;
            this.txtDireccion.ReadOnly = !valor;
            this.txtId_asoc.ReadOnly = !valor;
            this.cbEstado.Enabled = valor;
        }
        private void Botones()
        {
            if (this.IsNuevo || this.IsEditar)
            {
                this.Habilitar(true);
                this.btnNuevo.Enabled = false;
                this.btnInsertar.Enabled = true;
                this.btnEditar.Enabled = false;
                this.btnEliminar.Enabled = false;
            }
            else
            {
                this.Habilitar(false);
                this.btnNuevo.Enabled = true;
                this.btnInsertar.Enabled = false;
                this.btnEditar.Enabled = true;
                this.btnEliminar.Enabled = true;
            }

        }
        private void OcultarColumnas()
        {
            this.dataListado.Columns[0].Visible = false;
            this.dataListado.Columns[1].Visible = false;
        }
        private void Mostrar()
        {
            this.dataListado.DataSource = NAsociados.Mostrar();
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asociados: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void BuscarApellidos()
        {
            this.dataListado.DataSource =
NAsociados.BuscarApellidos(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asociados: " +
Convert.ToString(dataListado.Rows.Count);
        }
```

```
        private void BuscarEstado()
        {
            this.dataListado.DataSource =
NAsociados.BuscarEstado(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asociados: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void BuscarCodigo()
        {
            this.dataListado.DataSource =
NAsociados.BuscarCodigo(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asociados: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void BuscarSector()
        {
            this.dataListado.DataSource =
NAsociados.BuscarSector(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asociados: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void txtBuscar_TextChanged(object sender, EventArgs e)
        {
            if (cbBuscar.Text.Equals("Código"))
            {
                this.BuscarCodigo();
            }
            else if (cbBuscar.Text.Equals("Estado"))
            {
                this.BuscarEstado();
            }
            else if (cbBuscar.Text.Equals("Apellidos"))
            {
                this.BuscarApellidos();
            }
            else if (cbBuscar.Text.Equals("Sector"))
            {
                this.BuscarSector();
            }
        }

        private void btnNuevo_Click(object sender, EventArgs e)
        {
            this.IsNuevo = true;
            this.IsEditar = false;
            this.Botones();
            this.Limpiar();
            this.Habilitar(true);
            this.txtCodigo.Focus();
        }

        private void btnInsertar_Click(object sender, EventArgs e)
        {
            try
            {
                string rpta = "";
                if (this.txtCodigo.Text == string.Empty ||
this.txtCodigo.TextLength < 4)
```

```
                {
                    MensajeError("Falta ingresar algunos datos, serán
remarcados");
                    errorIcono.SetError(txtCodigo, "Ingrese un
Valor");
                    errorIcono.SetError(txtDni, "Ingrese un Valor");

                }
                else
                {
                    if (this.IsNuevo)
                    {
                        rpta =
NAsociados.Insertar(this.txtCodigo.Text, this.txtDni.Text,
this.txtApellidos.Text.Trim(), this.txtNombre.Text.Trim(),
                            this.txtDireccion.Text,
this.txtSector.Text, cbEstado.Text);

                    }
                    else
                    {
                        rpta =
NAsociados.Editar(Convert.ToInt32(this.txtId_asoc.Text),
                            this.txtCodigo.Text, this.txtDni.Text,
this.txtApellidos.Text.Trim(), this.txtNombre.Text.Trim(),
                            this.txtDireccion.Text,
this.txtSector.Text, cbEstado.Text);
                    }

                    if (rpta.Equals("OK"))
                    {
                        if (this.IsNuevo)
                        {
                            this.MensajeOk("Se guardo correctamente el
asociado");
                        }
                        else
                        {
                            this.MensajeOk("Se actualizó el asociado
correctamente");
                        }
                    }
                    else
                    {
                        this.MensajeError(rpta);
                    }

                    this.IsNuevo = false;
                    this.IsEditar = false;
                    this.Botones();
                    this.Limpiar();
                    this.Mostrar();


                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message + ex.StackTrace);
            }
        }
```

```
        private void btnEditar_Click(object sender, EventArgs e)
        {
            if (!this.txtId_asoc.Text.Equals(""))
            {
                this.IsEditar = true;
                this.Botones();
                this.Habilitar(true);
            }
            else
            {
                this.MensajeError("Debe de seleccionar primero el
asociado a modificar");
            }
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.IsNuevo = false;
            this.IsEditar = false;
            this.Botones();
            this.Limpiar();
            this.Habilitar(false);
        }

        private void dataListado_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.ColumnIndex ==
dataListado.Columns["Eliminar"].Index)
            {
                DataGridViewCheckBoxCell ChkEliminar =
(DataGridViewCheckBoxCell)dataListado.Rows[e.RowIndex].Cells["Eliminar
"];
                ChkEliminar.Value =
!Convert.ToBoolean(ChkEliminar.Value);
            }
        }

        private void dataListado_DoubleClick(object sender, EventArgs
e)
        {
            this.txtId_asoc.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["id_asoc"].Value);
            this.txtCodigo.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["codigo"].Value);
            this.txtDni.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["dni"].Value);
            this.txtApellidos.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["apellidos"].Value)
;
            this.txtNombre.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["nombre"].Value);
            this.txtDireccion.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["direccion"].Value)
;
            this.txtSector.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["sector"].Value);
            this.cbEstado.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["estado"].Value);
        }
```

```
        private void chkEliminar_CheckedChanged(object sender,
EventArgs e)
        {
            if (chkEliminar.Checked)
            {
                this.dataListado.Columns[0].Visible = true;

            }
            else
            {
                this.dataListado.Columns[0].Visible = false;
            }
        }

        private void btnEliminar_Click(object sender, EventArgs e)
        {
            try
            {
                DialogResult Opcion;
                Opcion = MessageBox.Show("¿Desea eliminar el asociado
seleccionado?", "Sistema de Control de Asistencias",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

                if (Opcion == DialogResult.OK)
                {
                    string Codigo;
                    string Rpta = "";

                    foreach (DataGridViewRow row in dataListado.Rows)
                    {
                        if (Convert.ToBoolean(row.Cells[0].Value))
                        {
                            Codigo =
Convert.ToString(row.Cells[1].Value);
                            Rpta =
NAsociados.Eliminar(Convert.ToInt32(Codigo));

                            if (Rpta.Equals("OK"))
                            {
                                this.MensajeOk("Se eliminó
correctamente");
                            }
                            else
                            {
                                this.MensajeError(Rpta);
                            }

                        }
                    }
                    this.Mostrar();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message + ex.StackTrace);
            }
        }

        private void txtCodigo_KeyPress(object sender,
KeyPressEventArgs e)
```

```
        {
            if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b')
             {
               e.Handled = true;
            }

           if (txtCodigo.TextLength > 3 && e.KeyChar != '\b')
              {

                e.Handled = true;
              }
        }

        private void txtDni_KeyPress(object sender, KeyPressEventArgs
e)
        {
            if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }

            if (txtDni.TextLength > 7 && e.KeyChar != '\b')
            {

                e.Handled = true;
            }
        }

        private void txtNombre_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }
        }
        private void txtApellidos_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!char.IsLetter(e.KeyChar) &&
!char.IsWhiteSpace(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }
        }

        private void btnImprimir_Click(object sender, EventArgs e)
        {
            frmReporte_Asociados frm = new frmReporte_Asociados();
            frm.ShowDialog();
        }

        private void btnGenerar_Click(object sender, EventArgs e)
        {
            BarcodeLib.Barcode Codigo = new BarcodeLib.Barcode();
            Codigo.IncludeLabel = true;
            panelResultado.BackgroundImage =
Codigo.Encode(BarcodeLib.TYPE.CODE128, txtCodigo.Text, Color.Black,
Color.White, 400, 100);
            btnGuardar.Enabled = true;
```

```
        }

        private void btnGuardar_Click(object sender, EventArgs e)
        {
            Image imgFinal =
(Image)panelResultado.BackgroundImage.Clone();

            SaveFileDialog CajaDeDiaologoGuardar = new
SaveFileDialog();
            CajaDeDiaologoGuardar.AddExtension = true;
            CajaDeDiaologoGuardar.Filter = "Image PNG (*.png)|*.png";
            CajaDeDiaologoGuardar.ShowDialog();
            if (!string.IsNullOrEmpty(CajaDeDiaologoGuardar.FileName))
            {
                imgFinal.Save(CajaDeDiaologoGuardar.FileName,
ImageFormat.Png);
            }
            imgFinal.Dispose();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Excel.Application xlApp;
            Excel.Workbook xlWorkBook;
            Excel.Worksheet xlWorkSheet;
            object misValue = System.Reflection.Missing.Value;

            xlApp = new Excel.Application();
            xlWorkBook = xlApp.Workbooks.Add(misValue);
            xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
            int i = 0;
            int j = 0;


            for (i = 0; i <= dataListado.RowCount - 1; i++)
            {
                for (j = 0; j <= dataListado.ColumnCount - 1; j++)
                {
                    DataGridViewCell cell = dataListado[j, i];
                    xlWorkSheet.Cells[i + 1, j + 1] = cell.Value;
                }
            }

            xlWorkBook.SaveAs("Asociados.xls",
Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue,
misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue,
misValue, misValue, misValue);
            xlWorkBook.Close(true, misValue, misValue);
            xlApp.Quit();

            releaseObject(xlWorkSheet);
            releaseObject(xlWorkBook);
            releaseObject(xlApp);

            this.MensajeOk("Se exportó correctamente a excel");
        }

        private void releaseObject(object obj)
        {
            try
```

```
            {
System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show("Error" + ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }

        private void btnMostrar_Click(object sender, EventArgs e)
        {
            this.Mostrar();
        }
    }
}
```

Anexo 11 Capa presentación – Asociado

```
namespace CapaDatos
{
    public class DReuniones
    {
        private int _Id;
        private DateTime _Fecha;
        private string _Agenda;
        public int Id
        {
            get { return _Id; }
            set { _Id = value; }
        }
        public DateTime Fecha
        {
            get { return _Fecha; }
            set { _Fecha = value; }
        }
        public string Agenda
        {
            get { return _Agenda; }
            set { _Agenda = value; }
        }
        public DReuniones()
        {

        }
        public DReuniones(int id, DateTime fecha, string agenda)
        {
            this.Id = id;
            this.Fecha = fecha;
            this.Agenda = agenda;
        }
        public string Insertar(DReuniones Reunion)
```

```
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCon.Open();

                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "insertar_reunion";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParId = new SqlParameter();
                ParId.ParameterName = "@id";
                ParId.SqlDbType = SqlDbType.Int;
                ParId.Direction = ParameterDirection.Output;
                SqlCmd.Parameters.Add(ParId);

                SqlParameter ParFecha = new SqlParameter();
                ParFecha.ParameterName = "@fecha";
                ParFecha.SqlDbType = SqlDbType.Date;
                ParFecha.Value = Reunion.Fecha;
                SqlCmd.Parameters.Add(ParFecha);

                SqlParameter ParAgenda = new SqlParameter();
                ParAgenda.ParameterName = "@agenda";
                ParAgenda.SqlDbType = SqlDbType.VarChar;
                ParAgenda.Size = 256;
                ParAgenda.Value = Reunion.Agenda;
                SqlCmd.Parameters.Add(ParAgenda);



                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
Ingreso el Registro";

            }

            catch (Exception ex)
            {
                rpta = ex.Message;
            }

            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }
        public DataTable BuscarFecha(String TextoBuscar)
        {
            DataTable DtResultado = new DataTable("reunion");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "buscar_fecha_reunion";
```

```
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParTextoBuscar = new SqlParameter();
                ParTextoBuscar.ParameterName = "@textobuscar";
                ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
                ParTextoBuscar.Size = 50;
                ParTextoBuscar.Value = TextoBuscar;
                SqlCmd.Parameters.Add(ParTextoBuscar);

                SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                SqlDat.Fill(DtResultado);

            }
            catch (Exception)
            {
                DtResultado = null;
            }
            return DtResultado;

        }
        public string Editar(DReuniones Reunion)
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {

                SqlCon.ConnectionString = Conexion.Cn;
                SqlCon.Open();

                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "editar_reunion";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParId = new SqlParameter();
                ParId.ParameterName = "@id";
                ParId.SqlDbType = SqlDbType.Int;
                ParId.Value = Reunion.Id;
                SqlCmd.Parameters.Add(ParId);

                SqlParameter ParFecha = new SqlParameter();
                ParFecha.ParameterName = "@fecha";
                ParFecha.SqlDbType = SqlDbType.Date;
                ParFecha.Value = Reunion.Fecha;
                SqlCmd.Parameters.Add(ParFecha);

                SqlParameter ParAgenda = new SqlParameter();
                ParAgenda.ParameterName = "@agenda";
                ParAgenda.SqlDbType = SqlDbType.VarChar;
                ParAgenda.Size = 256;
                ParAgenda.Value = Reunion.Agenda;
                SqlCmd.Parameters.Add(ParAgenda);

                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
Actualizo el Registro";


            }
            catch (Exception ex)
            {
```

```
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }
        public DataTable Mostrar()
        {
            DataTable DtResultado = new DataTable("reunion");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "mostrar_reunion";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                SqlDat.Fill(DtResultado);

            }
            catch (Exception)
            {
                DtResultado = null;
            }
            return DtResultado;
        }
    }
}
```

Anexo 12 Capa datos - Asamblea

```
using CapaDatos;
namespace CapaNegocio
{
    public class NReuniones
    {
        public static string Insertar(DateTime fecha, string agenda)
        {
            DReuniones Obj = new DReuniones();
            Obj.Fecha = fecha;
            Obj.Agenda = agenda;

            return Obj.Insertar(Obj);
        }
        public static string Editar(int id, DateTime fecha, string
agenda)
        {
            DReuniones Obj = new DReuniones();
            Obj.Id = id;
            Obj.Fecha = fecha;
            Obj.Agenda = agenda;

            return Obj.Editar(Obj);
        }
        public static DataTable BuscarFecha(string textobuscar)
        {
            DReuniones Obj = new DReuniones();
            return Obj.BuscarFecha(textobuscar);
        }
        public static DataTable Mostrar()
        {
            return new DReuniones().Mostrar();
        }

    }
}
```

Anexo 13 Capa megocio – Asamblea

```
using CapaNegocio;
namespace CapaPresentacion
{
    public partial class frmReuniones : Form
    {
        private bool IsNuevo = false;

        private bool IsEditar = false;
        public frmReuniones()
        {
            InitializeComponent();

        }
        private void MensajeOk(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencia", MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        private void MensajeError(string mensaje)
        {
```

```
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencia", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void frmReuniones_Load(object sender, EventArgs e)
        {
            this.Mostrar();
            this.Habilitar(false);
            this.Botones();
        }
        private void Limpiar()
        {
            this.txtAgenda.Text = string.Empty;
            this.txtIdReunion.Text = string.Empty;
        }
        private void Habilitar(bool valor)
        {
            this.txtAgenda.ReadOnly = !valor;
            this.txtIdReunion.ReadOnly = !valor;
        }
        private void Botones()
        {
            if (this.IsNuevo || this.IsEditar)
            {
                this.Habilitar(true);
                this.btnNuevo.Enabled = false;
                this.btnInsertar.Enabled = true;
                this.btnEditar.Enabled = false;
            }
            else
            {
                this.Habilitar(false);
                this.btnNuevo.Enabled = true;
                this.btnInsertar.Enabled = false;
                this.btnEditar.Enabled = true;
            }
        }
        private void BuscarFecha()
        {
            this.dataListado.DataSource =
NReuniones.BuscarFecha(this.dtFecha1.Value.ToString("dd/MM/yyyy"));
            this.OcultarColumnas();
            lblTotal.Text = "Total de Reuniones: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void Mostrar()
        {
            this.dataListado.DataSource = NReuniones.Mostrar();
            this.OcultarColumnas();
            lblTotal.Text = "Total de Reuniones: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void btnNuevo_Click(object sender, EventArgs e)
        {
            this.IsNuevo = true;
            this.IsEditar = false;
            this.Botones();
            this.Limpiar();
            this.Habilitar(true);
        }
        private void OcultarColumnas()
        {
```

```
            this.dataListado.Columns[0].Visible = false;
        }
        private void btnInsertar_Click(object sender, EventArgs e)
        {
            try
            {
                string rpta = "";
                if (this.txtAgenda.Text == string.Empty)
                {
                    MensajeError("Falta ingresar algunos datos, serán
remarcados");

                    errorIcono.SetError(txtAgenda, "Ingresar la
Agenda");

                }
                else
                {
                    if (this.IsNuevo)
                    {
                        rpta = NReuniones.Insertar(dtFecha.Value,
this.txtAgenda.Text.Trim());

                    }

                    else
                    {
                        rpta =
NReuniones.Editar(Convert.ToInt32(this.txtIdReunion.Text),
                            dtFecha.Value, this.txtAgenda.Text);

                    }

                    if (rpta.Equals("OK"))
                    {
                        if (this.IsNuevo)
                        {
                            this.MensajeOk("Se Insertó de forma
correcta el registro");

                        }
                        else
                        {
                            this.MensajeOk("Se Actualizó de forma
correcta el registro");

                        }
                    }
                    else
                    {
                        this.MensajeError(rpta);
                    }

                    this.IsNuevo = false;
                    this.IsEditar = false;
                    this.Botones();
                    this.Limpiar();
                    this.Mostrar();
                }
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show(ex.Message + ex.StackTrace);
            }
        }
        private void btnCancelar_Click(object sender, EventArgs e)
        {
            this.IsNuevo = false;
            this.IsEditar = false;
            this.Botones();
            this.Limpiar();
            this.Habilitar(false);
            this.Mostrar();
        }

        private void btnEditar_Click(object sender, EventArgs e)
        {
            if (!this.txtIdReunion.Text.Equals(""))
            {
                this.IsEditar = true;
                this.Botones();
                this.Habilitar(true);
            }
            else
            {
                this.MensajeError("Debe de seleccionar primero el
registro a Modificar");
            }
        }

        private void dataListado_DoubleClick(object sender, EventArgs
e)
        {
            this.txtIdReunion.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["id"].Value);
            this.dtFecha.Value =
Convert.ToDateTime(this.dataListado.CurrentRow.Cells["fecha"].Value);
            this.txtAgenda.Text =
Convert.ToString(this.dataListado.CurrentRow.Cells["agenda"].Value);
        }

        private void btnBuscar_Click(object sender, EventArgs e)
        {
            this.BuscarFecha();
        }
    }
}
```

Anexo 14 Capa presentación - Asamblea

```
namespace CapaDatos
{
    public class DAsistencias
    {
        private int _Id_asistencias;
        private string _Codigo;
        private string _Apellidos;
        private string _Nombre;
        private string _Sector;
        private string _Hora_entrada;
        private DateTime _Fecha;
        private string _TextoBuscar;

        public string TextoBuscar
        {
            get { return _TextoBuscar; }
            set { _TextoBuscar = value; }
        }
        public int Id_asistencias
        {
            get { return _Id_asistencias; }
            set { _Id_asistencias = value; }
        }
        public string Codigo
        {
            get { return _Codigo; }
            set { _Codigo = value; }
        }

        public string Apellidos
        {
            get { return _Apellidos; }
            set { _Apellidos = value; }
        }
        public string Nombre
        {
            get { return _Nombre; }
            set { _Nombre = value; }
        }
        public string Sector
        {
            get { return _Sector; }
            set { _Sector = value; }
        }
        public string Hora_entrada
        {
            get { return _Hora_entrada; }
            set { _Hora_entrada = value; }
        }
        public DateTime Fecha
        {
            get { return _Fecha; }
            set { _Fecha = value; }
        }
        public DAsistencias() { }
        public DAsistencias(int id_asistencias,string codigo, string
apellidos, string nombre,
             string sector,string hora_entrada, DateTime fecha, string
textobuscar)
        {
            this.Id_asistencias = id_asistencias;
```

```
            this.Codigo = codigo;

            this.Apellidos = apellidos;
            this.Nombre = nombre;
            this.Sector = sector;
            this.Hora_entrada = hora_entrada;
            this.Fecha = fecha;
            this.TextoBuscar = textobuscar;
        }
        public string Insertar(DAsistencias Asistencias)
        {
            string rpta = "";
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCon.Open();

                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "insertar_asistencia";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParId_asoc = new SqlParameter();
                ParId_asoc.ParameterName = "@id_asistencias";
                ParId_asoc.SqlDbType = SqlDbType.Int;
                ParId_asoc.Direction = ParameterDirection.Output;
                SqlCmd.Parameters.Add(ParId_asoc);

                SqlParameter ParCodigo = new SqlParameter();
                ParCodigo.ParameterName = "@codigo";
                ParCodigo.SqlDbType = SqlDbType.VarChar;
                ParCodigo.Size = 4;
                ParCodigo.Value = Asistencias.Codigo;
                SqlCmd.Parameters.Add(ParCodigo);

                SqlParameter ParApellidos = new SqlParameter();
                ParApellidos.ParameterName = "@apellidos";
                ParApellidos.SqlDbType = SqlDbType.VarChar;
                ParApellidos.Size = 50;
                ParApellidos.Value = Asistencias.Apellidos;
                SqlCmd.Parameters.Add(ParApellidos);

                SqlParameter ParNombre = new SqlParameter();
                ParNombre.ParameterName = "@nombre";
                ParNombre.SqlDbType = SqlDbType.VarChar;
                ParNombre.Size = 50;
                ParNombre.Value = Asistencias.Nombre;
                SqlCmd.Parameters.Add(ParNombre);

                SqlParameter ParSector = new SqlParameter();
                ParSector.ParameterName = "@sector";
                ParSector.SqlDbType = SqlDbType.VarChar;
                ParSector.Size = 50;
                ParSector.Value = Asistencias.Sector;
                SqlCmd.Parameters.Add(ParSector);

                SqlParameter ParHora_entrada = new SqlParameter();
                ParHora_entrada.ParameterName = "@hora_entrada";
                ParHora_entrada.SqlDbType = SqlDbType.VarChar;
                ParHora_entrada.Size = 40;
```

```
                ParHora_entrada.Value = Asistencias.Hora_entrada;
                SqlCmd.Parameters.Add(ParHora_entrada);

                SqlParameter ParFecha = new SqlParameter();
                ParFecha.ParameterName = "@fecha";
                ParFecha.SqlDbType = SqlDbType.Date;
                ParFecha.Value = Asistencias.Fecha;
                SqlCmd.Parameters.Add(ParFecha);

                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
Ingreso el Registro";
            }
            catch (Exception ex)
            {
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }
        public DataTable Mostrar()
        {
            DataTable DtResultado = new DataTable("asistencias");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "mostrar_asistencia";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
                SqlDat.Fill(DtResultado);
            }
            catch (Exception)
            {
                DtResultado = null;
            }
            return DtResultado;
        }

        public DataTable BuscarApellidos(DAsistencias Asistencias)
        {
            DataTable DtResultado = new DataTable("asistencias");
            SqlConnection SqlCon = new SqlConnection();
            try
            {
                SqlCon.ConnectionString = Conexion.Cn;
                SqlCommand SqlCmd = new SqlCommand();
                SqlCmd.Connection = SqlCon;
                SqlCmd.CommandText = "buscar_asistencia_apellidos";
                SqlCmd.CommandType = CommandType.StoredProcedure;

                SqlParameter ParTextoBuscar = new SqlParameter();
                ParTextoBuscar.ParameterName = "@textobuscar";
                ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
                ParTextoBuscar.Size = 50;
```

```
            ParTextoBuscar.Value = Asistencias.TextoBuscar;
            SqlCmd.Parameters.Add(ParTextoBuscar);

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);
        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;
    }

    public DataTable BuscarCodigo(DAsistencias Asistencias)
    {
        DataTable DtResultado = new DataTable("asistencias");
        SqlConnection SqlCon = new SqlConnection();
        try
        {
            SqlCon.ConnectionString = Conexion.Cn;
            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "buscar_asistencia_codigo";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParTextoBuscar = new SqlParameter();
            ParTextoBuscar.ParameterName = "@textobuscar";
            ParTextoBuscar.SqlDbType = SqlDbType.VarChar;
            ParTextoBuscar.Size = 50;
            ParTextoBuscar.Value = Asistencias.TextoBuscar;
            SqlCmd.Parameters.Add(ParTextoBuscar);

            SqlDataAdapter SqlDat = new SqlDataAdapter(SqlCmd);
            SqlDat.Fill(DtResultado);
        }
        catch (Exception)
        {
            DtResultado = null;
        }
        return DtResultado;
    }
    public string Eliminar(DAsistencias Asistencias)
    {
        string rpta = "";
        SqlConnection SqlCon = new SqlConnection();
        try
        {

            SqlCon.ConnectionString = Conexion.Cn;
            SqlCon.Open();

            SqlCommand SqlCmd = new SqlCommand();
            SqlCmd.Connection = SqlCon;
            SqlCmd.CommandText = "eliminar_asistencia";
            SqlCmd.CommandType = CommandType.StoredProcedure;

            SqlParameter ParId_asistencias = new SqlParameter();
            ParId_asistencias.ParameterName = "@id_asistencias";
            ParId_asistencias.SqlDbType = SqlDbType.Int;
            ParId_asistencias.Value = Asistencias.Id_asistencias;
            SqlCmd.Parameters.Add(ParId_asistencias);
```

```
                rpta = SqlCmd.ExecuteNonQuery() == 1 ? "OK" : "NO se
se Eliminó el Registro";
            }
            catch (Exception ex)
            {
                rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open)
SqlCon.Close();
            }
            return rpta;
        }
    }
}
```

Anexo 15 Capa datos – Asistencia

```
using CapaDatos;
namespace CapaNegocio
{
    public class NAsistencias
    {
        public static string Insertar(string codigo,string apellidos,
string nombre,
                string sector, string hora_entrada, DateTime fecha)
        {
            DAsistencias Obj = new DAsistencias();
            Obj.Codigo = codigo;

            Obj.Apellidos = apellidos;
            Obj.Nombre = nombre;
            Obj.Sector = sector;
            Obj.Hora_entrada = hora_entrada;
            Obj.Fecha = fecha;
            return Obj.Insertar(Obj);
        }
        public static DataTable Mostrar()
        {
            return new DAsistencias().Mostrar();
        }
        public static DataTable BuscarApellidos(string textobuscar)
        {
            DAsistencias Obj = new DAsistencias();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarApellidos(Obj);
        }
        public static string Eliminar(int id_asistencias)
        {
            DAsistencias Obj = new DAsistencias();
            Obj.Id_asistencias = id_asistencias;

            return Obj.Eliminar(Obj);
        }
        public static DataTable BuscarCodigo(string textobuscar)
        {
```

```
            DAsistencias Obj = new DAsistencias();
            Obj.TextoBuscar = textobuscar;
            return Obj.BuscarCodigo(Obj);
        }
    }
}
```

Anexo 16 Capa negocio – Asistencia

```
using CapaNegocio;
namespace CapaPresentacion
{
    public partial class frmAsistencias : Form
    {
        private bool IsNuevo = false;

        SqlConnection con = new SqlConnection(@"Data Source=.;Initial
Catalog=bd_sisca;Integrated Security=True");
        public frmAsistencias()
        {
            InitializeComponent();
            timer1.Enabled = true;
            this.ttMensaje.SetToolTip(this.txtCodigo, "Ingrese el
código del asociado");
        }
        private void MensajeOk(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        private void MensajeError(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void Limpiar()
        {
            this.txtCodigo.Text = string.Empty;

            this.txtApellidos.Text = string.Empty;
            this.txtNombre.Text = string.Empty;
            this.txtSector.Text = string.Empty;
        }
        private void Habilitar(bool valor)
        {
            this.txtCodigo.ReadOnly = !valor;

            this.txtApellidos.ReadOnly = !valor;
            this.txtNombre.ReadOnly = !valor;
            this.txtSector.ReadOnly = !valor;
        }
        private void OcultarColumnas()
        {
            this.dataListado.Columns[0].Visible = false;

        }
        private void Mostrar()
        {
            this.dataListado.DataSource = NAsistencias.Mostrar();
```

```
            this.OcultarColumnas();
            lblTotal.Text = "Asistentes: " +
Convert.ToString(dataListado.Rows.Count);
        }

        private void frmAsistencias_Load(object sender, EventArgs e)
        {
            this.Mostrar();
            this.Habilitar(false);
            this.btnCerrar.Enabled = false;

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            this.lblHora.Text = DateTime.Now.ToLongTimeString();
            this.lblFecha.Text = DateTime.Now.ToLongDateString();
        }

        private void btnNuevo_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Iniciar el Registro de Asistencias",
"Sistema Control de Asistencia", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            this.IsNuevo = true;
            this.Limpiar();
            this.Habilitar(true);
            this.btnCerrar.Enabled = true;
            this.txtCodigo.Focus();
            this.btnNuevo.Enabled = false;
        }

        private void btnCerrar_Click_1(object sender, EventArgs e)
        {
            MessageBox.Show("Finalizó el Registro de Asistencias",
"Sistema Control de Asistencia", MessageBoxButtons.OK,
MessageBoxIcon.Information); this.btnNuevo.Enabled = false;
            this.txtCodigo.Enabled = false;
            this.btnCerrar.Enabled = false;
        }

        private void txtCodigo_KeyPress(object sender,
KeyPressEventArgs e)
        {
            if (!char.IsDigit(e.KeyChar) && e.KeyChar != '\b')
            {
                e.Handled = true;
            }

            if (txtCodigo.TextLength > 3 && e.KeyChar != '\b')
            {

                e.Handled = true;
            }

            if (e.KeyChar == Convert.ToChar(Keys.Enter))
            {
                string cadSql = "Select * from asociados where
codigo='" + txtCodigo.Text + "'";
                SqlCommand comando = new SqlCommand(cadSql, con);
                con.Open();
```

```
                SqlDataReader leer = comando.ExecuteReader();

                if (leer.Read() == true)
                {

                    txtApellidos.Text = leer["apellidos"].ToString();
                    txtNombre.Text = leer["nombre"].ToString();
                    txtSector.Text = leer["sector"].ToString();
                }

                else
                {
                    txtApellidos.Text = "";
                    txtNombre.Text = "";
                    txtSector.Text = "";

                }
                con.Close();
                try
                {
                    string rpta = "";
                    if (this.txtCodigo.Text == string.Empty ||
this.txtCodigo.TextLength < 4)
                    {
                        MensajeError("Ingrese un código de asociado
correcto");

                        errorIcono.SetError(txtCodigo, "Ingresar
Codigo");

                    }
                    else
                    {
                        if (this.IsNuevo)
                        {
                            rpta =
NAsistencias.Insertar(this.txtCodigo.Text, this.txtApellidos.Text,
                                this.txtNombre.Text,
this.txtSector.Text, this.lblHora.Text, dtFecha.Value);

                        }

                        if (rpta.Equals("OK"))
                        {
                            if (this.IsNuevo)
                            {

                            }

                        }
                        else
                        {
                            this.MensajeError(rpta);
                        }

                        this.Limpiar();
                        this.Mostrar();


                    }
```

```
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message + ex.StackTrace);
                }
            }
        }

                    }
}
```

Anexo 17 Capa presentación – Asistencia

```
using CapaNegocio;
namespace CapaPresentacion
{
    public partial class frmConsulta_Asistencias : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=.;Initial
Catalog=bd_sisca;Integrated Security=True");
        public frmConsulta_Asistencias()
        {
            InitializeComponent();
        }
        private void frmConsulta_Asistencias_Load(object sender,
EventArgs e)
        {
            this.Mostrar();
        }
        private void MensajeOk(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void MensajeError(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de
Asistencias", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void Mostrar()
        {
            this.dataListado.DataSource = NAsistencias.Mostrar();
            this.OcultarColumnas();
            lblTotal.Text = "Total de Asistentes: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void OcultarColumnas()
        {
            this.dataListado.Columns[1].Visible = false;
        }
        private void BuscarApellidos()
        {
            this.dataListado.DataSource =
NAsistencias.BuscarApellidos(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Asistentes: " +
Convert.ToString(dataListado.Rows.Count);
        }
```

```
        private void BuscarCodigo()
        {
            this.dataListado.DataSource =
NAsistencias.BuscarCodigo(this.txtBuscar.Text);
            this.OcultarColumnas();
            lblTotal.Text = "Asistentes: " +
Convert.ToString(dataListado.Rows.Count);
        }
        private void txtBuscar_TextChanged(object sender, EventArgs e)
        {
            if (cbBuscar.Text.Equals("Apellidos"))
            {
                this.BuscarApellidos();
            }

            else if (cbBuscar.Text.Equals("Código"))
            {
                this.BuscarCodigo();
            }
        }
        private void btnSeleccionar_Click(object sender, EventArgs e)
        {
            for (int i = 0; i <= dataListado.Rows.Count-1; i++)
            {
                dataListado.Rows[i].Cells[0].Value = true;
            }
        }
        private void btnEliminar_Click(object sender, EventArgs e)
        {
            DialogResult Opcion;
            Opcion = MessageBox.Show("¿Desea eliminar la lista de
asistencia?", "Sistema de Control de Asistencias",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
            foreach(DataGridViewRow fila in dataListado.Rows)
            {
                string Codigo;
                string Rpta = "";

                if(Convert.ToBoolean(fila.Cells["Eliminar"].Value))
                {
                    Codigo =
Convert.ToString(fila.Cells["id_asistencias"].Value);
                    Rpta =
NAsistencias.Eliminar(Convert.ToInt32(Codigo));
                }
            }
            this.Mostrar();
        }
        private void btnQuitar_Click(object sender, EventArgs e)
        {
            for (int i = 0; i <= dataListado.Rows.Count-1; i++)
            {

if(Convert.ToBoolean(dataListado.Rows[i].Cells[0].Value))
                {
                    dataListado.Rows[i].Cells[0].Value = false;
                }
            }
        }
        private void btnMostrar_Click(object sender, EventArgs e)
        {
```

```
            this.Mostrar();
        }
    }
}
```

Anexo 18 Capa presentación – Consulta de asistencia

```
using CapaNegocio;
namespace CapaPresentacion
{
    public partial class frmConsulta_Inasistencias : Form
    {
        SqlConnection con = new SqlConnection(@"Data Source=.;Initial
Catalog=bd_sisca;Integrated Security=True");
        public frmConsulta_Inasistencias()
        {
            InitializeComponent();
        }
        private void frmConsulta_Inasistencias_Load(object sender,
EventArgs e)
        {

        }
        private void MensajeOk(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de Asistencias",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        private void MensajeError(string mensaje)
        {
            MessageBox.Show(mensaje, "Sistema de Control de Asistencias",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void btnInasistentes_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = new SqlCommand("select t1. * from asociados
as t1 left join asistencias as t2 on t1.codigo=t2.codigo where t2.codigo
is null", con);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            dtLista.DataSource = dt;
            con.Close();
            this.dtLista.Columns[0].Visible = false;
            lblFaltas.Text = "Total de Inasistentes: " +
Convert.ToString(dtLista.Rows.Count);
            this.dtLista.Columns[4].Visible = false;
        }
    }
}
```

Anexo 19 Capa presentación – Consulta de inasistencia

```
private void btnExportar_Click(object sender, EventArgs e)
        {
            Excel.Application xlApp;
            Excel.Workbook xlWorkBook;
            Excel.Worksheet xlWorkSheet;
            object misValue = System.Reflection.Missing.Value;

            xlApp = new Excel.Application();
            xlWorkBook = xlApp.Workbooks.Add(misValue);
            xlWorkSheet =
(Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);
            int i = 0;
            int j = 0;


            for (i = 0; i <= dtLista.RowCount - 1; i++)
            {
                for (j = 0; j <= dtLista.ColumnCount - 1; j++)
                {
                    DataGridViewCell cell = dtLista[j, i];
                    xlWorkSheet.Cells[i + 1, j + 1] = cell.Value;
                }
            }

            xlWorkBook.SaveAs("Asociados.xls",
Excel.XlFileFormat.xlWorkbookNormal, misValue, misValue, misValue,
misValue, Excel.XlSaveAsAccessMode.xlExclusive, misValue, misValue,
misValue, misValue, misValue);
            xlWorkBook.Close(true, misValue, misValue);
            xlApp.Quit();

            releaseObject(xlWorkSheet);
            releaseObject(xlWorkBook);
            releaseObject(xlApp);

            this.MensajeOk("Se exportó correctamente a excel");
        }
        private void releaseObject(object obj)
        {
            try
            {

System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show("Error" + ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }
```

Anexo 20 Capa presentación – Generar reporte