Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema de Gestión de la Información para el Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autora: Isabel Sánchez Pérez.

Tutores: Ms.Yarina Amoroso Fernández.

Ing. Rudel Cárdenas Díaz.

Co-tutor: Ing. Jorge Yunier Jorrín Perdomo.

Ciudad de la Habana, Junio del 2008

"... siembra escuelas y tendrás decenas de miles de científicos..."

DATOS DE CONTACTO

Tutor: Ms.Yarina Amoroso Fernández

Graduada en el curso 1984- 1985 de Licenciatura en Derecho. Master en Ciencias Jurídicas. Mención en Derecho Público. Universidad de Valencia. España. Diploma Superior “Políticas Públicas de Internet”. FLACSO sede Ecuador. Diploma DEA de Estudios de Doctorado "Aspectos éticos y jurídicos de la información digitalizada". Universidad de Valencia. España. Radica en la Universidad de las Ciencias Informáticas como especialista de la Infraestructura Productiva. Investigador Titulado por la Universidad de Valencia, Facultad de Derecho, Departamento de Filosofía del Derecho. Miembro del Consejo Nacional de la Unión Nacional de Juristas de Cuba. Presidente de la Sociedad Cubana de Derecho e Informática. Vicepresidente de la Federación Iberoamericana de Asociaciones de Derecho e Informática. Vicepresidente del Instituto Iberoamericano de Derecho e Informática. Miembro del Instituto para el Gobierno y la Sociedad, Pisa, Italia. Miembro de la Red Académica: "LEFIS E-government", "Legal framework for the infomation society" / Marco legal para la Sociedad de la Información. Miembro del Comité Cubano de Bioética.

Tutor: Ing. Rudel Cárdenas Díaz

Graduado en el año 2003 como Ingeniero Informático, posee la categoría docente Instructor y ha tutelado 7 trabajos de diploma.

Cotutor: Ing. Jorge Yunier Jorrín Perdomo.

Graduado en el año 2008 como Ingeniero Informático en la Universidad de las Ciencias Informáticas, posee la categoría, docente en Adiestramiento.

AGRADECIMIENTOS

Al Comandante en Jefe por idear una Universidad como la UCI.

A mis tutores por la confianza y el tiempo empleado en este trabajo de diploma, especialmente a mi cotutor.

A los amigos que contribuyeron con su esfuerzo al logro de este trabajo de diploma.

A mi familia por el apoyo incondicional que me han brindado.

A mi novio por todas las atenciones y ayuda prestadas.

A todas aquellas personas anónimas que han contribuido a mi formación como revolucionaria.

DEDICATORIA

A mis padres por inculcarme el espíritu del cumplimiento y ser el motor exigente e impulsor de mi obra.

A mi familia por siempre confiar en mí, cuando incluso yo no confiaba.

A mi hermano por su preocupación y apoyo incondicional.

A mi novio por estar siempre ahí cuando lo he necesitado.

A mis amigos, a los que nunca olvidaré y llevaré presente.

RESUMEN

El presente trabajo de diploma expone una propuesta de Sistema de Gestión de la Información para el Proceso de Diagnóstico a entidades legales, de la República Bolivariana de Venezuela. Este persigue el objetivo de ayudar a manejar información, de manera segura e íntegra, para potenciar la toma de decisiones vinculadas a los procesos de modernización e informatización de dichas instituciones y contribuir en específico, al proceso de desarrollo de software del proyecto Servicio Autónomo de Registros y Notarías (SAREN) de la facultad 3 en su fase II. También se tiene en cuenta la posibilidad de generalizar la herramienta a cualquier tipo de entidad, en base a esto se modelan funcionalidades, que sirven de apoyo, a la implementación futura, de una herramienta que proporcione la posibilidad de elaborar planillas, que gestionen información personificada con diversos objetivos.

Para realizar el sistema se utiliza la metodología ágil de desarrollo de software Scrum y se hace un estudio de los sistemas existentes, las herramientas y las diversas terminologías referentes al desarrollo de software, de manera que se seleccionen las adecuadas para la construcción del sistema en cuestión.

Palabras Claves: Sistema de Gestión de la Información, Proceso de Diagnóstico, Toma de Decisiones.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El proceso de diagnóstico en la comunidad informática contribuye de manera objetiva en el proceso de desarrollo de software, es donde se recolecta información objetiva que permite procesar y analizar datos para obtener resultados satisfactorios y gestionar riesgos en los procesos de despliegue de soluciones informáticas. Con este proceso se da a conocer cuáles son las condiciones que imperan en las entidades a informatizar así como las dificultades y disfuncionalidades fundamentales que pueden obstaculizar el proceso de despliegue de la solución.

El proyecto de Servicio Autónomo de Registros y Notarías (SAREN) pertenece a los convenios del ALBA efectuados entre Cuba y Venezuela, desempeñado por la UCI, el mismo finalizó su primera fase con el despliegue de una solución informática, el proceso de diagnóstico a los registros de esta primera fase, mantuvo toda la información recogida de forma manual en formato físico, provocando que la tarea fuese más engorrosa y menos confortable. La gestión de esta información recolectada fue fruto de un proceso extenso y en ocasiones agotador, debido a la manipulación de tantos datos para la confección del Informe Final y la gestión de la información para la toma de decisiones. La experiencia en el despliegue arrojó que pudo haberse recogido mejor la información e incluso aumentar los datos recolectados.

Previendo las dificultades enfrentadas en el Diagnóstico realizado, correspondiente a la primera fase, para la segunda, se necesita optimizar este proceso, con el principal objetivo de potenciar la toma de decisiones en el venidero Proceso de Diagnóstico.

Por lo antes expuesto se define como **problema científico** la siguiente interrogante: ¿Cómo contribuir en el proceso de diagnóstico a entidades legales de la República Bolivariana de Venezuela, para potenciar la toma de decisiones y el proceso de desarrollo de software?

Para darle solución al problema científico, este trabajo tiene como **objetivo:** Diseñar e implementar un Sistema de Gestión de la Información para potenciar la toma de decisiones en el proceso de diagnóstico a entidades legales de la República Bolivariana de Venezuela y contribuir con esto al proceso de desarrollo de software.

El **objeto de estudio** del presente trabajo se define como: El proceso de desarrollo del software.

Enfocado como **campo de acción** en: El diseño e implementación de Sistemas de Gestión de la Información (SGI) para el Proceso de Diagnóstico a entidades legales.

La hipótesis quedaría formulada de la siguiente forma**:** El diseño e implementación de un sistema de gestión de la Información, potenciará la toma de decisiones en el proceso de diagnóstico a entidades legales venezolanas y contribuirá al proceso de desarrollo de software de la II Fase del proyecto SAREN, perteneciente a la facultad 3.

Las **tareas investigativas** trazadas para dar solución al objetivo expuesto anteriormente, consisten en las siguientes:

- Análisis del proceso de diagnóstico de la primera fase del proyecto SAREN y la documentación generada para su realización, así como las planillas que sirvieron de soporte documental para unificar la recolección de los datos y su posterior procesamiento.
- Análisis de la necesidad de la creación de un sistema de gestión de Información para el proceso de diagnóstico a entidades legales.
- Análisis de la metodología y las herramientas que se usarán en el desarrollo del presente trabajo, entre ellas, técnicas de programación y lenguajes.
- Diseño e implementación de un Sistema de Gestión de la Información para el Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela.
- Validación a través de pruebas, del Sistema de Gestión de Información para el Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela.

Para realizar las tareas se emplearon los siguientes métodos:

Métodos teóricos:

- Análisis y síntesis: para el procesamiento de la información y arribar a las conclusiones de la investigación así como para precisar las características del software.

- Histórico - Lógico: permite, en la etapa facto perceptual, conocer desde sus orígenes hasta el instante actual, el objeto y campo de acción que se estudia.

- Hipotético - Deductivo: para la formulación de la hipótesis.

- Sistémico: es un método teórico que sirvió para la descomposición del sistema en módulos que luego se integró para formar parte de la herramienta Sistema de Gestión de la Información para el Diagnóstico a entidades legales, resultante como un todo.

Aportes prácticos esperados del trabajo.

El sistema que resulte de la investigación debe ser capaz de realizar todas las funcionalidades que necesita el cliente (Ver capítulo 2) y debe lograr potenciar la toma de decisiones en el proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela y contribuir al proceso de desarrollo del software del proyecto SAREN de la facultad 3.

Contenido de los capítulos.

La tesis está estructurada en 3 capítulos.

En el Capítulo 1 se expone la relación entre los conceptos proceso de diagnóstico y sistema de gestión de la información, la utilidad de un sistema de gestión de la información para la toma de decisiones y la importancia de un proceso de diagnóstico en el proceso de desarrollo de software, la factibilidad de un sistema de gestión de la información para el proceso de diagnóstico a entidades legales de Venezuela, así como un breve estudio sobre las herramientas, lenguajes y metodologías usadas para el desarrollo del software.

En el Capítulo 2 se hace una descripción de la propuesta de solución de este trabajo para ello se describen los procesos del negocio propuesto, los roles involucrados y los documentos empleados. Se definen las funcionalidades y se describen detalladamente, siguiendo los estándares de la metodología utilizada.

En el Capítulo 3 se aborda la construcción de la solución; se describe la arquitectura utilizada, los patrones de diseño y elementos de seguridad empleados, se presentan los diagramas de clases de diseño, los principios de diseño, el estándar de codificación que se tuvo en cuenta para el desarrollo del sistema, así como la validación del diseño y de la funcionalidad del sistema en cuestión.

Además esta tesis posee Conclusiones, Recomendaciones, Referencias Bibliográficas, Anexos y Glosario de Términos.

CAPÍTULO 1.FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En este capítulo se hace referencia a la definición de sistemas de gestión de información y su vinculación a los procesos de diagnósticos a entidades legales, se relacionarán terminologías referentes a estas últimas, necesarias para la comprensión del presente informe. Por otra parte se incluirán todos los aspectos teóricos que soportan la aplicación, incluyendo el análisis comparativo de las metodologías de desarrollo de software, las tecnologías y las herramientas utilizadas en el proceso de desarrollo de este trabajo, además se estudiarán aspectos referentes como los paradigmas de programación existentes y los patrones de diseño y arquitectura utilizados, se fundamentará en cada caso comparativo el aspecto elegido.

1.2 Fundamentación del Tema

1.2.1 Génesis y Origen del Proceso de Diagnóstico.

Etimológicamente el concepto de diagnóstico proviene del griego, tiene 2 raíces, dia- que significa a través de y gignoskein que significa conocer, así etimológicamente diagnóstico significa conocer a través de. El concepto de este significado es la identificación de la naturaleza o esencia de una situación o problema y de la causa posible o probable del mismo o es el análisis de la naturaleza de algo.

El proceso de diagnóstico es un razonamiento dirigido en la determinación de la naturaleza, causas u origen de un fenómeno. El mismo se aplica a un objeto determinado, generalmente para solucionar un problema. El desarrollo de dicho proceso permite experimentar en el problema, cambios cualitativos y cuantitativos, los que tienden a la solución del mismo [1].

Consta de varias etapas dialécticamente relacionadas; por ejemplo; Evaluación, Procesamiento mental de la información, Intervención y Seguimiento.

Cualquier afirmación o conclusión que se haga acerca de la causa o esencia del estado de un fenómeno, situación o problema, se puede plantear que se está en presencia de un diagnóstico.
El mismo se determina al realizar una serie de pasos sucesivos, los cuales son:

- Observación
- Descripción (Es necesario un lenguaje).
- Agrupación
- Identificación de relaciones significativas.
- Observación crítica de los atributos (Características).
- Selección de unas prioridades.
- Desarrollo de un criterio.
- Desarrollar una taxonomía (Para identificar las clasificaciones).
- Diagnosticar.

Para diagnosticar se necesitan conocimientos teóricos provenientes del marco conceptual de la naturaleza del fenómeno. Otro aspecto es el uso del razonamiento, adquirido de los métodos científicos que la ciencia brinda para la solución del problema y para apoyar el mismo se necesita reunir datos a través de la experiencia o intuición acerca del problema de investigación.
El diagnóstico dentro del método científico corresponde a la etapa de formulación de hipótesis, el cual contiene diferentes niveles de complejidad, está demostrado actualmente que se necesita del método científico para diagnosticar.
Estos métodos permiten observar hechos significativos, sentar hipótesis que expliquen satisfactoriamente estos hechos y deducir las mismas. Las etapas que constan estos métodos son:

- Realizar un estudio completo del problema a tratar.
- Definición lo más profunda y específica del posible problema.
- Búsqueda y recopilación de información pertinente al problema, tanto en lo referente a datos de interés como a los conocimientos existentes sobre este (bibliografía).
- Análisis y explicación de los datos.
- Formulación de hipótesis o conclusiones probables.
- Contrastación de las hipótesis, repetición de las experiencias deducidas de estas con los resultados positivos que avalarán el grado de certeza de las hipótesis.

- Elaboración de la teoría a la vista de las hipótesis enunciadas.

Estas etapas de los métodos de investigación se concentran a través del proceso de diagnóstico.

1.2.2 Sistema de Gestión de Información.

Una definición muy acertada para un Sistema de Gestión de Información (SGI) podría consistir en un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades que se realizan en una organización o para automatizar los procesos de trabajo que se efectúan dentro de la misma. Un SGI requiere de las técnicas informáticas para un correcto funcionamiento. La utilización del hardware necesario es de suma importancia para contar con la eficiencia requerida por el SGI. Se debe tener en cuenta además el recurso humano, que es quien interactúa con el sistema y obtiene beneficios del mismo. Los Sistemas de Gestión de Información están caracterizados por reproducir cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información [1].

La entrada de información en un SGI es un proceso donde el sistema recupera datos necesarios para procesar cierta información, esta actividad puede ser de forma manual o automática, dependiendo de su origen puede ser efectuada por un usuario directamente, o por otros sistemas internos o externos a la organización donde está desplegado el sistema en cuestión. El almacenamiento es fundamental en el SGI pues permite persistir toda la información gestionada en el proceso de entrada, haciéndola disponible para otros usuarios y sistemas que tengan privilegios sobre dicha información. El procesamiento de información consiste en la capacidad que posee el SGI de gestionar el flujo de información de acuerdo a restricciones y operaciones establecidas por el lenguaje de programación. Por último el proceso de salida de un SGI es lo que posibilita utilizar la información previamente entrada o almacenada ya sea mostrándola en una interfaz al usuario o sirviendo de entrada a otros sistemas internos o externos [1].

1.2.3 La función del Proceso de Diagnóstico en el Sistema de Gestión de Información.

La idea de este apartado es darle respuesta a las siguientes preguntas:

¿Qué tan relacionados pueden estar un proceso de diagnóstico y un sistema de gestión de información?

¿Qué aspectos del Proceso de diagnóstico se pueden visualizar a través de un SGI?

Como se explica en el apéndice anterior un SGI posee 4 actividades fundamentales, el Proceso de Diagnóstico a su vez posee 4 etapas dialécticas por las que se guía, la Evaluación, Procesamiento Mental de la Información, Intervención y Seguimiento.
Un estudio comparativo entre las etapas y los pasos a seguir en un Proceso de Diagnóstico agrupan estos de la siguiente manera. Es necesario aclarar que esto es solo una forma de agrupar, derivado de la percepción del autor del presente trabajo, con el objetivo de demostrar la estrecha relación que puede existir entre los conceptos analizados, no es absoluto , puede existir otra opinión al respecto pero igual siguen siendo los pasos y las etapas a seguir en dicho proceso.
En la etapa de la Evaluación, se pueden observar pasos como la *Observación, Descripción y Agrupación* de la información referente al fenómeno diagnosticado.

En la Etapa del Procesamiento de la Información se ponen de manifiesto la *Identificación de relaciones significativas*, la *Observación crítica de los atributos* o Características del objeto, la S*elección de unas prioridades, el Desarrollo de un criterio*, la necesidad de *Desarrollar una taxonomía* (para identificar las clasificaciones) y como paso final *Diagnosticar*.

Las Etapas de Intervención y Seguimiento contribuyen al objetivo del Diagnóstico de cambiar la naturaleza del objeto diagnosticado.
Llega el turno de distinguir como las etapas del Proceso de Diagnóstico se encuentran visualizadas en un SGI.
Evaluación: para evaluar la naturaleza de algo es necesario adquirir datos referentes al objeto en cuestión, esto se relaciona con una de las 4 actividades básicas de los SGI, ***la Entrada de información*** para la cuál es necesario evaluar el aspecto a tratar y recoger los datos necesarios, que puedan marcar pautas en el objetivo del SGI y servirán como entrada al mismo.
Procesamiento mental de la información, es donde se analiza toda la información evaluada y recolectada, teniendo esta etapa similitud con la actividad de ***almacenamiento y procesamiento*** de los SGI, proceso por el que pasa toda la información entrada y evaluada.

Para la etapas de **Intervención y Seguimiento** del Proceso de Diagnóstico es necesario luego de ser procesada, evaluada y analizada la información, sacar conclusiones y tomar decisiones a cerca de la

causa o esencia del estado del fenómeno, situación o problema del mismo, para de esta forma intervenir en la naturaleza del objeto, darle seguimiento y modificarla, que es en sí el objetivo que se deriva del resultado del diagnóstico, dicho resultado se corresponde con la 4 actividad básica de los SGI, ***la Salida de Información***, consistente en luego de un previo procesamiento y análisis de la información entrada y almacenada, visualizar a través de reportes los resultados para los que fue concebido el sistema.

El diagnóstico permite recabar información pertinente, analizarla e identificar un conjunto de variables que permitan establecer conclusiones [1].

Esta opinión corrobora cuán relacionados se encuentran ambos conceptos estudiados pues para recabar esta información hay que recogerla y almacenarla, analizarla, procesarla y luego obtener como salida un conjunto de variables que permitan llegar a conclusiones para la toma de decisiones, entonces con todas estas actividades involucradas, se está en presencia de un sistema de gestión de Información y con esto quedan las interrogantes abarcadas.

1.2.4 La utilidad del Sistema de Gestión de Información para la toma de decisiones en un Proceso de Diagnóstico.

La utilidad del sistema de gestión de información para la toma de decisiones en un Proceso de Diagnóstico, está dada por varios aspectos.

Se tiene la importancia de la información en la toma de decisiones la cual queda patente en la definición dada por Forrester (1972) y asumida en múltiples obras que tratan sobre el tema. Menguzatto (1985) cita a Forrester, quien define la decisión como " el proceso de transformación de la información en acción ", Proceso de cambio que denominamos toma de decisiones, siendo la decisión el conjunto de acciones adoptadas en un momento específico, como resultado de la aplicación de ciertas reglas y políticas a las condiciones particulares existentes en el momento.

El proceso de toma de decisiones es un acto volitivo en el que se evidencian cualidades volitivas de los seres humanos, esenciales en este proceso, tales como: independencia, decisión, perseverancia y autodominio. El proceso atraviesa por diferentes *fases*:

- Definición del problema.
- Análisis de la información disponible.
- Desarrollo de las soluciones alternativas.

- Selección de la decisión.
- Implantación de la estrategia elegida.

Definición del problema: Significa conocer las interioridades de la organización, lo que se complementa con el conocimiento del entorno externo de la organización y en qué medida estos pueden afectar a la solución del problema y con una definición de los objetivos perseguidos.

Análisis de la información disponible: Debe estar relacionada con el problema, los datos a utilizar pueden ser internos, externos del entorno, será necesario, de poseerla, combinar la información cuantificada con la no cuantificada o basada en experiencias.

Desarrollo de soluciones alternativas: Decidir significa también buscar soluciones alternativas al problema. Significa formular posibles hipótesis y evaluarlas, se busca la relación causa-efecto con el problema planteado, es decir, cual es la respuesta previsible a una alternativa de solución dada.

Selección de la decisión: Se ha de producir una vez que tengamos varias alternativas, mientras más alternativas haya, más correcta ha sido la fase anterior y en función del objetivo se revela la más adecuada.

Implantación de la estrategia elegida: Significa la puesta en práctica de la alternativa seleccionada, ya que por correcta que sea la solución sino se lleva a la práctica convertida en acción no tendrá ningún efecto, generalmente esto significa hacer cambios o transformaciones en la organización que pueden no ser bien acogidas (resistencia al cambio), corriendo el riesgo de su inoperancia, por lo que esta última fase debe ser cuidadosamente planificada, con el fin de lograr el éxito deseado.

Pero esto no es posible hacerlo funcionar si no se cuenta con una información útil y adecuada, que permita decidir a tiempo y con el menor riesgo e incertidumbre posibles. Esta información debe ser cuidadosamente recolectada, analizada y procesada, de aquí que el sistema de gestión de la información se convierta en una materia prima imprescindible para que pueda desarrollarse un proceso de toma de decisiones participativo, democrático y con calidad en función de los intereses de la organización que lo dirige y en este caso como parte de un Proceso de Diagnóstico, en el cual la toma de decisiones constituye un aspecto esencial a la hora de diagnosticar.

1.2.5 Sistema de Gestión de Información en el mundo.

En el mundo existen diferentes SGI, todos de acuerdo con las metas trazadas para su uso y con las organizaciones que los implementan.

Entre ellos se cuentan:

Proyecto SIRED (Sistema de Información de Redes)

Esta nueva solución diseñada conjuntamente por la Unidad Funcional de Sistemas de Información y por la Unidad de Estructura Territorial, integra la gestión y mecanización de cualquier operación o actividad que deban realizar las oficinas de la red en cualquier punto de España, procedente de los diversos negocios de seguro directo incluidos en las unidades operativas del grupo en este país.

Este sistema de información y gestión interorganizativa se ha desarrollado para la realidad interna de un grupo asegurador con diferentes entidades y líneas de negocio, buscando optimizar la gestión comercial de sus redes.

INFOTEC (Información Tecnológica)

Es un sistema de Información Científica y Tecnológica del sector agropecuario en las Américas, diseñado con la idea de proveer información a la comunidad agropecuaria americana acerca de aspectos referentes al tema. Tiene gran popularidad y goza de méritos y alabanzas en cuanto a su sector se refiere.

Cada sistema de gestión de la información del mundo se rige por las necesidades y características definidas por la organización por la que fue creado, es por eso que no resulta necesario abarcar más acerca de los diferentes ejemplares que existen, pues todos se basan en el mismo objetivo.

1.2.6 Sistema de Gestión de Información en Cuba.

Cuba no está al margen del desarrollo de los Sistemas de Gestión de la Información, de hecho ha establecido la Política Nacional de Información(PNI), lo cual sienta pautas a seguir por las diferentes empresas, organismos e instituciones, inmersas en el proceso de Perfeccionamiento Empresarial.

Este proceso se inicia en el país a finales del siglo XX e inicios del XXI, a propósito de este se emiten colateralmente el Decreto Ley 187 de Perfeccionamiento Empresarial, que establece entre sus subsistemas el de *Información Interna*. Posteriormente se emite el Decreto Ley 297 sobre el Control Interno de las Organizaciones el que establece como cuarto elemento el de *Información y Comunicación* y la Resolución 221 del 2001 del Sistema Nacional de Archivos, cuyo objetivo principal es conservar la

memoria de las organizaciones, entre otras legislaciones. Este cuerpo legal constituye el sustento de los Sistema de Gestión de la Información que aunque no es suficiente, sí constituyen un importante avance en Cuba en cuanto a desarrollo y cumplimiento de su PNI.

Otro ejemplo de SGI en Cuba es el Sistema Automatizado para el Diagnóstico de la Gestión de Información y Conocimiento en la Empresa, basado en la propuesta del Modelo operacional de gestión de información y conocimiento para la empresa cubana en perfeccionamiento (MOGICEP).

También cuenta Cuba con una herramienta para el diagnóstico de la gestión de información en la empresa, mediante proyectos de consultoría de la empresa de Gestión del Conocimiento y la tecnología (GECYT).

Son todos ejemplos de la utilidad de los SGI para el desarrollo de la Revolución Cubana.

1.2.7 Sistema de Gestión de Información para el proceso de Diagnóstico a entidades legales en la República Bolivariana de Venezuela.

Los sistemas de gestión de información se identifican con los objetivos que persiguen y se caracterizan por ser totalmente personalizados, debido a que las entidades que los requieren gestionan información referente a sus necesidades y peculiaridades.

Es por este motivo que un SGI para realizar diagnósticos es particular y referente a la naturaleza del objeto que se determina y a los fines que conlleva su implantación.

En este trabajo se tiene como objetivo diagnosticar entidades legales específicamente ubicadas en la República Bolivariana de Venezuela. Es necesario conocer los siguientes términos:

Las entidades legales o jurídicas: se rigen por principios legales o judiciales, creadas bajo la ley Federal o Estatal que es dueño de tierras o mercancía agrícola, productos o ganados.

Entre las entidades legales de Venezuela se encuentran:

Registros Públicos: se encargan de mantener la legalidad registral de los inmuebles, con la finalidad de garantizar la seguridad jurídica, la libertad contractual y el principio de legalidad de los Actos o negocios jurídicos, bienes y derechos reales.

Registros Mercantiles: oficina pública que contribuye en general a la tutela del principio de seguridad del tráfico mercantil, desarrollando además otras funciones complejas. Los Registros Mercantiles Territoriales, situados básicamente en las capitales de provincia, se ocupan de la inscripción de los empresarios y demás sujetos establecidos por la Ley y de los actos y contratos determinados por el Reglamento del Registro Mercantil; también de la legalización de los libros de contabilidad, el nombramiento de expertos independientes y de auditores de cuentas y el depósito y publicidad de los documentos contables. El Registro mercantil Central se ocupa entre otras tareas, de la publicación del Boletín Oficial del Registro mercantil, de la centralización y publicación de la información registral y de la expedición de certificaciones acerca de la denominación social.

La inscripción, que se practica en virtud de documento público, tendrá carácter obligatorio, salvo en los casos en que la Ley disponga lo contrario. Son principios rectores del Registro Mercantil los de legalidad, legitimación, fe pública, disponibilidad, prioridad, tracto sucesivo y publicidad formal.

Notarías: Oficina legal donde radica un funcionario público autorizado para dar fe de los contratos, testamentos y otros actos extrajudiciales, conforme a las leyes. Habilitado para practicar las correspondientes a la ejecución de actos, acuerdos o decretos judiciales.

En Venezuela se está fusionando una revolución digital en cuanto a lo referente a dichas entidades jurídicas, la cual tiene como objetivo la Implantación de la Nueva Oficina Judicial, la Modernización de la Gestión de Registros, la Gestión de la descentralización Judicial, la Transparencia Judicial, los Sistemas de Gestión Procesal, sistema de Indicadores Estratégicos en Materia Judicial o la nueva Secretaría Digital entre otras muchos sistemas que deben ser desarrollados y que abarcan todas las ramas de desempeño de la actividad jurídica.

El problema no se restringe a mejorar la estructura vigente en la institución, sino, también y principalmente, a dotarla de condiciones compatibles con las exigencias y expectativas del futuro.

Como parte del convenio entre Cuba y Venezuela, la Universidad de las Ciencias Informáticas (UCI) es partícipe en este proyecto de modernización de las entidades legales venezolanas. Producto de esta colaboración todos los Registros Inmobiliarios y Mercantiles cuentan con un sistema por el cual se rigen sus funciones, dicho sistema es el fruto de la Primera Fase del proyecto de SAREN enmarcado en la UCI, el cual en su segunda fase tiene como objetivo desarrollar un sistema para los Registros Principales y las Notarías.

Es por todo esto que surge la necesidad de diagnosticar estas últimas entidades con el objetivo de:

- Recolectar información objetiva que permita procesar y analizar datos para obtener resultados satisfactorios y gestionar riesgos en el proceso de Despliegue de la Solución Informática.
- Dar a conocer cuáles son las condiciones que imperan en las entidades a informatizar así como las dificultades y disfuncionalidades fundamentales que pueden obstaculizar el proceso de Despliegue de la Solución.
- Intensificar un proceso de concertación necesario entre el equipo que provee la solución y el receptor para identificar vías de solución de inconvenientes presentables o a presentar.
- Analizar cuáles son modificaciones legales necesarias para enfrentar al problema de raíz y cuáles requieren de acciones del receptor.
- Sugerir soluciones prácticas, en algunas áreas, que permitan una mayor fluidez del proceso.

Otro objetivo de esta segunda fase es mejorar el proceso de diagnóstico anterior utilizando las tecnologías de la información y los servicios que prestan de manera que se optimice dicho proceso y se potencie la toma de decisiones en el mismo, con la aplicación de un SGI personificado para diagnósticos a entidades legales.

A continuación serán explicadas la metodología y las herramientas usadas para el desarrollo del sistema de gestión de Información para el diagnóstico a las entidades legales. Es válido aclarar que aunque se utilizará principalmente para los Registros Principales y las Notarías de la República Bolivariana de Venezuela, dicho sistema es una herramienta genérica para todas las entidades jurídicas de este país.

1.3 Principales terminologías para el desarrollo de software.

Para crear la aplicación se ha tenido en cuenta el estudio de diferentes metodologías, herramientas y lenguajes de programación para su desarrollo, esto permitirá que al elegir la apropiada en cada uno de los casos, se realice el diseño e implementación de la herramienta con mayores éxitos. Antes de comenzar a adentrarnos en el mundo del software, se darán a conocer algunos conceptos necesarios, que en apéndices posteriores serán ampliamente desarrollados, como:

1. ¿Qué es el proceso de desarrollo de software?
2. ¿Qué es la programación orientada a objetos?
3. ¿Qué es una metodología de desarrollo de software?

4. ¿Qué es un IDE de programación?
5. ¿Qué es un gestor de base de datos?

1.3.1 ¿Qué es el proceso de desarrollo de software?

Un proceso determina quien está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. Un proceso efectivo, captura y presenta las mejores prácticas que el estado actual de las tecnologías permite. En consecuencia, reduce el riesgo y hace el producto más predecible. El proceso debe estar ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo en el que se encuentran implicados. [2]

En el proceso de desarrollo de software las necesidades del usuario son traducidas en requerimientos de software, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo.

El proceso de desarrollo de software no es único. No existe un proceso universal que sea efectivo para todos los contextos de proyectos de desarrollo. A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos. [3]

- Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
- Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.
- Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Además de estas actividades fundamentales, existe un conjunto de actividades protectoras, que se aplican a lo largo de todo el proceso del software. Son las siguientes: [4]

- Seguimiento y control de proyecto de software.
- Revisiones técnicas formales.
- Garantía de calidad del software.

- Gestión de configuración del software.
- Preparación y producción de documentos.
- Gestión de reutilización.
- Mediciones.

En el presente trabajo se hará énfasis en 2 de las principales actividades del proceso de desarrollo de software, específicamente un software orientado a objetos, estas son:

El diseño: es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. [5]

Pressman cita a Gamma y plantea que en el diseño de software orientado a objetos se deben identificar los objetos pertinentes, clasificarlos dentro de las clases en la granularidad correcta, definir interfaces de clases y jerarquías de herencia y establecer relaciones clave entre ellos. El diseño debe ser específico al problema que se tiene entre manos, pero suficientemente general para adaptarse a problemas y requerimientos futuros.

La implementación: tomando como punto de partida el modelo del diseño, se procede a programar o implementar los diseños especificados en el mismo, es decir convertirlo en código preferentemente entendible y reutilizable.

1.3.2 ¿Qué es la programación orientada a objetos?

La Programación Orientada a Objetos es la forma de expresar un programa en términos de objetos que colaboran entre sí para realizar determinadas tareas, siendo estos representaciones de elementos de la vida real y que a su vez contienen toda la información que permiten su definición e identificación frente a otros objetos, además de contener funciones llamadas métodos que sirven de comunicación entre ellos.
Tiene características fundamentales que la identifican y son la base de las potencialidades que brinda a la hora de usarla para implementar cualquier sistema informático; ellas son: Herencia, Polimorfismo, y Encapsulamiento.

Es totalmente distinta a la programación estructurada, el programador trata de acercarse más al mundo real donde todo se vería como un objeto. Todo se refleja en términos de objetos, propiedades, métodos y otros elementos. A continuación se citarán algunos conceptos de programación orientada a objetos según algunos autores:

La programación orientada a objetos es mucho más que una frase comercial (aunque haya llegado a ser así por culpa de algunas personas), una nueva sintaxis o una nueva interfaz de programación de aplicación. La programación orientada a objetos es un conjunto completo de conceptos e ideas. Es una manera de pensar en el problema al que va dirigido un programa de ordenador y de afrontarlo de modo más intuitivo e incluso más productivo. [6]

Los principios de la programación al objeto es “modelar” y representar, a través de elementos de programación, objetos que existen en la realidad (tangible o intangibles).[7]

1.3.3 ¿Qué es una metodología de desarrollo de software?

Una metodología de desarrollo de software es un proceso que persigue un determinado objetivo. Este proceso debería ser capaz de servir como guía para todos los participantes (clientes, usuarios, desarrolladores y directores ejecutivos), evolucionar durante muchos años permitiendo limitar su alcance en un momento del tiempo dado a las realidades de la tecnología, herramientas, personas y patrones de organización.[8]

Se puede decir que, una metodología de desarrollo de software es un proceso que guía a los desarrolladores a los que les brinda métodos y herramientas, proporcionándoles una ayuda muy importante e indispensable para que el producto final posea las funcionalidades requeridas por el cliente y que cumpla con las necesidades del mismo y del usuario final, es una secuencia de actividades organizadas y bien pensadas que transforman los requisitos del cliente en el producto final.

1.3.4 ¿Qué es un IDE de programación?

IDE: Herramienta de soporte al proceso de desarrollo de software que integra las funciones básicas de edición de código, compilación y ejecución de programas. [9]

Los IDEs (Integrated Development Environment), tal y como su nombre indica, son entornos de desarrollo integrados.[10]

Resumiendo, un IDE de programación es simplemente un software, pero que a diferencia de otros es una herramienta para ayudar a los programadores a escribir programas informáticos, interpretando uno o varios lenguajes de programación, compilándolos y mostrándole al programador si hubo o no un error en el código. Estos programas contienen un compilador que interpreta uno o varios lenguajes de programación.

1.3.5 ¿Qué es un sistema gestor de base de datos?

Los Sistemas de Gestión de Base de Datos (SGBD) se pueden definir como el «conjunto de herramientas que suministra a todos (administrador, analistas, programadores, usuarios) los medios necesarios para describir, recuperar y manipular los datos almacenados en la base de datos, manteniendo la seguridad, integridad y confidencialidad de los mismos». [11]

1.4 Metodologías de Desarrollo de Software.

En el mundo existen varias metodologías que son usadas para realizar el desarrollo de un software, entre ellas se encuentran las metodologías tradicionales y las metodologías ágiles. Las metodologías tradicionales se centran en el control de los procesos estableciendo actividades involucradas como: los artefactos que se deban producir, las herramientas que se deben de usar, sin embargo las metodologías ágiles se centran en otras dimensiones como el factor humano o el producto de software dándose mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones cortas. [12]

Por estas razones la metodología ágil está cobrando gran importancia en el mundo actual de desarrollo del software producto de la necesidad de satisfacer al cliente y obtener como resultado lo que en verdad este desea, teniendo resultados positivos en una gran cantidad de proyectos con requisitos que están en constantes cambios, que es lo más común que sucede cuando se desarrolla un software en el que se tiene un cliente que complacer y además de que en el mundo actual se necesita de un desarrollo rápido. A continuación se citarán algunas de las metodologías existentes.

1. Metodología Cristal
2. Método de Desarrollo de Sistemas Dinámicos (DSDM)
3. Desarrollo de Software Adaptativo (ASD)

4. Proceso Unificado Racional (RUP)
5. Proceso Unificado Ágil (AUP)
6. Programación Extrema (XP)
7. SCRUM

De una forma u otra, todas han contribuido en el desarrollo de la industria del software, sirviéndoles de guía a los desarrolladores para un mejor desempeño y cumplimiento de los requisitos de un determinado producto. Pero como todo, ellas tienen su forma particular de ser útil, pues no plantean lo mismo y no se pueden utilizar en cualquier proyecto, debido a que no existe una metodología estándar para cualquier tipo de desarrollo en la industria del software, pues todo depende del cliente, o de la empresa, de las características de la misma, entre otros factores como la cantidad de personal, el tiempo en que se debe de entregar el producto y otros agentes que influyen en el desarrollo de un software.

Para el desarrollo del sistema en cuestión, se necesita una metodología liviana, configurable, adaptable, de poca documentación pero no nula, que responda a los intereses del cliente y el producto, que mantenga al cliente interesado, que le muestre productos funcionales en poco tiempo, y que permita que estos productos sean simples y fáciles de modificar, en aras de satisfacer los cambios recurrentes del jefe del producto. Estos requisitos determinan el uso de la corriente ágil y no la tradicional.

Existen diversas metodologías ágiles, que por su potencial son utilizadas comúnmente en grandes desarrollos de software a nivel mundial, a pesar de esto, no serán objeto de atención en el presente estudio, debido principalmente al tamaño del equipo de desarrollo que requieren para una mayor eficiencia; es el caso de la Metodología Cristal, la cual necesita como mínimo 3 integrantes, esto no coincide con el equipo en cuestión, pues se cuenta con solo un integrante, también influyen los documentos que generan, como es el caso del AUP, el cual es el RUP ágil, a pesar de fusionar sus flujos para lograr agilidad en su desempeño, en cuanto a los artefactos es bastante radical, pues para eliminar algunos de los que plantea su matriz (RUP) es necesario llevar una serie de actividades y cumplir algunos requisitos que deriven el uso o no de dichos artefactos, aspecto que afectaría el tiempo de desarrollo de la aplicación.

Por estas razones, se centrará el estudio comparativo en Scrum y XP ambas metodologías ágiles, seleccionadas por ser de las más utilizadas en la universidad, aportando resultados satisfactorios, además por ser las más cercanas a los intereses que se persiguen con el desarrollo en cuestión.

1.4.1 Scrum

Características de SCRUM

- Es una metodología de desarrollo ágil.
- Está pensada para equipos de desarrollos pequeños (no más de 8 personas)
- Se especifican pocos artefactos eliminando el “papeleo” innecesario y dedicando más tiempo a la implementación.
- Permite la entrega de un producto funcional al finalizar cada Sprint.
- Da la posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.
- Permite hacer una visualización del proyecto diaria.
- Tiene un alcance acotado y viable.
- Se aplica en equipos integrados y comprometidos con el proyecto y que se auto administran.
- No es una metodología de análisis, ni de diseño (aunque puede adaptarse para que lo sea) sino de gestión del trabajo.
- Puede ser aplicado a distintos modelos de calidad (como podría ser CMMI) puesto que estos te dicen qué tienes que hacer, es decir, te dicen que tienes que gestionar el proyecto, pero no te dicen cómo. Ahí es donde entra SCRUM como modelo de gestión del proyecto. [13]

Breve descripción de la metodología SCRUM.

“Scrum” es una metodología para la gestión de proyectos, expuesta por Hirotaka Takeuchi e Ikujiro Nonaka, en el artículo The New Product Development Game [Harvard Business Review Ene-Feb. 1986]. “El precepto básico de esta metodología es que el mercado competitivo de los productos tecnológicos, además de los conceptos básicos de calidad, coste y diferenciación, exige también rapidez y flexibilidad”. [14] Sin embargo, más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. Scrum ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. Igualmente permite seguir de forma clara el avance de las tareas a realizar, de forma que los "jefes" puedan ver día a día cómo progresa el trabajo. Scrum es una metodología ágil que no responde a ninguna moda, sino a una necesidad realmente demandada en el desarrollo del Software. No es ni la mejor metodología ni la única pero sí es la que está empujando muy fuerte por la facilidad de implantación y por su agilidad en cuanto a cambios y lo que propiamente aporta en comparación con otras. Por un lado,

Scrum evita la burocracia y la generación documental: No significa esto que no se deba o no se pueda documentar, si no que no se exige documentar nada para iniciar un proyecto, algo que en otras metodologías es impensable. La idea principal es la de ponerse a trabajar prácticamente desde el primer momento y empezar a sacar frutos de ese trabajo para que el cliente vaya viendo los avances y se quede satisfecho con lo que se está haciendo y cómo se está haciendo. Hay dos aspectos fundamentales a diferenciar, los actores y las acciones. Los actores de forma general, serán:

- Product Owner (Propietario del Producto): conoce y marca las prioridades del proyecto o producto.
- Scrum Master (Maestro del Scrum): es la persona que asegura el seguimiento de la metodología guiando las reuniones y ayudando al equipo ante cualquier problema que pueda aparecer. Su responsabilidad es entre otras, la de hacer de paraguas ante las presiones externas.
- Scrum Team (Equipo de Scrum): son las personas responsables de implementar la funcionalidad o funcionalidades elegidas por el Product Owner.
- Usuarios o Clientes: son los beneficiarios finales del producto, y son quienes viendo los progresos, pueden aportar ideas, sugerencias o necesidades.

Las acciones de Scrum forman parte de un ciclo iterativo repetitivo, por lo que el mecanismo y forma de trabajar que a continuación se indica, tiene como objetivo minimizar el esfuerzo y maximizar el rendimiento en el desarrollo. [15]

Las acciones fundamentales de Scrum son:

- Product Backlog (Pila de Tareas del Producto): corresponde con todas las tareas, funcionalidades o requerimientos a realizar. El Product Owner es la persona que se encarga de marcar las prioridades, y es al fin y al cabo, la persona que mantiene y actualiza dado el caso, la lista de tareas.
- Sprint Planning Meeting (Reunión de Planificación del Ciclo): es una reunión que tiene por objetivo, planificar el Sprint a partir del Product Backlog. El objetivo de esta reunión es la de mover las tareas del Product Backlog al Sprint Backlog. En esta reunión, suelen participar el Product Owner, el Scrum Master y el Scrum Team. De esta reunión surge el Sprint Goal, que es un pequeño documento o una breve descripción que indica lo que el Sprint intentará alcanzar.
- Sprint Backlog (Pila de Tareas del Sprint o Ciclo): corresponde con una o más tareas que provienen del Product Backlog de donde se saca una o más tareas que van a formar parte del

Sprint Backlog. Estas se deben acometer (recomendado) en unas 2 ó 4 semanas. Eso debe de ser marcado antes de iniciar el Sprint. Una norma fundamental es que mientras un Sprint Backlog se inicia, éste NO puede ser alterado o modificado. Hay que esperar a que concluya el Sprint Backlog para realizar la correspondiente modificación o alteración cuya tarea, formaría parte de otro Sprint Backlog.

- Daily Scrum Meeting (Reunión Diaria de Scrum): es una tarea iterativa que se realiza todos los días que dure el Sprint Backlog con el equipo de desarrollo o de trabajo. Se trata de una reunión operativa, informal y ágil, de un máximo de 30 minutos, en la que se le hace 3 preguntas a cada integrante del equipo:
    - Qué tareas ha realizado desde la última reunión (que he hecho).
    - Sobre qué va a trabajar en el día actual (que voy a hacer hoy).

Identificación de obstáculos o riesgos que impiden o pueden impedir el normal avance (que ayuda necesito). El Scrum Master, debe eliminar aquí cualquier obstáculo que encuentre.

"Cuando se ha finalizado un Sprint Backlog, se debe tener un entregable que se pueda mostrar y que evidencie los avances acometidos en el Sprint." [16]

Pero no acaban aquí las acciones o actividades:

- Sprint Review (Revisión del Sprint o Ciclo): se revisa en unas 2 horas como máximo el Sprint finalizado. Al llegar a este punto, se debe tener un entregable que el Cliente o el Usuario pueda valorar. En esta reunión, suelen asistir el Product Owner, el Scrum Master, el Scrum Team y personas que podrían estar involucradas en el proyecto. El Scrum Team es quién muestra los avances realizados en el Sprint.
- Sprint Retrospective (Retroalimentación del Ciclo): comienza luego de finalizar un Sprint Review. El Product Owner revisará con el equipo los objetivos marcados inicialmente en el Sprint Backlog concluido, se aplicarán los cambios y ajustes si son necesarios, y se marcarán los aspectos positivos (para repetirlos) y los aspectos negativos (para evitar que se repitan) del Sprint.

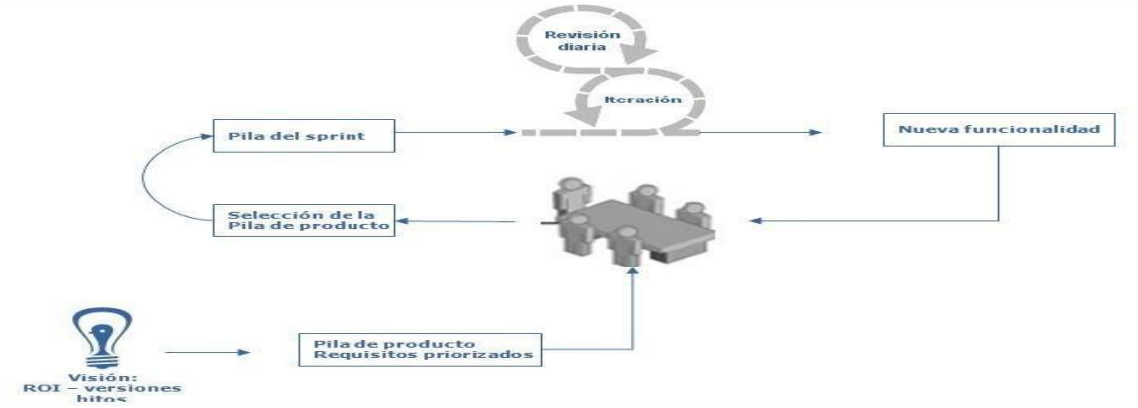


Figura 1.Ciclo de Scrum

Una vez terminado el Sprint si en el Product Backlog quedan tareas se planifica un descanso para el Scrum Team donde se resuelven posibles cuestiones vistas en el Sprint Retrospective. Comienza así una nueva fase en la que se comenzará nuevamente planificando un Sprint Planning Meeting.

1.4.2 Extreme Programing (XP)

Desarrollada por Kent Beck. Es una de las metodologías de desarrollo de software más exitosas en la actualidad, utilizadas para proyectos de corto plazo, entrega y equipo de desarrolladores pequeños. La misma consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

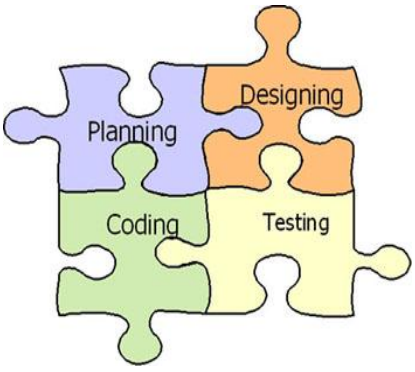


Figura 2.Metodología XP

Características de XP [17]

Está pensada para un grupo pequeño y muy integrado (máximo 12 personas) dónde la comunicación sea más factible que en grupos de desarrollo grandes. Sus características principales son:

- Comunicación: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.
- Simplicidad: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo y se tienen que desechar y partir de cero.
- Retroalimentación (Feedback): Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.
- Coraje: Se debe tener coraje o valentía para cumplir los tres puntos anteriores; Hay que tener valor para comunicarse con el cliente y enfatizar algunos puntos, a pesar de que esto pueda dar sensación de ignorancia por parte del programador, hay que tener coraje para mantener un diseño simple y no optar por el camino más fácil y por último hay que tener valor y confiar en que la realimentación sea efectiva.

1.4.3 Fundamentación de la selección de la metodología.

Como se ha podido apreciar las metodologías de desarrollo de software antes tratadas tienen su forma de guiar a los desarrolladores de software, teniendo sus métodos particulares. Además el uso de estas depende del producto que se quiera desarrollar, del usuario final, de las características de la empresa, entre otras circunstancias que se presenten. Para el desarrollo del trabajo se tendrá en cuenta utilizar como metodología a SCRUM, a pesar de las novedades de XP en cuanto a rapidez, comunicación y retroalimentación, en el equipo de desarrollo no se cuenta con un cliente como parte del mismo, factor fundamental en esta metodología ágil, todas las ventajas de la XP en cuestiones como la rapidez y énfasis en el producto, las porta SCRUM, es por esto y por el resto de sus características explicadas anteriormente que se selecciona para el progreso del presente trabajo.

1.5 Lenguaje de modelado

Para una mejor comunicación entre los clientes y los desarrolladores es preciso modelar todo el proceso de desarrollo de software, de esta forma la documentación y la validación de los requerimientos aumentan su nivel de calidad, así como todos los artefactos generados en el proceso.

Existen múltiples lenguajes para modelar en el ámbito del desarrollo de la solución informática, sin embargo algunos no serán objeto de estudio del presente trabajo, debido principalmente al factor de la capacitación, pues para ganar en tiempo es necesario utilizar lenguajes del dominio del equipo de desarrollo, pero existen otras causas del por qué no usar otros, es el caso del lenguaje de modelado i*, propuesto por Yu Eric en la década del 90, el cual permite expresar de forma clara y sencilla los objetivos de los actores que aparecen en sus modelos y las dependencias entre ellos, sin embargo contiene ambigüedades, contradicciones e incompletitudes pues no existe una definición única del lenguaje. También está el caso del lenguaje Orientado a Objetivos y Requisitos no Funcionales, forma parte de la Notación de los Requisitos de Usuarios (URN), que ha sido propuesta como estándar de la ITU-T (International Telecommunication Union –Telecommunication Standardization Sector), ofrece constructores para establecer relaciones con elementos externos al modelo, pero posee ambigüedades y contradicciones en la especificación sintáctica formal.

Todo esto arrojó la idea de estudiar lenguajes más cercanos a las necesidades del sistema, completos, robustos y probados universalmente, por sus grandes beneficios y ser los más comúnmente usados en la comunidad del software actualmente, se han escogido para su estudio comparativo el Lenguaje Unificado de Modelado (UML) y la Notación para el Modelado de Procesos del Negocio (BPMN).

1.5.1 BPMN

BPMN proporciona a los negocios la capacidad de entender sus procedimientos internos en una notación gráfica, facilitando a las organizaciones la habilidad para comunicar esos procedimientos de una manera estándar. Por tanto sus principales objetivos son:

- Proveer una notación que sea fácilmente entendida por todos los usuarios, desde el analista de negocio, el desarrollador técnico y hasta la propia gente del negocio.

- Crear un puente estandarizado para el vacío existente entre el diseño del proceso de negocio y su implementación.
- Asegurar que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común.
- BPMN es usado para comunicar una amplia variedad de información a una amplia variedad de audiencias.

Esta notación define un único diagrama, denominado Diagrama de Procesos de Negocio (BPD por sus siglas en inglés) que está basado en la técnica de gráficos de flujo, permitiendo la creación de modelos gráficos de las operaciones de procesos de negocio.
Una de las principales ventajas que posee BPMN es que originalmente fue concebida como una notación enfocada en procesos y no en objetos.

1.5.2 UML

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Es el lenguaje de modelado de sistemas de Software más conocido y utilizado en la actualidad. Como principales características de éste lenguaje que le han merecido su amplia popularidad en la industria, pueden mencionarse las siguientes:

- Posee concurrencia, es un lenguaje distribuido y adecuado a las necesidades de conectividad actual y futura.
- Reemplaza a decenas de notaciones utilizadas en otros lenguajes, agrupando las mejores técnicas de modelado.
- Modela estructuras complejas, las más importantes que soporta tienen su fundamento en las tecnologías orientadas a objeto.
- Emplea operaciones abstractas como guía para variaciones futuras, añadiendo variables si es necesario.

Las funciones principales de UML se pueden apreciar como siguen:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.

- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.5.3 Fundamentación de la selección del lenguaje de modelado.

UML, a lo largo de los años, se ha destacado por su utilidad para representar fenómenos del mundo real, razón por la cual, desde hace varios años se desarrollaron y popularizaron una serie de extensiones para el modelado. Entre los diagramas más útiles para este fin se encuentra: el diagrama de clases.

A pesar de que ambos lenguajes poseen características tentadoras el BPMN es más orientado a procesos y en este caso, los diagramas que se van a utilizar para documentar el sistema serán los diagramas de clases de diseño, es por eso que se aprovecharán los beneficios que aporta UML en este aspecto, pues es sabido que se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh.

UML es más expresivo, claro y uniforme para el diseño Orientado a Objetos, no garantiza el éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios, por esto precisamente el Lenguaje Unificado de Modelado es el escogido para el desarrollo de los artefactos del presente trabajo.

1.6 Herramienta de Modelado

Las Herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de *Software* Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática y documentación o detección de errores.

Para el estudio comparativo de las herramientas de modelado se tuvo en cuenta la necesidad de utilizar una que estuviera probada a nivel mundial, comúnmente usada, robusta, íntegra, de fácil aprendizaje, abundante documentación y que soportara a UML como lenguaje, aunque existen algunas bastantes cercanas, no se estudian debido a que poseen características no convenientes, es el caso de Dia; herramienta que permite modelar muchos diagramas, sin embargo es difícil de usar y no aprovecha todas las funcionalidades que ofrece UML. También está el caso del Enterprise Architec, el cual ofrece funcionalidades muy ventajosas para el modelado pero es vendido como un producto licenciado, limitante que unido al desconocimiento de su empleo, lo hacen no utilizable.

Es por esto que se han escogido 2 de las más usadas mundialmente, conocidas por el quipo de desarrollo, pues se estudian y emplean en la universidad, además cumplen con las cualidades requeridas. Estas son Rational Rose y Visual Paradigm.

1.6.1 Rational Rose

El Rational es una herramienta CASE desarrollada por Rational Corporation, basada en UML, permite crear los diferentes diagramas que se generan en el proceso de Ingeniería durante el desarrollo del software. Presenta un gran número de estereotipos que permiten el proceso de modelación del software. Dicha herramienta es capaz de generar el código fuente de las clases definidas en el flujo de trabajo de diseño. Proporciona mecanismos para realizar Ingeniería Directa e Inversa, posibilita la construcción de un modelo de casos de usos, identifica los objetos y representa cómo interactúan con los diagramas de secuencia y colaboración, así como otras operaciones. Además el Rational Rose organiza sus diagramas en vistas: la vista de casos de uso, la vista lógica, la vista de componentes y la vista de despliegue. EL uso de estas vistas facilita la organización del trabajo para una mejor comprensión del mismo.

1.6.2 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es muy popular por sus disimiles características, producto de calidad, que soporta aplicaciones Web, es muy fácil de instalar y actualizar. Permite la generación de

código para varios lenguajes. Su diseño está centrado en casos de uso y enfocado al negocio generando un software de mayor calidad. Presenta capacidades de ingeniería directa e inversa y disponibilidad en múltiples plataformas.

Ventajas del Visual Paradigm:

- Soporta UML versión 2.1.
- Interoperabilidad con modelos UML2 a través de XMI.
- Permite realizar el Diagramas de flujo de datos.
- Ofrece un Generador de informes para generación de documentación.
- Importación y exportación de ficheros XMI.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE.

1.6.3 Fundamentación de la selección de la herramienta de modelado.

Visual Paradigm es la herramienta seleccionada para el trabajo de este software, por todas sus comodidades de trabajo y comprensión de la misma, además de la calidad del producto que resulta, la facilidad de su utilización para hacer los diagramas que se necesiten, su documentación, también es sabido que trabajar con software libre es uno de los objetivos del país para la producción de software cubano y la utilización de esta herramienta lo permite debido a su capacidad de ser multiplataforma.

Existen varias versiones del Visual Paradigm, para darle solución a la problemática existente se trabajará con la Enterprise, la Universidad de las Ciencias Informáticas posee la licencia de esta última, no siendo así con la de Rational Rose Enterprise Suite.

1.7 Paradigmas de Programación.

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión y alta rentabilidad. Es una colección de modelos conceptuales que juntos modelan el proceso de diseño y

determinan, al final, la estructura de un programa. Los tipos o técnicas de programación son bastante variados.

Entre los paradigmas de programación más reconocidos y utilizados se encuentran:

- Programación Orientada a Objetos (POO).
- Programación Orientada a Aspectos (POA).
- Programación Dirigida por Eventos.

Para aprovechar los beneficios que aporta la Programación Orientada a Objetos en cuanto a:

- **Abstracción**: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.
- **Encapsulamiento**: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.
- **Principio de ocultación**: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas.
- **Polimorfismo**: Las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado.
- **Herencia**: Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir y extender su comportamiento sin tener que volver a implementarlo.

- **Recolección de basura**: Es la técnica por la cual el ambiente de objetos se encarga de destruir automáticamente, y por tanto desasignar de la memoria, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando.

Se ha escogido este paradigma para el desarrollo de la solución informática, el resto de los paradigmas ralentizarían el progreso del sistema, debido a que no son conocidos por el equipo de desarrollo, pues el paradigma estudiado en los años de la carrera ha sido la POO, utilizar otros implicaría, dedicar tiempo al estudio y dominio de los mismos, aspecto que afectaría la duración y entrega del producto, además algunos ofrecen ventajas no convenientes para el sistema en cuestión, en cuanto a la implementación de cada funcionalidad del sistema de manera modular e independiente; como es el caso de la POA, para proyectos pequeños, logra un efecto contrario al requerido, pues provoca que el diseño de las clases se torne engorroso, lo que traería consigo un aumento de la complejidad del diseño y su mantenimiento, lo cual provocaría, más esfuerzo en la realización de las pruebas, otra funcionalidad que no se necesita en el sistema en cuestión, es el poder del usuario o el ente que interactúe con el sistema de dirigir su flujo; es el caso de la Programación Dirigida por Eventos, esto implicaría que el flujo de ejecución del software escape del control del programador, lo que traería consigo errores fatales en tiempo de ejecución. Todo esto inclina la balanza en aras de aprovechar las potencialidades que ofrece la POO, en el sistema en cuestión.

1.8 Lenguajes de Programación.

Los lenguajes de programación y los IDE de desarrollo tienen suma importancia para cualquier proyecto de software. ¿Se imaginan que sería de los desarrolladores sin la existencia de un lenguaje de programación y de un IDE que lo ayude en la tarea de corregir código? Sería fatal, pues con estos se han creado innumerables programas y herramientas que han ayudado al hombre a controlar de una manera más sencilla a los ordenadores, así como de realizar múltiples tareas y actividades.

Hoy en día existen muchos lenguajes a los que se puede acudir para la realización de cualquier software, pero como unos son más fáciles de utilizar que otros debido a la eliminación de código inseguro, el

cumplimiento con el paradigma de la programación orientada a objetos, sentencia más amigable y sencilla, la facilidad que proporcionan para su aprendizaje, la propiedad de que sean multiplataforma, entre otras disímiles características que permiten a los programadores elegirlos, entre ellos se encuentran los lenguajes orientados a objetos, se pueden mencionar:

- C++
- C#
- VB.NET
- PHP
- Delphi
- Java

El estudio comparativo del lenguaje de programación estuvo condicionado por el conocimiento del equipo de desarrollo, debido a la premura de la aplicación y la necesidad de su uso de manera urgente y precisa, es por esto que a pesar de existir diversos ejemplares, con características envidiables, se centra el estudio en Java y C#, con el objetivo de ganar en tiempo y aprovechar la facilidad de aprendizaje y novedades que ambos aportan.

1.8.1 Java

Java es un lenguaje de programación creado por SUN Microsystems muy parecido al estilo de programación del lenguaje “C” y basado en lo que se llama programación orientada a objeto.[18]
Entre las principales características de Java se pueden citar:

1. Sintaxis similar a la de C++, lenguaje que se imparte en nuestra universidad. Aunque se simplifican algunas características del lenguaje como:

   La sobrecarga de operadores, la herencia múltiple, el paso por referencia de parámetros, la gestión de punteros, la liberación de memoria y las instrucciones de pre compilación.

2. Soporte homogéneo a la Programación Orientada a Objetos. A diferencia de C++, que puede considerarse un lenguaje multiparadigma, Java está diseñado específicamente para utilizar el paradigma de orientación a objetos.

3. Independencia de la plataforma. En Java se pretende que con una sola compilación se obtenga código ejecutable en diferentes Sistemas Operativos e incluso sobre diferente hardware.[19]

La compañía Sun describe el lenguaje Java como “simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico”. [20]

Orientado a objetos: Desde un principio fue diseñado orientado a objetos que agrupan en estructuras encapsuladas tanto sus datos como los métodos que manipulan esos datos.

Distribuido: Proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez: Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, debido a que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real.

Robusto: Fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores como la aritmética de punteros, porque ya en este lenguaje se ha prescindido por completo de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria. Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

Multihilo: En la actualidad muchas aplicaciones realizan varias operaciones al mismo tiempo, este lenguaje posee la característica de que los programadores puedan explotar esto, pues java permite la programación multihilo o multiproceso en el cual se crean múltiples procesos que se encargan de realizar cálculos y operaciones distintas en el mismo instante.

1.8.2 C Sharp

C#, como parte de la plataforma.NET, está normalizado por ECMA desde diciembre de 2001 (ECMA-334).Este lenguaje posee una estructuración y una sintaxis muy parecida a la de C++ o Java. También es un lenguaje orientado a objetos simple, elegante y con seguridad en el tratamiento de tipos, que permite a los programadores de aplicaciones empresariales crear una gran variedad de aplicaciones.

En este lenguaje es posible interaccionar con otros lenguajes, entre plataformas distintas, y con datos heredados, posee compatibilidad con XML para interacción con componentes basados en tecnología Web y capacidad de control de versiones para facilitar la administración y la implementación.

Facilidad de uso: el ambiente de trabajo es muy cómodo ya que tiene un ambiente amigable y clásico de las aplicaciones de Windows. En cuanto a la forma de programar, será fácil de usar para quien está familiarizado con C++, ya que su estructuración básica es muy similar, sin embargo C# ahorra muchos pasos “tediosos” de otros lenguajes como la creación de funciones complejas desde cero y declaración de variables globales.

Programación orientada a objetos: C# presenta las características necesarias para considerarlo como un lenguaje orientado a objetos, una de las mejoras que presenta este lenguaje con respecto a este tipo de programación es que para evitar confusiones no existen variables o funciones globales, sino que se definen dentro de los tipos de datos. En cuanto a la herencia, esta solo puede ser herencia simple, con lo cual se evitan confusiones que si fuera herencia múltiple.

Administración de memoria: C# tiene la característica de inicializar los datos o variables declaradas en el programa, además de que también de forma automática libera la memoria cuando el mismo programa lo cree conveniente. Es decir tiene constructores y destructores, y estos actúan automáticamente a menos que se manipulen desde el código.

Seguridad en el manejo de datos: C# tiene la característica de estar comprobando que efectivamente los tipos de datos que se estén manejando correspondan a los validados para las funciones que han sido creadas; así también vigila que no se produzcan errores en operaciones matemáticas, además de que también impide el uso de variables que no han sido inicializadas. Todo esto permite que no se produzcan errores en el momento de la ejecución.

Sistema de tipos unificado: todos los tipos de datos que se definan siempre se derivarán, incluso de forma implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase.

La ventaja de que todos los tipos se deriven de una clase común es que facilita el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.
Uso de operadores: este lenguaje permite de forma automática la manera en que pueden trabajar los operadores, ya sea de tipo lógico, aritmético, etc. Es decir dependiendo del contexto de donde se encuentre el operador, el programa detecta que tipo de uso debe tener el operador.

1.8.3 Fundamentación de la selección del lenguaje.

Para la elección de uno de los lenguajes que se tomaron como referencia, se tuvo en cuenta principalmente la facilidad de aprendizaje, debido a la premura de la aplicación y la capacitación del equipo de desarrollo. Por esta razón se ha elegido al C# como el lenguaje idóneo para la implementación del software. Este posee algunas supremacías sobre el java en cuanto al rendimiento, es por lo general, mucho mejor, el CIL (el lenguaje intermedio de .NET) está estandarizado, mientras que los bytecodes de java no lo están, soporta más tipos primitivos (value types), incluyendo tipos numéricos sin signo, posee indizadores que permiten acceder a cualquier objeto como si se tratase de un array, compilación condicional, aplicaciones multi-hilo simplificadas, soporta la sobrecarga de operadores, que aunque pueden complicar el desarrollo son opcionales y algunas veces muy útiles y permite el uso (limitado) de punteros cuando realmente se necesiten, como al acceder a librerías nativas que no se ejecuten sobre la máquina virtual.
Cualquiera sea el lenguaje que se utilice, es necesario contar con medios para que los desarrolladores puedan editar el código, compilar, y ejecutar programas con mayor facilidad, es ahí donde radica la importancia de un IDE de desarrollo. C# es un lenguaje que posee una larga oferta de IDEs entre los que se encuentran Visual Studio, CSharpDevelop y Monodevelop.

1.9 IDEs para C Sharp(C#).

La elección de un IDE no es simple y sobre todo para un lenguaje como C# donde existen muchas variedades como se había explicado anteriormente. Muchos son los parámetros a la hora de elegirlos, debido a que algunos solo funcionan en unas plataformas y otras no, su facilidad para desarrollar, la interfaz gráfica, que sea libre o no, entre otras.

Al igual que en el lenguaje de programación, el estudio comparativo de los IDEs para C# estuvo condicionado entre otras cosas por el conocimiento del equipo de desarrollo, factor que influye en el retraso de la entrega del producto, es por eso que a pesar de que existen diversos IDEs con características tentadoras, se hace énfasis en el estudio de Visual Studio para aprovechar su curva de aprendizaje y las facilidades y potencialidades que porta para un desarrollo ágil, además se tiene en cuenta el Monodevelop, pues servirá para en un futuro migrar el software a entornos libres, aspecto que se recomienda en el presente trabajo.

1.9.1 Monodevelop.

El proyecto Monodevelop es un IDE libre y gratuito, diseñado para la programación de C# y algunos otros lenguajes de .NET, este proyecto es liderado por Miguel de Icaza y otros programadores, la versión 1.0 fue liberada el 14 de Marzo del 2008.

Esta aplicación de open source incluye nuevas características tanto básicas como avanzadas, como herramientas para la administración de proyectos, interfaz gráfica de diseño, sistema de pruebas, integración del control en la versión, extensibilidad con un add-in para los sistemas. También se ha agregado la función que auto-completa código, notificación de errores, navegación por el código, funcionalidad para identar el código en C#, VB.NET, y C/C++.

Una de las mejores características, es que puede realizar un debug gráfico, pero aún no se encuentra completamente desarrollada hasta la versión 1.1. Los desarrolladores planean corregir errores y aumentar sus capacidades cada seis meses, pero a comparación de otros editores de Mono, este es el mejor debido a su interfaz gráfica.

1.9.2 Visual Studio 2005.

Visual Studio 2005 se empezó a comercializar a través de Internet a partir del 4 de Octubre de 2005 y llegó a los comercios a finales del mes de Octubre en inglés. En castellano no salió hasta el 4 de Febrero de 2006. Microsoft eliminó .NET, pero eso no indica que se alejara de la plataforma .NET, de la cual se incluyó la versión 2.0.

La actualización más importante que recibieron los lenguajes de programación fue la inclusión de tipos genéricos, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos

más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

Utilizando XML, una tecnología estándar del sector para la descripción de datos, se pueden crear aplicaciones de alto rendimiento controladas por datos. Se puede utilizar herramientas de ADO.NET integradas y orientadas a una gran variedad de bases de datos, incluidos SQL Server, Oracle o cualquier otro origen XML. La compatibilidad intrínseca de ADO.NET con XML permite compartir datos entre diferentes plataformas informáticas. Además, Visual Studio .NET incluye Microsoft Data Engine (MSDE), una base de datos totalmente compatible con SQL Server que proporciona una base de datos de desarrollo viable y compatible en modo nativo con XML para ofrecer la máxima interoperabilidad.

Los formularios Windows Forms muy intuitivos y eficaces cuando se produzcan código en cualquier lenguaje .NET, incluidos Visual Basic .NET y Visual C# .NET. Con la herencia visual, se pueden simplificar enormemente la creación de aplicaciones para Windows, mediante la centralización en formularios principales de la interfaz de usuario y la lógica común de toda la solución. La delimitación y el acoplamiento de controles permiten proporcionar formularios cuyo tamaño puede cambiarse de manera automática sin código, mientras que el editor de menús permite crear visualmente menús directamente en el diseñador de formularios.

Posee las ventajas de un cuadro de herramientas, un depurador y una ventana de tareas comunes, lo que reduce enormemente la curva de aprendizaje del desarrollador y garantiza que siempre elegirá el lenguaje más apropiado para su tarea y conocimientos. Con la comprobación automática de errores de sintaxis, Visual Studio 2005 informa cuando el código no es correcto y proporciona una visión inmediata de las jerarquías de clases y API. Con el Explorador de soluciones, se pueden reutilizar fácilmente el código de unos proyectos a otros e, incluso, generar soluciones multilenguaje que satisfagan sus necesidades empresariales con mayor eficacia. Además, gracias al IDE totalmente ampliable, se pueden disfrutar de las ventajas de una comunidad de proveedores de complementos y componentes de otros fabricantes que ayuda a personalizar y ampliar aún más el entorno para ajustarlo a diversas necesidades.

Un aspecto muy importante de este IDE es la curva de aprendizaje, es decir el grado de éxito obtenido durante el aprendizaje de los contenidos de la herramienta en el transcurso del tiempo. Es el tiempo necesario para desarrollar un proyecto con resultados razonables.

El Visual Studio 2005 posee una curva de aprendizaje suave o fácil, debido a que saca partido a todas las novedades de la plataforma y sus diferentes apartados y además amplía sus capacidades para hacer más fácil la vida a los programadores. El editor de código es proactivo y ofrece al programador ayuda continuamente mediante un mejorado Intellisense o el uso de "smart tags" como en Office, que ofrece información, resalta errores de sintaxis e incluso ofrece tareas concretas a realizar según el contexto (por ejemplo corrección automática de errores). También se indica mediante colores qué partes de un archivo se ha modificado desde que se cargó, distinguiendo entre cuáles de éstas están ya guardadas a disco y cuáles no.

Existen muchos más asistentes y diseñadores. Por ejemplo, el diseñador de XML permite validar los datos contra esquemas XSD. El diseñador de conjuntos de datos con tipo es intuitivo, con nuevos asistentes y con la incorporación de los TableAdapters que evitan tener que escribir componentes propios separados para rellenarlos de datos.

Posee grandes capacidades de creación de aplicaciones enlazadas a datos, se pueden crear potentes interfaces de interacción con los datos, con navegación y visualización. El depurador integrado otorga unas características muy ampliadas, como la posibilidad de editar y continuar, mucha más información sobre los errores incluso con opciones de resolución automática de problemas, los útiles visualizadores, o la opción para depurar sólo cierto código de interés.

Existen multitud de pequeños detalles que mejoran la experiencia del programador por ejemplo las opciones de refactorización de código incluidas. Todos estos aspectos alivian la curva de aprendizaje de este IDE, siendo una de las más suaves del ambiente desarrollador.

1.9.3 Fundamentación de la selección del IDE.

Debido a la necesidad urgente de la aplicación a desarrollar, el conocimiento del equipo de desarrollo, la curva de aprendizaje y las amplias funcionalidades y ventajas que ofrece el Visual Studio para el rápido desarrollo de aplicaciones escritorio, su gran velocidad, facilidad de uso y de aprendizaje se ha escogido este IDE, a pesar de no pertenecer a la revolución del software libre, inconveniente para el cual se estudiará la posibilidad de migrar la aplicación en fases posteriores para el otro IDE estudiado o para algún otro que use plataformas libres. Es válido aclarar que el proyecto que avala esta aplicación tiene comprada las licencias requeridas, no obstante el software tendrá un uso interno en el equipo de desarrollo.

1.10 Patrones de diseño.

A lo largo de los años los desarrolladores se han encontrado con problemas que se repiten reiteradamente en el desarrollo de un software, es por estas razones que los mismos han creado los patrones para usarlos en determinadas situaciones que se presenten. Los patrones han servido de ayuda y hoy en día se aplican mucho en distintos proyectos. En el diseño existen patrones los cuales son: “Soluciones ya probadas y eficaces de los problemas de diseño que pueden expresarse como un conjunto de principios”. [21]

Una de las principales razones de las ciencias de la computación en cuanto a los patrones de diseño, es la necesidad de obtener soluciones elegantes, simples y reutilizables.

Existen una gran cantidad de patrones y cada uno hace un pequeño aporte para solucionar determinados problemas. Para el desarrollo del sistema se tuvo en cuenta el uso de los siguientes patrones.

Patrones GRASP (General Responsability Asignment Software Paterns (Patrones Generales de Software para Asignar Responsabilidades))

Describen principios fundamentales de diseño de objetos para la asignación de responsabilidades las que están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. [21]

Patrones GOF (Gang of Four (Banda de los Cuatro))

Se dividen en patrones de creación, estructurales y de comportamiento. Estos patrones fueron publicados por Gamma, Helm, Jonson y Vlissides en su libro “Design Patterns: Elements of Reusable Object-Oriented Software". Los de creación son los que se encargan de crear los objetos, mientras que los estructurales se dedican al planteamiento de las relaciones entre las clases combinándolas para la formación de estructuras, finalmente los de comportamiento se dedican a la interacción y cooperación entre las clases, estudiando las relaciones entre los mensajes que ocurren entre los objetos. [22]

1.11 Patrones arquitectónicos.

La arquitectura del software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. Su objetivo principal es aportar elementos que ayuden a la toma de decisiones y al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto.

Esta no solo está relacionada con la estructura y el comportamiento del sistema, sino que está también dictada por el uso, la funcionalidad, el rendimiento, la flexibilidad, la reutilización, la facilidad de comprensión, las restricciones y compromisos económicos y tecnológicos, y la estética.

En la actualidad existen cientos de definiciones de Arquitectura de Software pero desde el 2001 hay una definición que ha sido la más aceptada.

Arquitectura – IEEE 1471-2000. “La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución”. Adoptada por Microsoft en estrategia arquitectónica.

Una buena práctica referente a la arquitectura de software es el uso de los estilos y patrones arquitectónicos, es por eso que en este apartado se hará un especial énfasis en estos últimos, por ser los utilizados en el presente trabajo.

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos.

Frank Buschmann define los **patrones arquitectónicos** como sigue: “Un problema particular recurrente del diseño que surge en un contexto de diseño específico y presenta esquema genérico bien probado para la solución de dicho problema. Esta solución es especificada describiendo sus componentes constitutivos las responsabilidades de estos componentes, sus relaciones y la forman en que colaboran” [23].

1.12 Gestor de base de datos.

La información, sea de la naturaleza que sea, y la posibilidad de obtener el máximo control sobre ella ha sido uno de los principales objetivos del ser humano desde hace ya siglos. De hecho, el poder de gestionar grandes cantidades de datos se ha constituido, especialmente a través de las última décadas, en uno de los factores más significativos en lo que respecta al nivel de desarrollo del hombre. Hoy en día es difícil encontrar un solo lugar en el mundo civilizado en el que no exista un completo control sobre todo

lo que sea “registrable”, ya sea datos personales de la población, información estadística de cualquier índole o, incluso, datos antes tan difíciles de registrar de forma exhaustiva como son el material literario. Es sabido que en el ámbito empresarial, llegó un momento en el que la información de negocios alcanzó un volumen superior al que la capacidad humana puede procesar, en cuestiones de búsqueda, análisis y cruces entre los datos. A partir de ese momento, el ordenador y el proceso computarizado tomaron el testigo, y mediante las Bases de Datos se comenzó a almacenar, procesar y gestionar los datos de las empresas más importantes.

La propia evolución de la Informática, en todos sus aspectos, fue abaratando costes y simplificando procesos. Las herramientas de administración de Bases de Datos se desarrollaron incesantemente (aún hoy día siguen avanzando) hasta convertirse en poderosos motores capaces de manejar enormes cantidades de información en pocos segundos. Para superar los límites de los sistemas de ficheros surgieron los denominados Sistemas Gestores de Bases de Datos. Estas aplicaciones daban cabida a múltiples sistemas reales de datos (múltiples bases de datos). Esto se debe a que no solo almacenaban los datos en sí, sino que además almacenaban la definición lógica de la base de datos (esquemas y subesquemas), así como la definición de los usuarios y sus respectivos permisos de acceso.

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de los datos. A continuación se citarán algunos de los sistemas de gestión de base de datos existentes:

- Fox Pro.
- IBM DB2 Universal Database (DB2 UDB).
- IBM Informix.
- MAGIC.
- Microsoft Access.
- Microsoft SQL Server.
- Oracle.
- PostgreSQL.
- MySQL.

El sistema a desarrollar necesita un gestor potente, con gran capacidad de almacenamiento, alto rendimiento, además de independencia, seguridad y consistencia en los datos. Existen grandes gestores que cumplen con estos requisitos, pero no serán objeto de estudio del presente trabajo por tener algunas importantes limitantes, es el caso de Oracle, uno de los más reconocidos mundialmente, pues tiene sobradas cualidades que lo hacen ser el preferido, pues posee soporte de transacciones, gran escalabilidad, estabilidad y es multiplataforma, sin embargo es propietario, lo mismo pasa con SQL, ambos son software privativos y limitarían la independencia tecnológica que se persigue.

Es por eso que al escoger entre los gestores que pertenecen a la Revolución de Software Libre, los que más resaltan por sus potencialidades, ventajas y teniendo en cuenta las necesidades de la aplicación vienen a ser PostgreSQL y MySQL. De esta forma se contribuye a la independencia tecnológica del sistema en cuestión, aspecto que por razones de tiempo y capacitación del equipo de desarrollo, no es completo, pero en futuras implementaciones se logrará en su totalidad. Seguidamente se estudiarán de modo comparativo para establecer sus diferencias.

1.12.1 MySQL

MySQL es un SGBD multiplataforma, es un software de fuente abierta lo que significa que cualquier persona puede usarlo y modificarlo para satisfacer sus necesidades, puede ser utilizado gratuitamente. Se encuentra bajo la Licencia Pública General de GNU (GPL). Por su sencillez y sus características es usado por muchas personas, consume muy pocos recursos, se usa tanto en aplicaciones sencillas como complejas. Es utilizado en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/).Posee un alto rendimiento, debido al bajo consumo que emplea. Gran facilidad de configuración e instalación. Tiene una probabilidad muy reducida de corromper los datos, incluso en los casos en los que los errores no se produzcan en el propio gestor, sino en el sistema en el que está.

1.12.2 PostgreSQL

Es un magnífico gestor de bases de datos, es multiplataforma. Se encuentra bajo la licencia Distribución de Software Berkeley (BSD). Permite una fácil gestión de los usuarios y de las bases de datos que contenga el sistema. Sirve de soporte al protocolo de comunicación encriptado por SSL. El número de base de datos que puede contener es ilimitado. Posee implementación del estándar SQL92/SQL99.

Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, imágenes, videos, datos sobre redes (MAC, IP), cadenas de bits, entre otros. También permite la creación de tipos propios. Incorpora una estructura de datos array. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, entre otros. Permite la declaración de funciones propias, así como la definición de disparadores. Soporta el uso de índices, reglas y vistas. Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales. Al utilizar PostgreSQL se disipa cualquier dificultad a la hora de seleccionar el tipo de tabla que se va a utilizar debido a que cuenta con un único mecanismo de almacenamiento.

1.12.3 Fundamentación de la selección del gestor.

A pesar de ser ambos gestores muy potentes, a partir de las características que ofrecen se decidió escoger a PostgreSQL como gestor, éste sistema gestor de BD de código abierto, posee características como el manejo de la integridad referencial, es decir, diferencia el trato a las llaves foráneas, no contempladas en MySQL y necesarias para el desarrollo de la solución informática de este trabajo. Es válido aclarar que aunque MySQL posee ventajas en cuanto al consumo y el rendimiento en pequeñas base de datos, PostqreSQL mantiene el mismo rendimiento en base de datos pequeñas como grandes, factor digno de admirar, es un fuerte gestor, comparable con grandes gestores comerciales como Oracle. Por todas sus potencialidades y en base a las necesidades funcionales de la aplicación en cuestión, es el SGBD escogido.

1.13 Conclusiones del Capítulo.

Con la investigación de este capítulo se logró cumplir el objetivo por el cual fue desarrollado, el estudio de los SGI, la factibilidad de los mismos, su vinculación a los procesos de diagnósticos a entidades legales, las principales terminologías referentes a estas últimas, la importancia de un proceso de diagnóstico en el proceso de desarrollo de software, todo esto contribuyó a establecer el fundamento teórico del presente trabajo. Además se hizo un estudio conceptual y comparativo de muchas de las metodologías, tecnologías y herramientas existentes en el mundo informático actual, dentro de las cuales, por convenios con el cliente y características del sistema a automatizar, se eligieron los candidatos más apropiados para guiar e implementar el proceso productivo en cuestión.

CAPÍTULO 2.Descripción de la Propuesta de Solución.

2.1 Introducción

En el presente capítulo se hace una descripción de la propuesta de solución de este trabajo basado en la descripción de los procesos de negocio propuestos, se explican los principales artefactos de la metodología empleada y que debe tener el sistema propuesto.

Igualmente, se especifican los aportes prácticos y vías de solución de la situación problemática, se describe el sistema y su distribución, se planifica el Product Backlog y los distintos Sprint.

2.2 Descripción de los procesos del negocio.

Este trabajo pretende modelar los procesos que se llevan a cabo al diagnosticar entidades legales con el objetivo de contribuir al desarrollo de software de sistemas legales. En este apartado se verán los roles involucrados, los documentos empleados y los procesos en cuestión.

Roles Involucrados

Para realizar un buen Diagnóstico es necesario definir un conjunto de roles especializados que deben estar presentes durante este proceso. Estos roles deben dividirse en dos grupos; roles de la parte Cliente, se refiere a las personas que laboran en las entidades que serán diagnosticadas y roles del Equipo de Diagnóstico, se refiere al personal que va a realizar el diagnóstico. En esta sección se identifican cuales son los roles por cada una de las partes involucradas en el proceso y también se especifican las responsabilidades que deben cumplir los mismos.

Roles de la parte cliente:

Nombre	Descripción	Responsabilidades

Representante de la entidad.	Es la persona que representa al equipo por la parte cliente.	Estar presente a la hora del diagnóstico de su entidad. Acompañar al equipo de diagnóstico durante el recorrido por la instalación.
Funcionario por área.	Es la persona especializada en uno o varios procesos.	Estar presente a la hora del recorrido por el área de desarrollo del proceso. Colaborar con las respuestas a cualquier duda que tenga el equipo de diagnóstico referente al proceso que desarrolla.

Tabla 1. Roles de la parte cliente.

Figura 3. Roles de la Parte Cliente.

Roles del Equipo de Diagnóstico.

Nombre	Descripción	Responsabilidades
Jefe del Diagnóstico	Es un especialista encargado de dirigir al equipo técnico y tomar las decisiones referentes al diagnóstico.	Gestionar los recursos humanos necesarios para el recorrido por las entidades. Garantizar el cumplimiento de las actividades involucradas en cada uno de los procesos del diagnóstico.

		Tomar las decisiones necesarias. Dirigir al equipo de diagnóstico y definir qué responsabilidades tienen cada uno dentro del equipo.
Informático.	Es un especialista en el Diagnóstico que además de ser el jefe del equipo en la entidad visitada, debe recoger la información relacionada con los procesos y los servicios que brinda la entidad.	Presentar y exponer los motivos de la visita, caracterizar de manera general la entidad. Identificar los procesos y las operaciones de los servicios. Concluir la visita.
Especialista en áreas a diagnosticar.	Es un especialista en la parte eléctrica y de redes o en la infraestructura de la entidad.	Evaluar el estado de las Instalaciones eléctricas. Evaluar el estado de la Red Informática. Evaluar la infraestructura de la entidad.
Especialista de Cambio Transformacional	Es un especialista en la identificación de procesos.	Identificar los procesos que se realizan en la entidad, determinando el orden, los responsables y los elementos de entrada y salida de cada uno.

Tabla 2. Roles del equipo de Diagnóstico

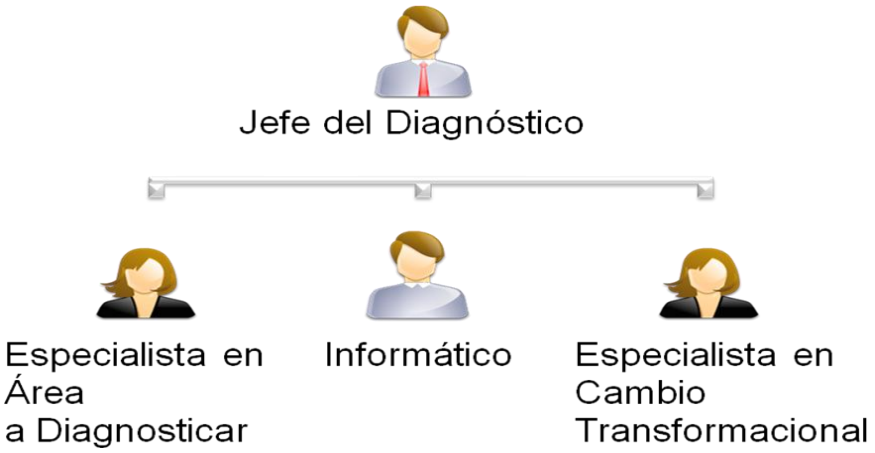


Figura 4.Roles del Equipo de Diagnóstico.

Definición de la Documentación

En cada una de las actividades que se realizan durante el proceso de Diagnóstico se generan un conjunto de artefactos que son de vital importancia para su realización.

En esta sección se describen los documentos a entregarse en el proceso de Diagnóstico. De cada uno de los artefactos que se generan durante este proceso se especifica: nombre, descripción y rol responsable de su realización.

Es preciso conocer en qué consiste un entregable: son documentos que no condicionan la continuidad de la ejecución del proyecto y serán entregados al finalizar el diagnóstico, previo a esto, los artefactos podrán ser revisados por el cliente durante la visita por la entidad.

Nombre	Descripción	Responsable
Informe General de la situación de la entidad.	Diagnóstico de la entidad, incluyendo la situación eléctrica, la de la red informática, situación tecnológica de la oficina, así como los procesos que tienen lugar en la misma. Documento que recoge las observaciones obtenidas por el equipo de diagnóstico, incluyendo el análisis de posibles riesgos en cuanto a los aspectos mencionados.	Jefe de Equipo.
Expediente por Entidad.	Caracterización de cada entidad, incluyendo un resumen de la información recogida.	Jefe del Diagnóstico.
Resumen Ejecutivo de las Entidades por Estado.	Resumen de la situación de cada estado en cuanto a las	Jefe del Diagnóstico.

	características de sus entidades.	

Tabla 3.Documentación

Estrategia del Diagnóstico.

El objetivo que se persigue en el proceso de Diagnóstico es recolectar información referente a los procesos que se rigen en la entidad, las instalaciones eléctricas, la red informática existente, el persona, la infraestructura de la entidad en general, de manera que esto contribuya a gestionar futuros riesgos en venideros procesos de Despliegue y de una forma u otra a modernizar e informatizar la entidad diagnosticada.

En esta sección se describen los procesos relacionados con el Diagnóstico en un nivel detallado declarando sus objetivos, responsables, involucrados, documentos de entrada y salida.

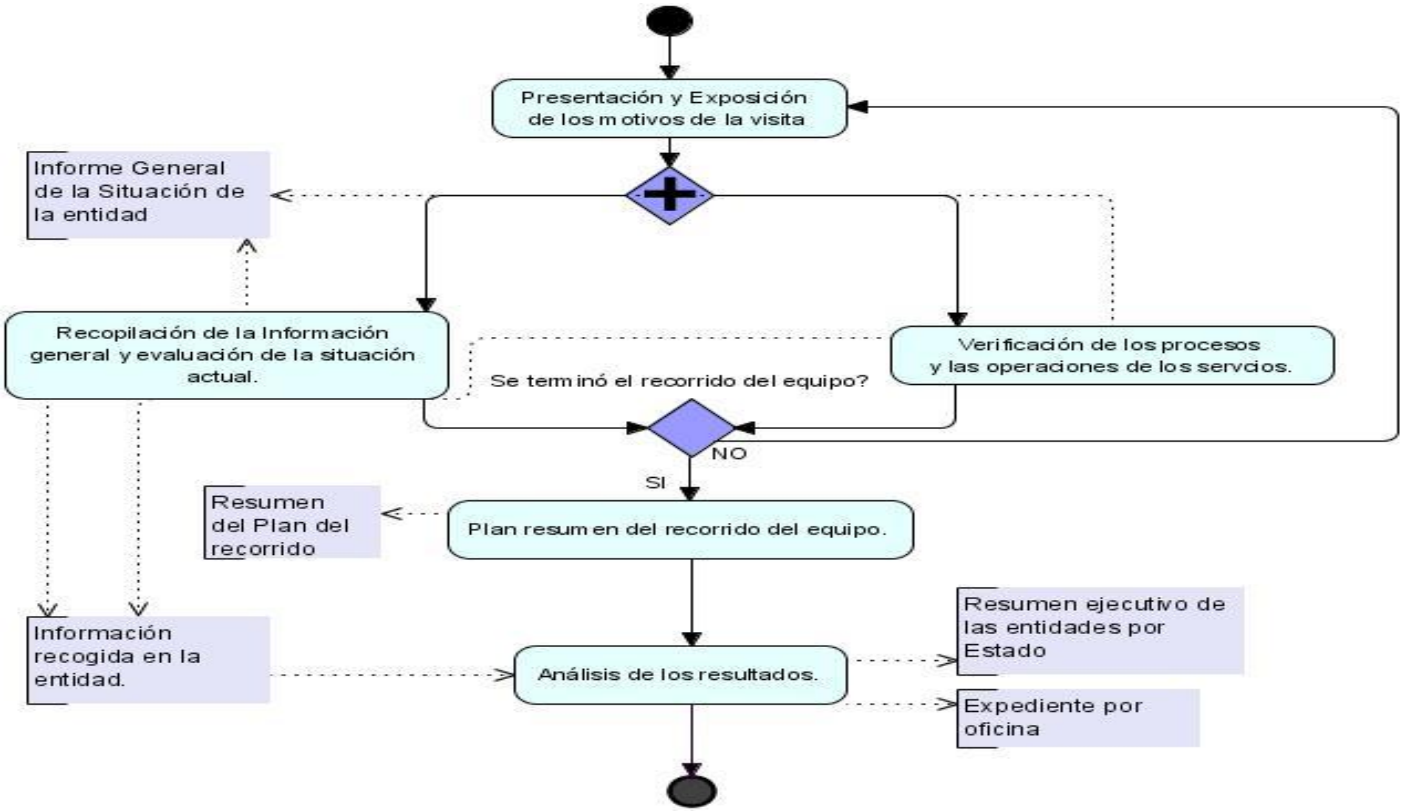


Figura 5. Diagrama de Procesos del Diagnóstico

Proceso	**Presentación y exposición de los motivos de la visita.**
Responsable	Equipo de Diagnóstico (Jefe de Equipo)
Objetivos	Realizar la presentación del equipo de diagnostico y exponer los motivos de la visita.
Involucrados	Los roles de ambas partes.
Entradas/ Responsable	
Salidas/ Responsable	

Tabla 4. Proceso: Presentación de los motivos de la visita.

Proceso	**Recopilación de la Información General y evaluación de la situación Actual**
Responsable	Equipo de Diagnóstico (Jefe de quipo y Especialista en áreas a diagnosticar.)
Objetivos	Obtener los datos relacionados con la plantilla del personal y sus características fundamentales, la estructura que presenta la oficina con las relaciones entre ellas, la cantidad de equipamiento existente y la cantidad posible o necesaria que puede asumir en el cambio. Conjuntamente se deben especificar los servicios que presta la entidad, así como cuál de estos es el más relevante en sus resultados y objetivos de trabajo. Además se determina el estado de las instalaciones eléctricas y de la red local. Se

	mitigan riesgos y se recomiendan soluciones referentes a los aspectos diagnosticados.
Involucrados	Los roles de ambas partes.
Entradas/ Responsable	
Salidas/ Responsable	Informe General de la situación de la Entidad / **Jefe de equipo**. Información recogida en la entidad / **Jefe de equipo**.

Tabla 5. Proceso: Recopilación de la Información General y evaluación de la situación Actual.

Proceso	**Verificación de los procesos y las operaciones de los servicios.**
Responsable	Equipo de Diagnóstico (Especialista de Cambio Transformacional)
Objetivos	Realizar un recorrido por las áreas de la entidad e identificar los procesos y las operaciones de cada servicio que presta la misma. Determinar de forma general el nivel de organización del servicio así como el levantamiento de aquellos problemas organizativos que puedan ser detectados o aportados por el personal entrevistado.
Involucrados	Los roles de ambas partes
Entradas/ Responsable	
Salidas/ Responsable	Informe General de la situación de la entidad. / **Jefe de Equipo**.

Tabla 6.Proceso: Verificación de los procesos y las operaciones de los servicios.

Proceso	**Plan resumen del recorrido del equipo**
Responsable	Equipo de Diagnóstico (Jefe de equipo)
Objetivos	Resumir y analizar el cumplimiento del plan del recorrido propuesto al equipo.

Involucrados	Equipo de diagnóstico.
Entradas/ Responsable	
Salidas/ Responsable	Resumen del plan de recorrido / **Jefe de equipo.**

Tabla 7. Proceso: Plan Resumen de la Jornada del Recorrido

Proceso	**Análisis de los resultados.**
Responsable	Equipo de Diagnóstico (Jefe de Diagnóstico).
Objetivos	Analizar los datos recogidos en el proceso de diagnóstico. Elaborar los informes finales.
Involucrados	Equipo de Diagnóstico.
Entradas/ Responsable	Información recogida en las entidades/ **Jefe de Equipo.**
Salidas/ Responsable	Resumen ejecutivo de las entidades por Estado/ **Equipo de Diagnóstico** Expediente por entidad/ **Equipo de Diagnóstico.**

Tabla 8.Proceso: Análisis de los Resultados.

Actividades y Descripción de los procesos.

A. **Presentación y exposición de los motivos de la vista**

En esta primera etapa se pretende realizar una presentación adecuada del equipo de trabajo, del personal de la entidad que los atenderá, para posteriormente el jefe del equipo exponer cuales son los objetivos de la visita, la propuesta de cambio planificada y sus características generales. Este intercambio inicial se debe realizar con la intención de motivar a la dirección de la entidad y a su personal implicado. En esta presentación se debe utilizar el formato donde se exponen los aspectos fundamentales a presentar, así como la propuesta de cambio que se ha planificado implantar en la entidad.

Actividades

En esta sección se definen un conjunto de actividades que son importantes para el proceso de Presentación y exposición de los motivos de la visita como son:

Exponer los objetivos de la Visita: En esta actividad se exponen a los clientes cuales son los objetivos de la visita.

Exponer los cambios propuestos a realizar: Se basa en explicar a la contraparte la propuesta de trabajo a realizar, en este caso a través de la aplicación se recogen los datos relacionados con la entidad.

Motivar el compromiso de la dirección: Se trata de comprometer al responsable de la entidad y al personal que estará involucrado con las actividades que se desarrollarán a colaborar con todo lo que el equipo de Diagnóstico necesite.

B. **Recopilación de la Información General y evaluación de la situación Actual.**

En este proceso el Jefe de equipo debe recopilar toda la información general que pueda caracterizar a la entidad, es necesario realizar un recorrido por la instalación constituye parte esencial del Diagnóstico.

En esta etapa el intercambio puede ser de ambas partes y el jefe de equipo puede ayudar y explicar todos los aspectos capturados, de igual modo puede adicionar alguna información que no haya sido analizada en el mismo y sea de gran importancia para los resultados de la visita.

Actividades

Realizar una caracterización general de la entidad: Se recogen datos referentes al tamaño de la plantilla y sus características, áreas de la entidad y sus relaciones, equipamiento existente y posibilidades futuras de adquisición, servicios que presta y perspectivas de desarrollo planificadas y se medir conocimientos informáticos de los trabajadores.

Evaluar las Instalaciones eléctricas: Se inspecciona el local verificando el aspecto referente a las instalaciones eléctricas, se llegan a conclusiones de la situación de la entidad, mitigando riesgos y ofreciendo recomendaciones por parte del especialista.

Evaluar la Red Informática: Se inspecciona el local verificando el aspecto referente a la red informática del lugar, se llegan a conclusiones de la situación de la entidad, mitigando riesgos y ofreciendo recomendaciones por parte del especialista.

C. **Verificación de los procesos y las operaciones de los servicios**.

Se realiza un recorrido por las áreas de la entidad y se identifican los procesos y las operaciones de cada servicio que presta el mismo. Se hace una valoración general del nivel de organización del servicio (alto, Medio o Bajo), así como el levantamiento de aquellos problemas organizativos que puedan ser detectados o aportados por el personal entrevistado.

Actividades

Realizar una verificación general de los procesos: Se identifica la organización de los procesos de la entidad que en un futuro serán comparados con los procesos de otras entidades para una mejor comprensión de los procesos de todas las entidades de manera general.

Realizar una evaluación del nivel de organización del servicio que se presta: Se analiza que tan organizados están los procesos de la entidad, de forma que sirvan de referencia a la hora de crear la aplicación.

Realizar un levantamiento de los problemas detectados: Si se encuentra algún tipo de problema durante la identificación de los procesos se hace un levantamiento de éstos para analizarlos entre ambas partes al final de la visita.

D. **Plan resumen del recorrido del equipo.**

En este proceso el Jefe de Equipo hace un resumen general de todo lo ocurrido durante la jornada, se reúne el equipo y recoge en una plantilla el porciento de cumplimiento que llevan en cada jornada y clasifican el recorrido, permitiendo corroborar errores y mejorar el proceso.

Actividades

Llenar los datos referidos en la planilla: En esta actividad se recogen los datos referidos a la cantidad de entidades visitadas en correspondencia con el plan establecido, determinando el porciento de

cumplimiento del plan. Se incluyen aspectos referidos al alojamiento del equipo en todo su recorrido y se anexan observaciones.

Elaborar conclusiones de la jornada del recorrido: Se mitigan riesgos, se resumen experiencias y se ofrecen recomendaciones para mejorar el proceso del recorrido. Se elabora el informe final del equipo.

E. **Análisis de los resultados.**

Se gestionan los datos recogidos durante el proceso de Diagnóstico. Se clasifica la información recogida en el diagnóstico, realizando un estudio a fondo de cada entidad, evaluando su situación actual. Es el proceso en el que se toman las decisiones a tener en cuenta en el proceso de desarrollo de software, específicamente en el Despliegue.

Actividades

Estudiar la información recogida: En esta actividad se estudian a fondo las planillas llenadas por cada uno de los equipos de diagnóstico en el recorrido por los estados y se clasifican las entidades.

Generar informes finales: Se generan los informes con la información general obtenida en cada una de las oficinas visitadas, entre ellos se encuentran el Resumen ejecutivo de las entidades por Estado y el Expediente de la entidad.

Reglas del negocio

Acción

Restricciones de Operaciones

- Se debe recoger de forma obligatoria datos referentes a las instalaciones eléctricas, se incluyen descripción, estado y cantidad.
- Se debe recoger de forma obligatoria datos referentes a la red informática, se incluyen descripción, observaciones y cantidad.
- Se debe recoger de forma obligatoria todos los datos referentes a la entidad en general.
- Se deben recoger de manera obligatoria, en caso de existencia, observaciones referentes a cada equipo existente en la entidad.
- Se deben dejar en la entidad en algún formato de texto las observaciones generales y las recomendaciones referentes a las instalaciones eléctricas y la red informática existente.

2.3 Descripción del Sistema Propuesto.

Para cumplimentar el objetivo propuesto al inicio de este trabajo y teniendo en cuenta los procesos de negocio planteados, el sistema que se propone constará de 2 componentes:

1. Componente Diagnóstico: En este componente se recoge la información necesaria para diagnosticar cada entidad. Tiene como salida una planilla en formato XML la cual contiene los datos capturados y sirve como entrada para el otro componente.

2. Componente Procesador: En este componente se procesa toda la información recogida a través de las planillas en formato XML que se generan en el primer componente y se visualizan los reportes necesarios para auxiliar la elaboración de los informes finales y potenciar la toma de decisiones.

2.3.1 Despliegue

Los componentes del sistema estarán distribuidos de la siguiente manera:

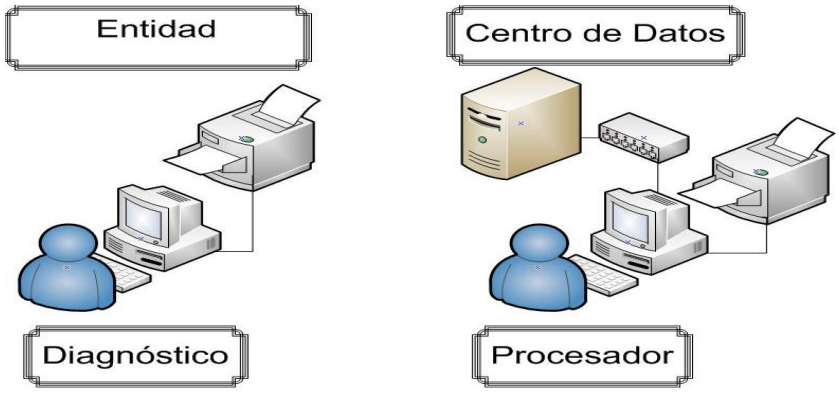


Figura 6. Distribución

El componente Diagnóstico se encuentra en cada entidad que será diagnosticada, de manera que se facilite la captura de la información necesaria para procesar la información en el componente Procesador, este se encontrará en el Centro de Datos del proyecto ubicado en Caracas y a través de él se podrán tomar las decisiones pertinentes y llegar a los resultados del Proceso de Diagnóstico.

2.3.2 Product Backlog / Pila de Tareas del Producto.

Los requisitos del sistema o del producto desarrollado se enumeran en el Product Backlog. El Product Owner es responsable del contenido, priorización, y disponibilidad de este: nunca se acaba y es usado en la planificación del proyecto, es simplemente una estimación inicial de los requisitos. El Product Backlog se desarrolla paralelamente a medida que el producto y el ambiente en el cual se trabaja evoluciona, es dinámico; maneja constantemente los cambios para identificar que necesita el producto para ser: apropiado, competitivo, y útil. Mientras exista un producto, el Product Backlog también existe.

ID	Nombre	Descripción	Estado
Requisitos Críticos			
1	Capturar Información.	El sistema debe recoger el tamaño de la plantilla y sus características. ▪ La cantidad de trabajadores, de ellos la cantidad de fijos y contratos. ▪ La cantidad de hombres y mujeres. ▪ La cantidad de trabajadores por puesto de trabajo y la descripción de dichos puestos.	100%
2	Capturar Información.	El sistema debe recoger la localización de la entidad. El nombre, el estado en que se encuentra, el municipio, el nombre del responsable principal, la dirección y el teléfono.	100%
3	Capturar Información.	El sistema debe recoger el porciento de los conocimiento informáticos del personal de la entidad, caracterizándolos en : ▪ Expertos. ▪ Medios. ▪ Iniciales. ▪ Ninguno.	100%
4	Capturar Información.	El sistema debe permitir recoger aspectos generales	100%

		de la entidad como: ▪ Si el local de la entidad es alquilado o no. ▪ El estado constructivo de la entidad, caracterizándolo en bueno o malo. ▪ La cantidad de trámites en un período determinado. ▪ Si CantV ha instalado el equipamiento previsto. ▪ Si cuenta con algún Sistema de Seguridad en la instalación. ▪ Si tienen Sistema Informático, Red o Internet. ▪ ¿Qué tipo de servicios presta? ▪ Si ofrece servicios gratuitos o no. ▪ ¿En cuál banco pagan? ▪ Normas o disposición legal por la que se rigen. ▪ Tamaño de los documentos que generan.	
5	Capturar Información.	El sistema debe recoger los procesos y operaciones de los servicios que se prestan en la entidad. Se debe recoger el orden en que se realizan, las entradas y salidas que posee el proceso, el responsable y la descripción del mismo.	100%
6	Capturar Información	El sistema debe recoger la cantidad de equipos que posee la entidad, caracterizado en tipo de equipo, descripción y cantidad del mismo. Además debe permitir conocer si hay posibilidad de adquisición futura de algún equipamiento, recogiendo el responsable del mantenimiento de los futuros equipos y el proveedor.	100%

7	Capturar Información	El sistema debe recoger las perspectivas de desarrollo o movimiento de la entidad en cuanto al local. Permitiendo conocer si hay posibilidad de mudanza o de reparación del sitio.	100%
8	Capturar Información.	El sistema debe recoger cualquier observación o sugerencia por parte de los especialistas del diagnóstico referente a cualquier aspecto de la entidad y permitir imprimirla.	100%
9	Capturar Información.	El sistema debe recoger las características del equipamiento existente en la entidad. En caso de que sea : ▪ Computadora: La capacidad de disco duro, RAM, tipo de procesador, número de serie, si posee monitor LCD o UPS, el estado en que se encuentra y alguna observación referente al equipo. ▪ Impresora: La marca, el modelo, la forma, el número de serie, el estado en que se encuentra y observaciones referentes al equipo. ▪ Scanner: La marca, el modelo, si cuenta con un CD de instalación, el estado en que se encuentra, el número de serie y observaciones referentes al equipo. ▪ Fotocopiadora: La marca, el modelo, el número de serie, el estado en que se encuentra y	100%

		observaciones referentes al equipo. ▪ Cualquier otro equipo: El tipo de quipo, la marca, le modelo, el número de serie y las observaciones referentes al equipo. Además el sistema debe permitir recoger los riesgos observados por el especialista del equipo de diagnóstico en cuanto a los equipos de la instalación.	
10	Capturar Información.	El sistema debe permitir realizar un levantamiento de las instalaciones eléctricas de la entidad. Recogiendo en el caso necesario la descripción, cantidad y el estado en que se encuentra de: ▪ Tableros Eléctricos. ▪ Conductores Eléctricos. ▪ Capacidad de los Conductores Eléctricos. ▪ Conductor de Tierra. ▪ Concentración de Carga en los Nodos. ▪ Tomacorrientes. ▪ Empalmes Eléctricos. ▪ Estabilidad del Suministro Eléctrico. ▪ Densidad de tomacorrientes por PC a instalar. En caso de la existencia de otro tipo de instalación eléctrica, el sistema debe permitir recoger esta misma información referente al nuevo tipo. Además debe recoger las recomendaciones o adecuaciones eléctricas de la entidad y los riesgos u	100%

		observaciones de la misma en cuanto a este aspecto e imprimir las primeras.	
11	Capturar Información.	El sistema debe permitir realizar un levantamiento sobre la red informática que existe en la instalación. Recogiendo en el caso necesario la cantidad, descripción y observación referente a: ▪ Computadoras a Instalar. ▪ Acces Points a montar. ▪ UPS de 300 VA a montar. ▪ UPS de 750 VA a montar ▪ Estructura constructiva. En caso de la existencia de otro tipo de clasificación en cuanto a la red informática, el sistema debe permitir recoger esta misma información referente al nuevo tipo. Además debe recoger las recomendaciones o adecuaciones de la entidad y los riesgos u observaciones de la misma en cuanto a este aspecto e imprimir las primeras.	100%
12	Capturar Información.	El sistema debe permitir recoger información referente al parte resumen de la Jornada de Recorrido e imprimir dicho parte. Donde se capturarán datos para contribuir al seguimiento del cumplimiento del recorrido tales como: ▪ Día ▪ Fecha ▪ Estado	100%

		▪ Municipio ▪ Porciento del Plan de Visita ▪ Porciento del Plan Real ▪ Porciento de Cumplimiento ▪ Alojamiento ▪ Observaciones referentes.	
13	Procesar Información.	El sistema debe permitir visualizar e imprimir reportes en cuanto a la clasificación de las entidades, según la complejidad de las mismas. Clasificadas en cuanto a la cantidad de funcionarios en : ▪ Entidades Pequeñas (Hasta 5) ▪ Entidades Medianas (Entre 6 y 10) ▪ Entidades Grandes (Desde 11 en adelante) Este reporte debe mostrar el nombre de la entidad, el estado, municipio, tipo de entidad y la cantidad de funcionarios.	100%
14	Procesar Información.	El sistema debe permitir visualizar e imprimir reportes en cuanto a la caracterización de las entidades en Desplegables o no en cuanto a la Situación eléctrica y de redes: Clasificadas en cuanto a la clasificación de los especialistas de redes e instalaciones eléctricas en: ▪ Categoría1 Desplegables sin dificultad (Serán las entidades que están categorizadas	100%

		por ambos especialistas como desplegables sin recomendaciones (adecuaciones eléctricas o de red) ni riesgos (observaciones)). ▪ Categoría 2 Desplegables si se hacen adecuaciones (Serán las entidades que están categorizadas como desplegables por ambos especialistas pero tienen recomendaciones). ▪ Categoría 3 Desplegables con responsabilidad de fallos (Serán las entidades que están categorizadas como desplegables por ambos especialistas, con alguna recomendación (adecuación eléctrica o de red)) o categorizadas por uno de los especialistas. ▪ Categoría 4 No desplegables (Entidades categorizadas por ambos especialistas como no desplegables) Este reporte mostrará el nombre de la entidad, estado, municipio y el nombre de los especialistas. El sistema debe permitir una opción para visualizar los detalles de las recomendaciones y riesgo, escritas por los especialistas.	
15	Procesar Información.	El sistema debe permitir visualizar e imprimir reportes de los procesos que se realizan en cada entidad. En este reporte se mostrará por entidad los procesos	100%

		que realiza, el orden del proceso, nombre, las entradas, salidas, responsable y descripción del proceso.	
16	Procesar Información.	El sistema debe permitir auxiliar a través de reportes de los datos recogidos en cada entidad, la elaboración de los informes finales: ▪ Resumen ejecutivo de los Registros por Estado. ▪ Expediente de la entidad. Este reporte mostrará los datos generales de la entidad, la información recogida sobre el equipamiento existente, las instalaciones eléctricas y la red informática existente. Debe permitir la impresión de cada reporte.	100%
Requisitos Opcionales (Alcance_Modelación)			
17	Generar Planillas.	El sistema debe permitir elaborar planillas para recoger información, detallando el tipo de información a capturar y la forma de captura.	100%
18	Generar Planillas.	El sistema debe contener elementos de selección simple, múltiple y condicional.	100%
19	Generar Planillas.	El sistema debe contener elementos de texto.	100%
22	Generar Planillas.	La planilla puede tener varias páginas.	100%
23	Generar Planillas.	Los elementos deben ser configurables.	100%
24	Generar Planillas.	La planilla debe poder generarse y guardarse en un formato portable, preferentemente XML.	100%

25	Completar Planillas.	El sistema debe permitir cargar las planillas previamente elaboradas.	100%
26	Completar Planillas	El sistema debe permitir completar dichas planillas.	100%
27	Completar Planillas	Validar las entradas.	100%
28	Completar Planillas	Permitir generar las planillas completadas en un formato portable, preferentemente XML, exportando solamente las preguntas y sus respuestas.	100%
29	Procesar Planillas.	El sistema debe permitir cargar las planillas generadas en el componente “Cargar planillas”.	100%
30	Procesar Planillas.	Almacenar las preguntas y respuestas recogidas.	100%
31	Procesar Planillas.	Permitir mostrar reportes convenidos.	100%
32	Procesar Planillas.	Permitir imprimir reportes convenidos.	100%
33	Procesar Planillas.	Permitir la autenticación para el acceso a datos.	100%

Tabla 9. Product Backlog

2.3.3 Sprint Backlog / Pila de Tareas del Sprint.

El Sprint Backlog define el trabajo, o las tareas, que el equipo desarrollará para poder convertir el Product Backlog seleccionado para ese Sprint, en un incremento potencialmente funcional del producto. El equipo crea una lista inicial de estas tareas en la segunda parte del Sprint planning meeting Las tareas deben ser divididas de modo que cada una demore entre 4 a 16 horas finalizarlas. Las tareas de largo mayor de 16 horas se consideran secundarias, ya que todavía no se han definido apropiadamente. Solamente el equipo puede cambiar el Sprint Backlog. El Sprint Backlog es un cuadro altamente visible, es lo que debe conseguir durante una iteración del Sprint. A continuación se tabulan las planificaciones de cada los Sprint creados para implementar el sistema.

Pila del Sprint I (Base de Datos)				
Backlog Tarea	**Componente**	**Estado**	**Estimado**	**Real**

Diseño y Generación de la Base de Datos.	Procesador	100%	5	6

Tabla 10 Pila del Sprint I (Base de Datos)

Con el término del Sprint I se tiene logrado el 10% de la solución total puesto que sus funciones son vitales para el funcionamiento del sistema.

Pila del Sprint II (Construcción del Componente Diagnóstico)				
Backlog Tarea	**Componente**	**Estado**	**Estimado**	**Real**
Diseñar e Implementar formularios				
F-01-02 (Información General de la Entidad).	Diagnóstico	100%	10	11
F_01_02_01 (Características del equipamiento existente).	Diagnóstico	100%	6	6
F_01_03 (Levantamiento de las Instalaciones Eléctricas).	Diagnóstico	100%	7	7
F_01_04 (levantamiento de la Red Informática Existente).	Diagnóstico	100%	6	7
F_01_05 (Parte Resumen de la Jornada del Recorrido).	Diagnóstico	100%	6	6
Impresión de las Observaciones de las Instalaciones Eléctricas.	Diagnóstico	100%	5	5
Impresión de las Observaciones de la Información General.	Diagnóstico	100%	4	4
Impresión de las Observaciones de la Red Informática Existente.	Diagnóstico	100%	4	4
Impresión del Resumen del Recorrido.	Diagnóstico	100%	4	5
Formulario Principal.	Diagnóstico	100%	6	7
Serializar Información recogida.	Diagnóstico	100%	3	3

Tabla 11 Pila del Sprint II (Construcción del Componente Diagnóstico)

El Sprint II deja completamente funcional el componente Diagnóstico. Con este se puede lograr recoger toda la información necesaria para diagnosticar la entidad. Con la culminación del componente Diagnóstico se garantiza el cumplimiento del 50 % del sistema propuesto.

Pila del Sprint III (Construcción del Componente Procesador)				
Backlog Tarea	**Componente**	**Estado**	**Estimado**	**Real**
Deserializar la información recogida.	Procesador	100%	1	1
Implementar el Acceso a Datos.	Procesador	100%	10	11
Diseñar e Implementar formularios				
Complejidad.	Procesador	100%	9	9
Desplegable.	Procesador	100%	6	6
Detalles.	Procesador	100%	7	8
Procesos.	Procesador	100%	6	7
Datos Generales.	Procesador	100%	9	9
Impresión Complejidad.	Procesador	100%	5	5
Impresión Desplegables.	Procesador	100%	4	6
Impresión Detalles.	Procesador	100%	4	4
Impresión Procesos.	Procesador	100%	4	4
Impresión Características Entidad	Procesador	100%	10	10
Impresión Funciones	Procesador	100%	7	6
Impresión Características Instalaciones	Procesador	100%	5	5
Impresión Características Red	Procesador	100%	4	5
Impresión Características Observaciones	Procesador	100%	3	3

Impresión Características Equipo	Procesador	100%	3	3
Impresión Características Personal	Procesador	100%	2	2
Formulario Principal.	Procesador	100%	5	5

Tabla 12 Pila del Sprint III (Construcción del Componente Procesador)

El Sprint II deja completamente funcional el componente Procesador. Con este se puede lograr procesar toda la información necesaria para diagnosticar la entidad. Con la culminación del componente Procesador se garantiza el cumplimiento del 100 % de las funcionalidades críticas del sistema propuesto. Dejando como recomendación la implementación de los requisitos funcionales opcionales para próximas etapas de implementación.

Pila del Sprint IV (Modelación de los requisitos opcionales)				
Backlog Tarea	**Componente**	**Estado**	**Estimado**	**Real**
Diagrama de Casos de Uso del Sistema.	Todos	100%	4	4
Descripción de Casos de Uso del Sistema.	Todos	100%	2	2
Descripción de actores del Sistema	Todos	100%	1	1
Diagramas de interacción (Secuencia de Diseño).	Todos	100%	2	2
Diagramas de Clases de Diseño por Casos de Uso del Sistema.	Todos	100%	3	3
Diagrama de Clases Persistentes.	Todos	100%	2	2
Diagrama Entidad Relación	Todos	100%	2	2

Tabla 13 Pila del Sprint IV (Modelación de los requisitos opcionales)

El Sprint IV se desarrolla para modelar los requisitos opcionales del Product Backlog con el objetivo de utilizarlo para futuras aplicaciones, es por este motivo que se adaptan a esta metodología algunos

artefactos que no son propios de ella, se toma como base los artefactos de la metodología RUP (Proceso Unificado Racional).

Pila del Sprint V (Validación)				
Backlog Tarea	**Componente**	**Estado**	**Estimado**	**Real**
Pruebas de caja negra por parte del equipo de calidad del proyecto SAREN.	Todos	100%	14	15
Verificación del diseño a través de métricas de diseño por parte del equipo de calidad del proyecto SAREN.	Todos	100%	12	12

Tabla 14 Pila del Sprint V (Validación)

El Sprint V se concibe para dar soporte al desarrollo de la aplicación. En este Sprint es donde se documenta y prueba al dedillo el correcto cumplimiento de las funcionalidades planteadas en el Product Backlog y se verifica el diseño a través de métricas del sistema en cuestión.
Existen también requerimientos adicionales que la aplicación debe tener asociados al cumplimiento de las funcionalidades divididas en Sprint:

2.3.4 Requisitos adicionales.

- Apariencia o interfaz externa:

  Diseño sencillo, una interfaz simple de usar e interactiva para que al usuario le sea fácil el trabajo con el Sistema.

- Usabilidad:

  La instalación del sistema trae consigo una mayor rapidez en la recogida y procesamiento de información necesaria para potenciar la toma de decisiones en el proceso de Diagnóstico.

- Software:

  El servidor de base de datos debe ser PostgreSQL.

  En las máquinas debe estar instalado el framework 2.0 de .Net.

  Se necesita el Crystal Reports para .NET Framework 2.0.

Se necesita el Visor de Informes de Microsoft Visual Studio 2005.

Se necesita el Windows Installer 3.1 o versiones más avanzadas.

Se necesita el Microsoft Data Access Components 2.8

- Hardware:

  Para trabajar con el sistema de forma eficiente se necesita una máquina servidor que como mínimo debe tener las siguientes características: Pentium IV con 512 MB de RAM, un microprocesador a 2.00 GHz y una tarjeta de red Protocolo Ethernet 10/100 MB/s.

  Para máquinas clientes: Procesador a 1.4 GHz y memoria RAM de 254 MB o superior.

2.4 Conclusiones del Capítulo.

En este capítulo se ha logrado entender el negocio del sistema a desarrollar, los roles y procesos involucrados, se gestionaron los requisitos, siendo los mismos priorizados y administrados y se planificaron los diferentes ciclos o iteraciones a implementar, incluyendo un análisis estadísticos de los mismos. Se ha obtenido una panorámica del sistema a implementar, teniendo en cuenta sus componentes y la distribución de los mismos.

CAPÍTULO 3. Construcción y Validación del Sistema.

3.1 Introducción

En el presente capítulo se realiza el diseño, implementación y prueba del sistema, se describe el modelo de datos, se detalla la arquitectura utilizada, la seguridad y los patrones de diseño empleados. Se realiza la construcción de la propuesta de solución desarrollando el Product Backlog, la planificación de los Sprint, los diagramas de clases de diseño asociados a los Sprint Backlog de Scrum y se valoran los principios de diseño, el tratamiento de excepciones y el estándar de codificación. Además se realiza la validación del Sistema.

3.2 Construcción del Sistema.

3.2.1 Patrón arquitectónico.

Los patrones arquitectónicos actualmente están muy difundidos, existe una gran variedad de ellos, todos aportan sus peculiaridades con el objetivo de resolver una problemática determinada, sin embargo no siempre es beneficioso utilizar alguno, esto está condicionado a las característica del software a implementar, pues un mal uso de un patrón, puede causar pérdidas lamentables en cuanto a la complejidad del diseño y el mantenimiento del mismo, es por esto que en el presente trabajo se aprovechan solo las ventajas que brinda el patrón arquitectónico 3 capas, las cuales se enuncian a continuación.

Layers (Capas): Ayuda a estructurar aplicaciones que puede estar descompuestas en grupos de subtareas en las cuales cada grupo de subtareas está en un nivel particular de abstracción.

Aprovechando las ventajas que el mismo aportará al sistema, dadas las características del modelo, se puede implementar y dejar operativa una solución de negocios en tiempos extremadamente cortos,

permitiendo conseguir una ventaja competitiva particular respecto a otros negocios. También permite la modificación del sistema en períodos de tiempo reducidos, incluso cuando es necesario agregar características especiales a las aplicaciones.

Maneja distintas presentaciones, debido a que separa la presentación de la lógica de negocios, es mucho más sencillo realizar tantas presentaciones diferentes como dispositivos con capacidades e interfaces se tenga.

Encapsula los datos, esto permite un acceso a los datos consistente, seguro y auditable. Con esto se pretende que si hay cambios en la capa de datos, la capa de negocios se haga cargo de administrar tales cambios y la capa de presentación, en la mayor parte de los casos ni se entere.

Además ahorra tiempo y costos en el desarrollo de nuevas aplicaciones y la integración en el resto de los procesos de gestión de la empresa.

La arquitectura n-capas facilita el mantenimiento, la flexibilidad, el bajo acoplamiento, la alta cohesión, haciendo el sistema más seguro y robusto.

Ambos componentes del sistema están divido en 3 capas fundamentales:

- Presentación.
- Negocio.
- Acceso a Datos.

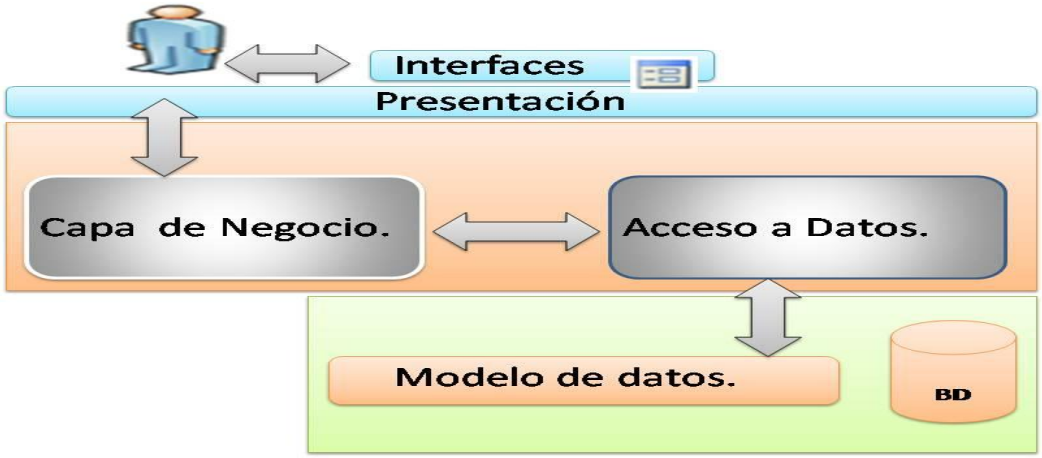


Figura 7. Arquitectura en 3 Capas

El componente Diagnóstico en la capa de Presentación posee interfaces que interactúan con el usuario, las cuales permiten la recogida de información referente a las entidades, consisten en los datos de las planillas elaboradas llevados a los formularios. Además contiene formularios para auxiliar las funciones de Impresión.

La capa de negocio, implementa toda la lógica de negocio que posibilita, auxiliada por la capa de Acceso a Datos la salva de la información en formato XML.

Por su parte el componente Procesador en su capa de Presentación posee todas las funcionalidades de reportes e impresión necesarias para diagnosticar las entidades.

El procesamiento de la información se realiza a través de la Capa de Negocio y la de Acceso a Datos.

3.2.2 Patrones de Diseño.

La realidad de los patrones de diseño está muy vinculada a la de los patrones arquitectónicos, existen diversos patrones, frutos del ingenio y la necesidad de resolver problemáticas comunes, sin embargo, la idea no es utilizarlos todos con el objetivo de tener un diseño perfecto, la estrategia es emplear, solo los que brinden ventajas al software que se implementa, es el caso de los patrones GOF, este trabajo se auxilia de los beneficios que aporta uno de ellos, el resto, sin quitarle importancia, lejos de ayudar, implicaría el aumento de clases, no necesarias en este caso, como por ejemplo la utilización del patrón estructural Fachada, empleado para el uso de interfaces que simplifiquen la interacción entre subsistemas o niveles, no resultaría provechoso en el sistema en cuestión, pues se trata de un sistema sencillo de relativamente pocas clases. Todo esto arroja la necesidad del uso de los siguientes patrones de diseño:

Patrones GRASP [21]

1. Experto.
   ¿Quién asumirá la responsabilidad?
   Asignar una responsabilidad al experto en información: la clase que posee la información necesaria para cumplir con la responsabilidad.

Se utiliza en la clase Controladora de la capa de Negocio.

2. Creador.
   ¿Quién tiene la responsabilidad de crear?

Asignar a la clase B la responsabilidad de crear una instancia de la clase A, si se cumple una de las siguientes condiciones:

B contiene A.

B agrega A.

B tiene los datos de inicialización de A.

B registra A.

B utiliza A muy de cerca.

Se utiliza en las clases de la capa de Presentación al crear objetos de la clase Entidad y la clase Controladora, pues las clases de presentación son las que poseen los argumentos para crear estos objetos.

3. Bajo Acoplamiento.

   ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

   Asignar una responsabilidad para mantener bajo acoplamiento. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras.

En todas las clases se maneja la funcionalidad de este patrón.

4. Alta cohesión.

   ¿Cómo mantener la complejidad dentro de límites manejables?

   Asignar una responsabilidad de modo que la cohesión siga siendo alta.

   En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

En todas las clases se maneja la funcionalidad de este patrón, principalmente en la clase Entidad_DAO.

5. Controlador

   ¿Quién debería encargarse de atender un evento del sistema?

   Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones:

   - el "sistema" global (controlador de fachada).

- la empresa u organización global (controlador de fachada).
- algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas).
- un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados "Manejador<NombreCasodeUso>" (controlador de casos de uso).

Se utiliza en la clase Controladora de la capa de Negocio.

Patrones GOF [21]

De ellos en el presente trabajo se utiliza uno de los patrones de creación:

1. Singleton: Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia.

Se utiliza en la clase Conexión del Negocio, para crear una única instancia del objeto conexión.

3.2.3 Seguridad.

La seguridad es un punto que debe garantizar los siguientes aspectos:

- Confidencialidad: garantizar que la información no es visible a extraños.
- Autenticación: garantiza que el acceso a la información sólo puede ser realizado por quienes proporcionan la identidad adecuada.
- Integridad: asegurar que el mensaje no ha sido modificado accidental o deliberadamente
- No repudio: garantiza que el emisor del mensaje no puede negar haberlo enviado.

En el sistema están vulnerables varios niveles, para cada uno se han detectado las posibles fallas de vulnerabilidad y se han estudiado estrategias para minimizar los riesgos.

1. Nivel de Información o Datos

   Este nivel está referido a la información que se genera en formato XML, la cual puede ser enviada por vía email o en los mejores casos entregada personalmente.

   Para establecer la seguridad en el caso que se envíe por la red:

Se utilizará el método de firma de archivos XML, empleado en el archivo que contiene los datos recogidos en las entidades, utilizando el cálculo del código hash del mismo, de esta manera identificamos que ha sido enviado por la persona autorizada y que no ha sido modificado por entes externos.

Para complementar esto se utiliza uno de los métodos de encriptación para los archivos XML, empleado en un fichero XML que contiene el código Hash del archivo anterior y el identificador de este último, a este archivo XML que contiene estos datos se le encriptará el nodo del código hash, de manera que ambos archivos viajen por la red lo más seguro posible.

Ambos métodos se auxilian el uno del otro con el objetivo de obtener una seguridad si no al 100% al 99%.

XML Encryption: el cual tiene como objetivo, proteger cierta información que se envía del acceso no autorizado por terceras partes.

Permite cifrar:

- Un documento XML completo.
- Un elemento de un documento XML.
- El contenido de un elemento
- Datos textuales que no son XML.
- Contenidos ya cifrados.

Código hash: Un HASH no es más que un número, hexadecimal generalmente, un compendio de bits que dependen bit a bit de un conjunto de bits original. Dicho conjunto de bits original puede ser un fichero, una cadena de texto entre otros. En informática, Hash se refiere a una función o método para generar claves o llaves que representen de manera casi unívoca a un documento, registro o archivo, resumir o identificar un dato a través de la probabilidad, utilizando una función hash o algoritmo hash. Un hash es el resultado de dicha función o algoritmo. [24]

Específicamente el algoritmo utilizado es el SHA-1 el cual toma como entrada un mensaje de longitud máxima 264 bits (más de dos mil millones de Gigabytes) y produce como salida un resumen de 160 bits.

También se tiene en cuenta las validaciones de entrada de datos, aspecto el cuál se le dedica un apartado en este capítulo, con el objetivo de establecer confianza en los datos recogidos.

2. Nivel de Acceso a Datos.

Se valida la entrada de datos a la base de datos, incluyendo la autenticación para insertar dichos datos y obtener reportes de la misma.

3.2.4 Diagramas de Clases de Diseño por Sprint Backlog.

Sprint I

En el diagrama siguiente se muestran las clases de diseño asociadas a esta fase que establece el diseño y construcción de los modelos de datos persistentes.

El Diagrama de Clases Persistentes: muestra toda la información que se debe recoger. Muestra el modelo de datos persistentes. Este diagrama es muy extenso por las relaciones asociadas a sus clases y la cantidad de las mismas. Se ha agrupado según sus características, entidades que muestran las características del negocio en cuestión. [Diagrama de Clases Persistentes Sprint I.]

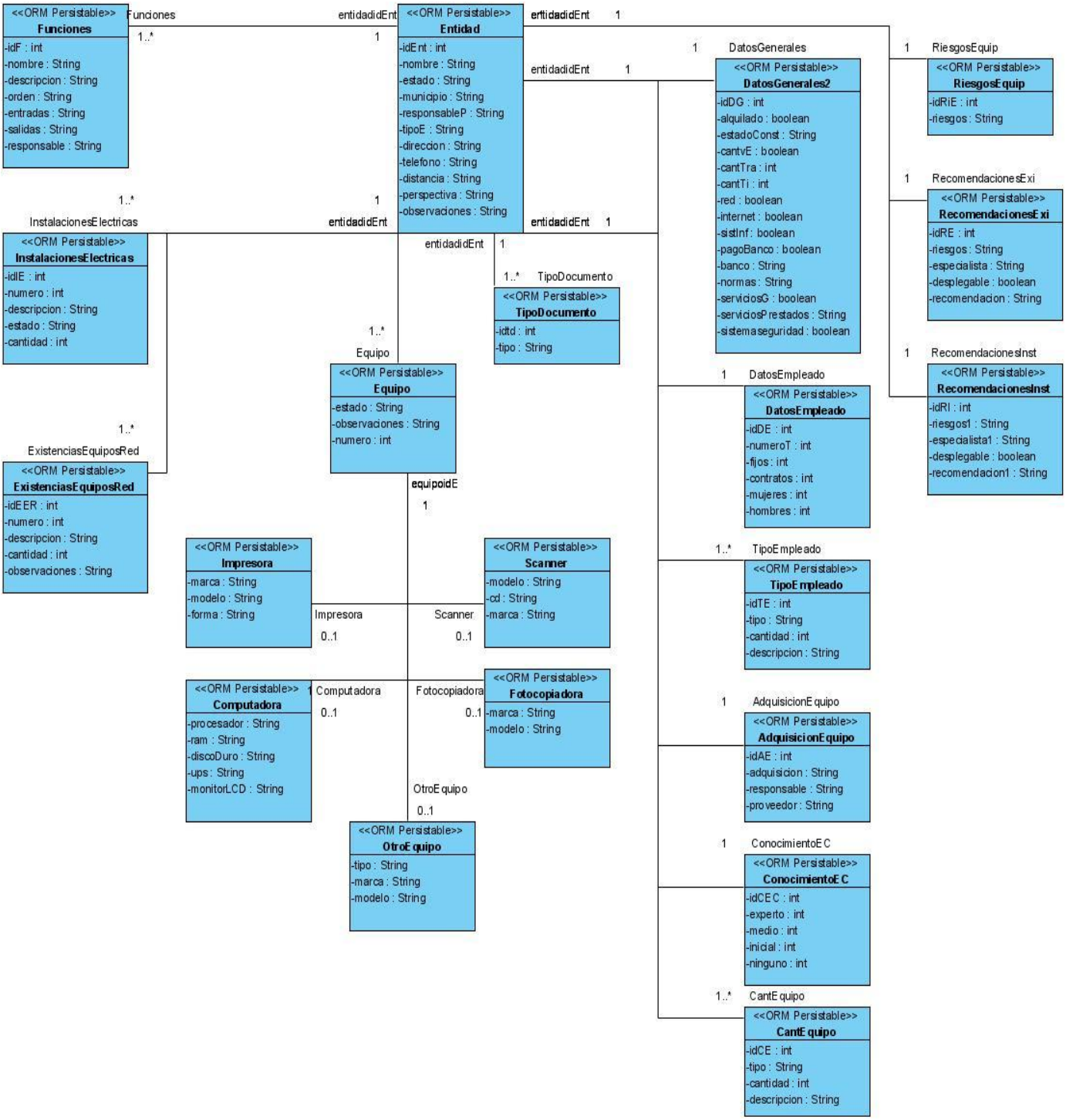


Figura 8 . Diagrama de Clases Persistentes Sprint I.

Sprint II

El Diagrama de Clases de Diseño. Muestra el patrón arquitectónico utilizado en la aplicación, está fraccionado en paquetes que encapsulan el [Diagrama para la Capa de Presentación Sprint II], el [Diagrama para la Capa de Negocio Sprint II] y el [Diagrama para la Capa de Acceso a Datos Sprint II].

Diagrama para la Capa de Presentación. Muestra todos los formularios que se utilizan en la captura de información, incluyendo aquellos formularios implementados para las funcionalidades de impresión. Algunos de sus modelos de interfaces se muestran en los anexos (1, 2, 3, 4, 5, 6, 7).

Diagrama para la Capa de Negocio. Muestra toda la lógica del negocio empleada para la captura de la información.

Diagrama para la Capa de Acceso a datos. Muestra las clases de acceso a datos, utilizadas en este Sprint para serializar los datos.

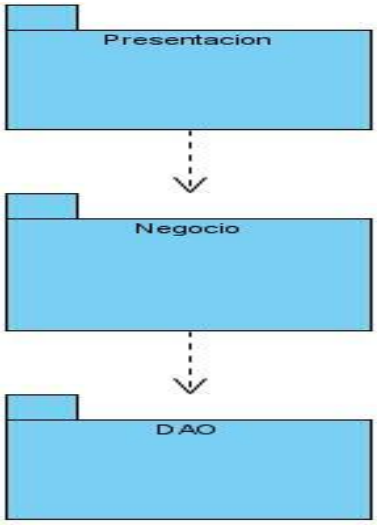


Figura 9 Diagrama de Clases de Diseño Sprint II

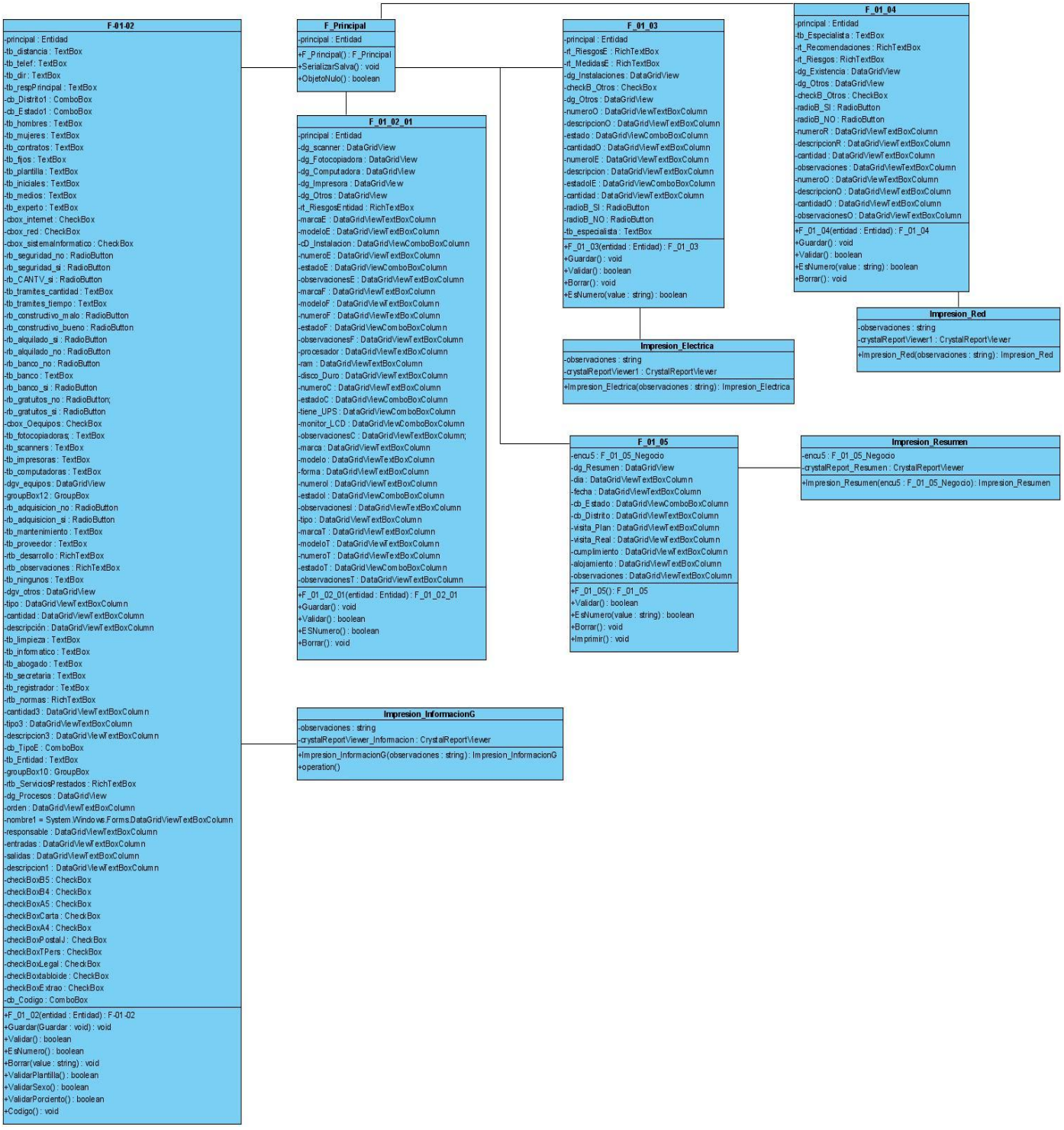


Figura 10. Diagrama de Clases de Diseño de la Capa de Presentación Sprint II.

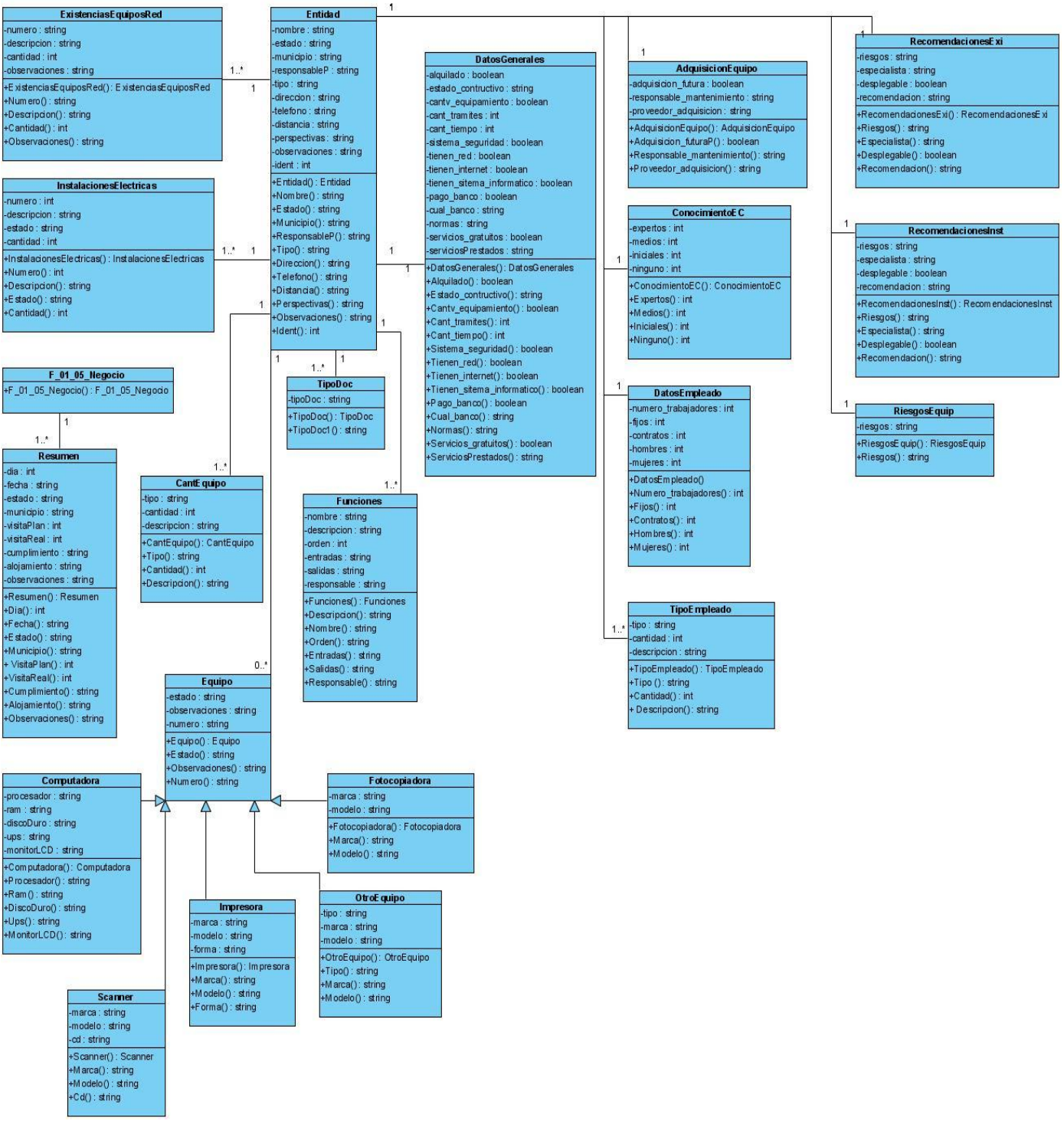


Figura 11. Diagrama de Clases de Diseño de la Capa de Negocio Sprint II

Figura 12 Diagrama de Clases de Diseño de la Capa de Acceso a Datos Sprint II.

Sprint III

El Diagrama de Clases de Diseño del Sprint III muestra el patrón arquitectónico utilizado en la aplicación, está igualmente fraccionado que el Diagrama de Clases de Diseño del Sprint II, por lo que no será representado en este apartado, no siendo así con sus respectivas partes.

Diagrama para la Capa de Presentación. Muestra todos los formularios que se utilizan en el procesamiento de la información, incluyendo aquellos formularios implementados para las funcionalidades de impresión. Algunos de sus modelos de interfaces se muestran en los anexos (8, 9, 10, 11).

Diagrama para la Capa de Negocio. Muestra toda la lógica del negocio empleada para el procesamiento de la información, incluyendo la clase auxiliar para la conexión a la base de datos.

Diagrama para la Capa de Acceso a Datos. Muestra la lógica del acceso a los datos, necesaria para lograr el procesamiento de toda la información recogida.

Figura 13 Diagrama de Clases de Diseño de la Capa de Negocio Sprint III.

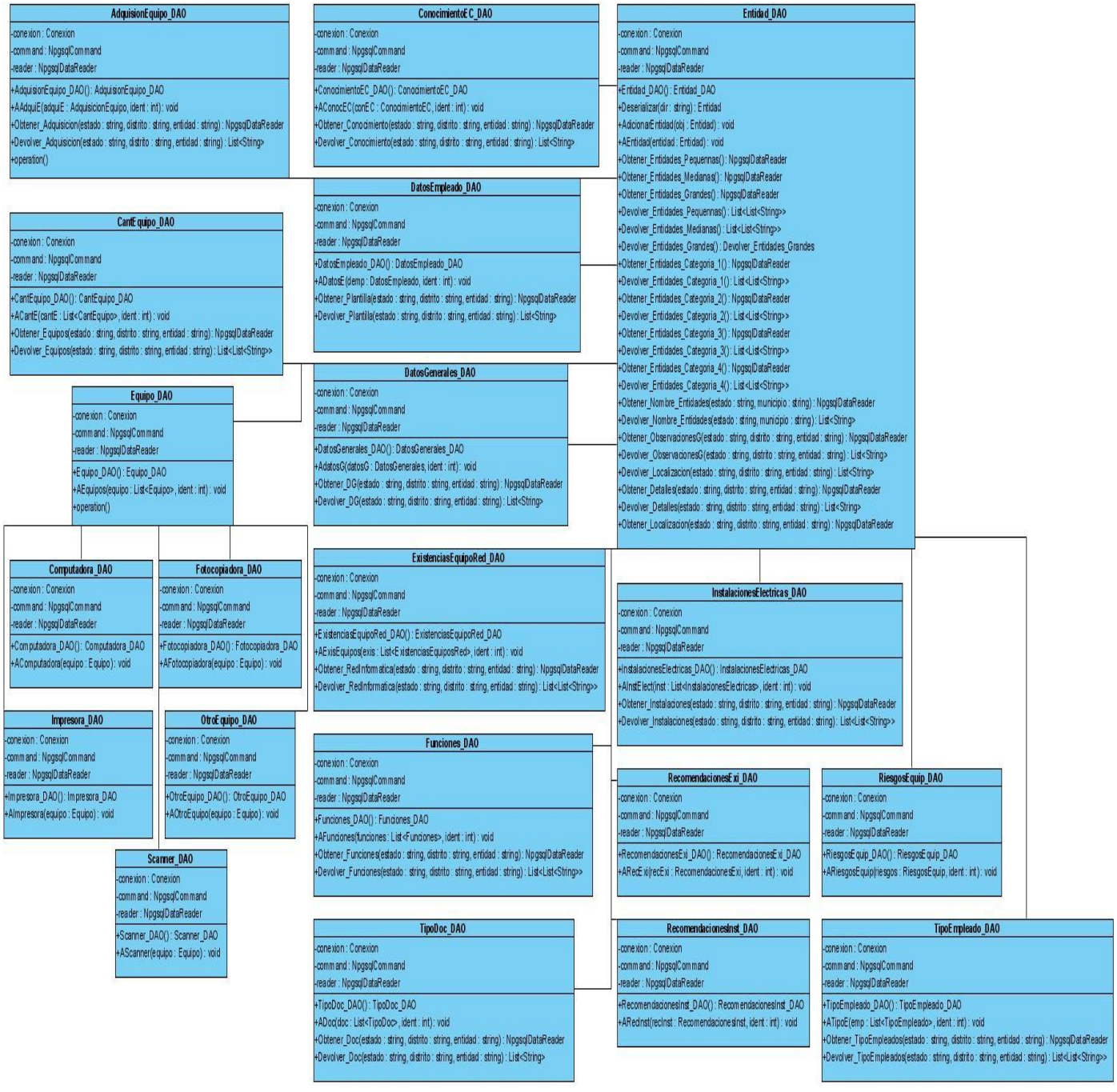


Figura 14 Diagrama de Clases de Diseño de la Capa de Acceso a Datos Sprint III.

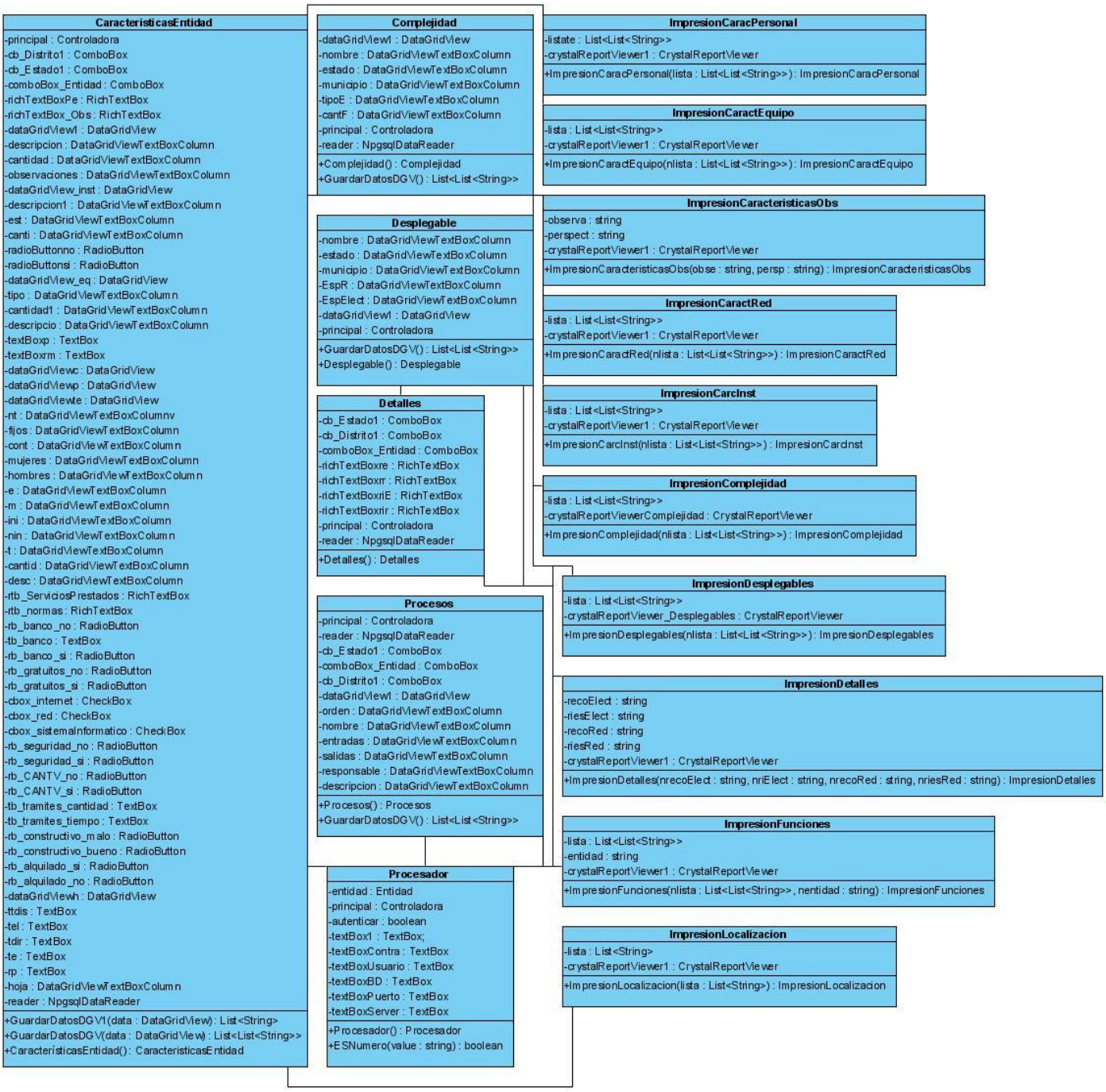


Figura 15 Diagrama de Clases de Diseño de la Capa de Presentación Sprint III

3.2.5 Diseño de la Base de Datos.

El diseño de la base de datos estuvo basado en aras de abarcar las distintas entidades necesarias en el proceso, a través del diagrama entidad relación se muestran las relaciones entre las mismas, entre estas se encuentra la herencia, la relación de uno a muchos y la de uno a uno, teniendo como clase rectora a la entidad a diagnosticar, la cual contiene al resto de las entidades, formando un ente común y compacto.

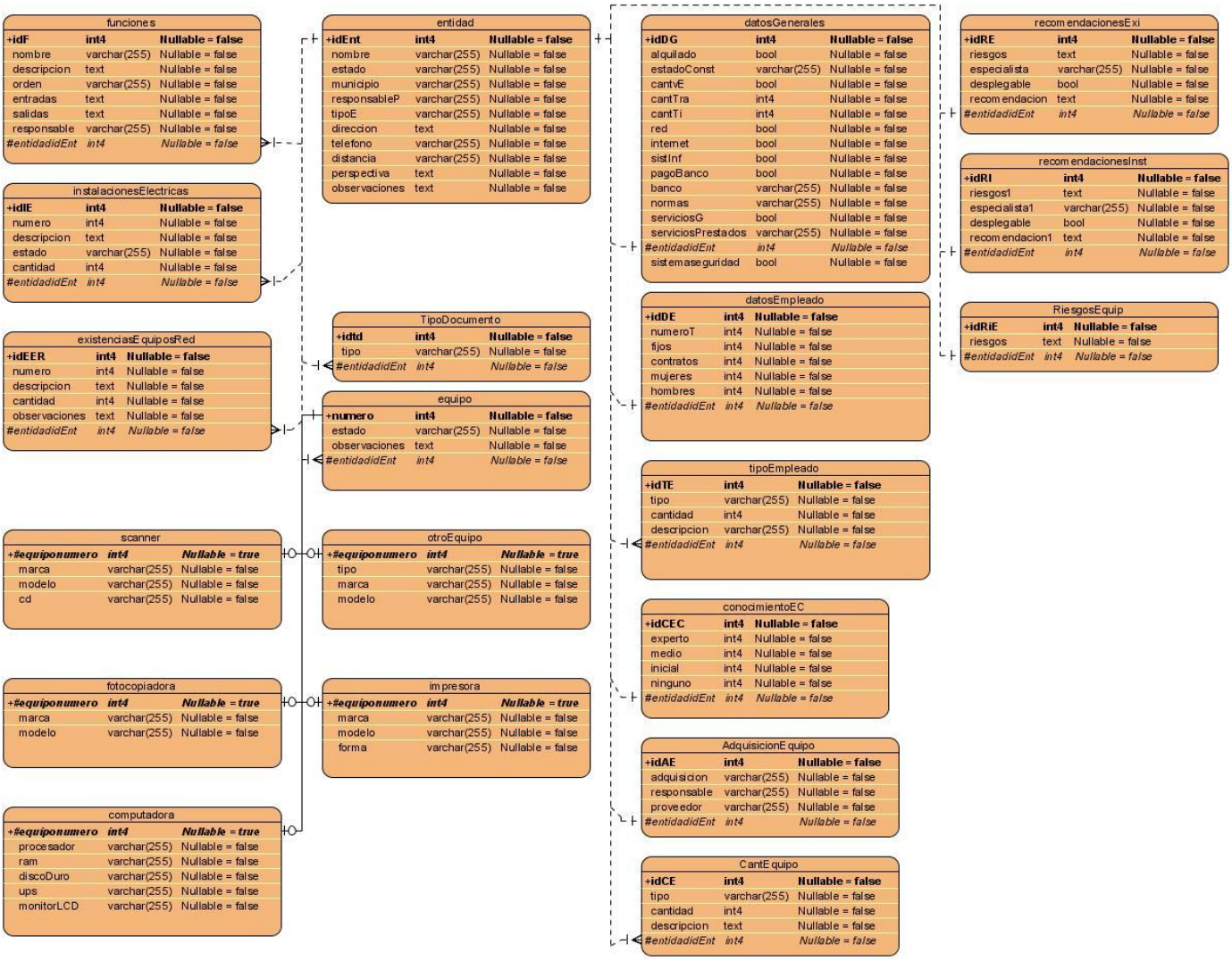


Figura 16 Modelo Físico de Datos.

3.2.6 Modelación de los requisitos opcionales.

Descripción de los Actores del Sistema

Nombre	Descripción
Generador	Se encarga de generar las planillas.
Completador	Se encarga de completar las planillas.
Procesador	Se encargan de procesar las planillas, esto incluye Autenticar, Adicionar Encuestas, Mostrar Reportes e Imprimir.

Tabla 15 Descripción de los Actores del Sistema de los Requisitos Opcionales

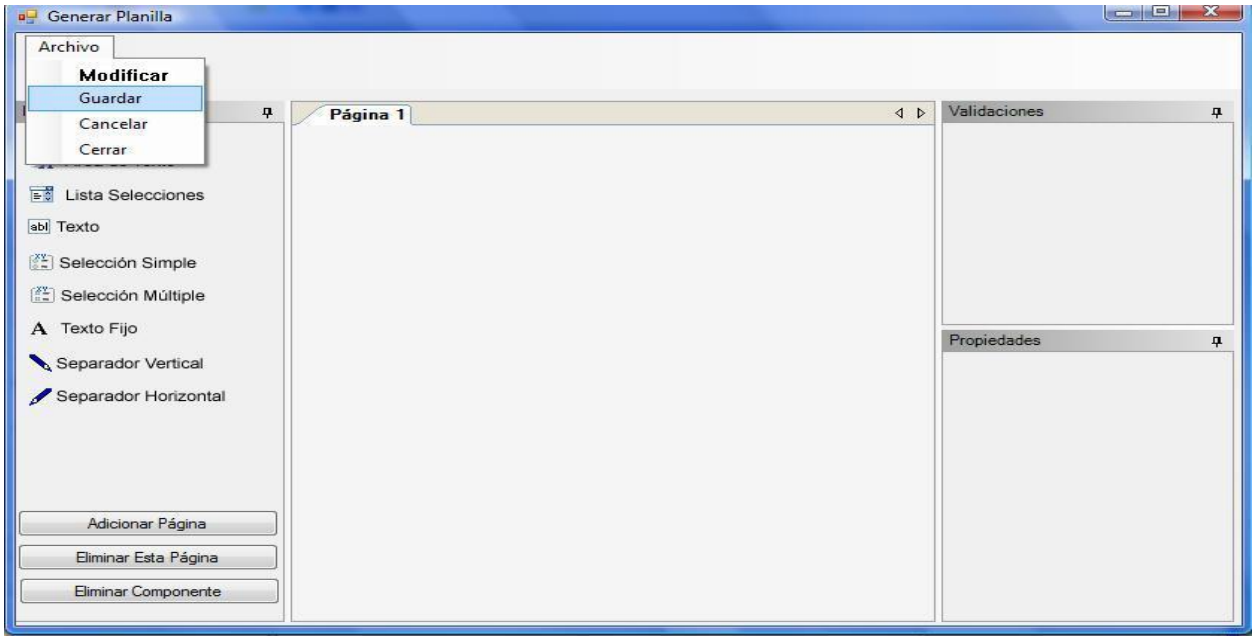
Descripción de los Casos de Uso del Sistema.

Caso de Uso:	GenerarPlanilla.
Actor:	Generador
Resumen:	El caso de uso se inicia cuando el usuario Generador se dispone a crear o modificar las planillas. Luego de elaborarlas o modificarlas las guarda en el directorio escogido por él.
Pre-condición:	El sistema debe estar instalado y ejecutándose correctamente.
Referencias	RF_17, RF_18, RF_19, RF_20, RF_21, RF_22, RF_23, RF_24
Prioridad	Opcional
Complejidad	Complejo

Nivel del caso de uso	Usuario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona una de las siguientes opciones: “Crear Planilla”. (Ver sección: Crear Planilla). “Modificar Planilla. (Ver sección: Modificar Planilla).	
Sección “Crear Planilla”	
	2. Muestra una interfaz con los componentes necesarios para crear una planilla: Texto. Etiquetas. Selección simple. Selección múltiple. Selección condicional. Y las Opciones: ▪ Guardar ▪ Modificar ▪ Cancelar ▪ Cerrar
3. Elabora la planilla y selecciona la opción “Guardar”.	4. Muestra la interfaz de salva y pide el nombre del archivo y la dirección donde se guarda.

5. Introduce los datos necesarios y selecciona la opción “Aceptar”. 7. Selecciona la opción “Aceptar” del mensaje. 8. Selecciona la opción “Cerrar”.	6. Muestra un mensaje de acción concluida satisfactoriamente. 9. Cierra la interfaz. 10. El caso de uso termina.

Prototipo de Interfaz



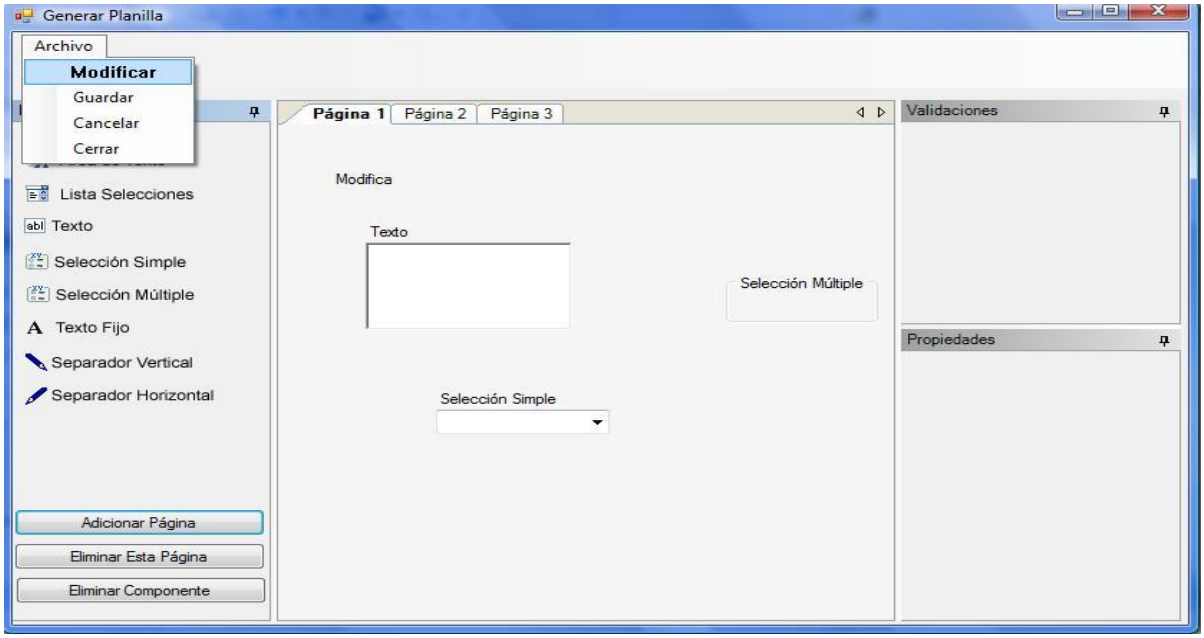
Flujo Alterno al paso 5“Operación cancelada”

Acción del Actor	Respuesta del Sistema
1. Selecciona la opción “Cancelar”.	2. Muestra un mensaje de acción cancelada y regresa al paso 2 del flujo normal de eventos.

Sección “Modificar Planilla”

Flujo Normal de Eventos

Acción del Actor	**Respuesta del sistema**
	2.Muestra una interfaz para modificar una planilla a partir de los siguientes componentes: Texto. Etiquetas. Selección simple. Selección múltiple. Selección condicional.
3. Carga la planilla a modificar a través de la opción “Cargar”.	4. Muestra la planilla cargada.
5. Modifica la planilla y selecciona la opción “Guardar”.	6. Muestra la interfaz de salva y pide el nombre del archivo y la dirección donde se guarda.
7. Introduce los datos necesarios y selecciona la opción “Aceptar”. 9. Selecciona la opción “Aceptar” del mensaje. 10. Selecciona la Opción “Cerrar”.	8. Muestra un mensaje de acción concluida satisfactoriamente. 11. Cierra la interfaz. 12. El caso de uso termina.
Prototipo de Interfaz	



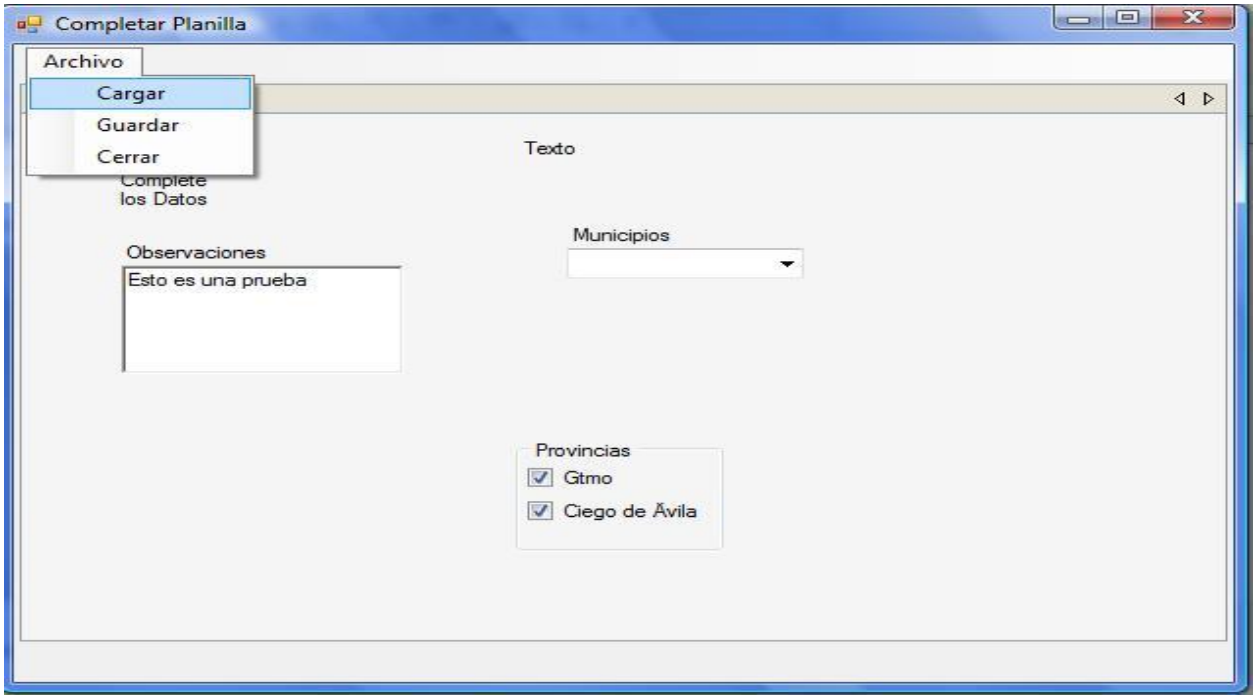
Flujo Alterno al paso 7"Operación cancelada"	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "Cancelar". 3. Selecciona la opción "Aceptar" del mensaje y regresa al paso 2 del flujo normal de eventos.	2. Muestra un mensaje de acción cancelada.
Post-condiciones	El sistema queda con nuevas planillas creadas y generadas en formato XML. El sistema queda con planillas modificadas y generadas en formato XML.

Tabla 16 Descripción del Caso de Uso del Sistema "GenerarPlanilla" de los Requisitos Opcionales.

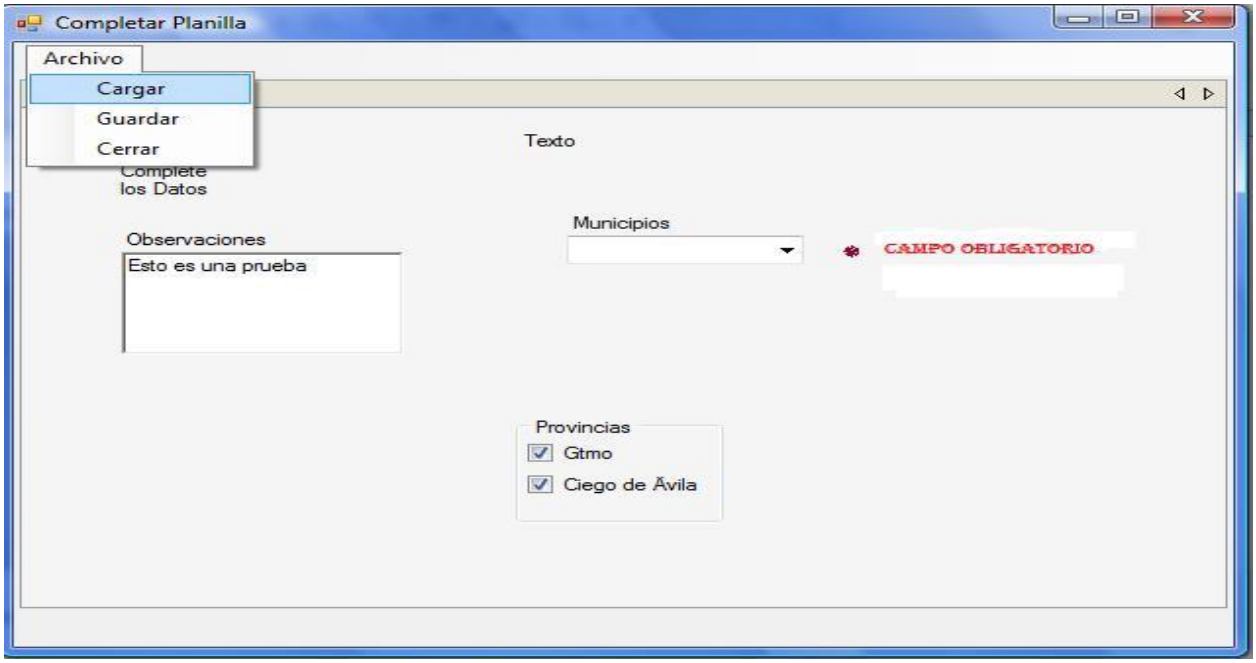
Caso de Uso:	CompletarPlanilla.
Actor:	Completador
Resumen:	El caso de uso se inicia cuando el usuario Completador se dispone a llenar las planillas. Luego de cargarlas en el sistema las completa y las guarda en el directorio escogido por él.
Pre-condición:	El sistema debe estar instalado y ejecutándose correctamente.
Referencias	RF_25, RF_26, RF_27, RF_28
Prioridad	Opcional
Complejidad	Complejo
Nivel del caso de uso	Usuario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción "Completar Planilla".	2. Muestra una interfaz con las siguientes opciones: ▪ Cargar ▪ Guardar. ▪ Cerrar.
3. Selecciona la opción	4. Muestra la interfaz para cargar la planilla y pide

"Cargar".	nombre del archivo y dirección de la salva.
5. Introduce los datos.	6. Muestra la planilla a completar.
7. Completa la planilla y selecciona la opción "Guardar".	8. Valida las respuestas. 9. Muestra la interfaz de salva y pide el nombre del archivo y la dirección donde se guarda.
10. Introduce los datos necesarios y selecciona la opción "Aceptar". 12. Selecciona la opción "Aceptar" del mensaje. 13. Selecciona la opción "Cerrar".	11. Muestra un mensaje de acción concluida satisfactoriamente. 14. Cierra la interfaz. 15. El caso de uso termina.
Prototipo de Interfaz	



Flujo Alterno al paso 9 "Datos incorrectos y/o campos vacíos"	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra símbolos de error al lado de los campos obligatorios vacíos o en los cuales se introdujeron datos no correctos. 2. Retorna al paso 5 del flujo normal de eventos.
Prototipo de Interfaz	



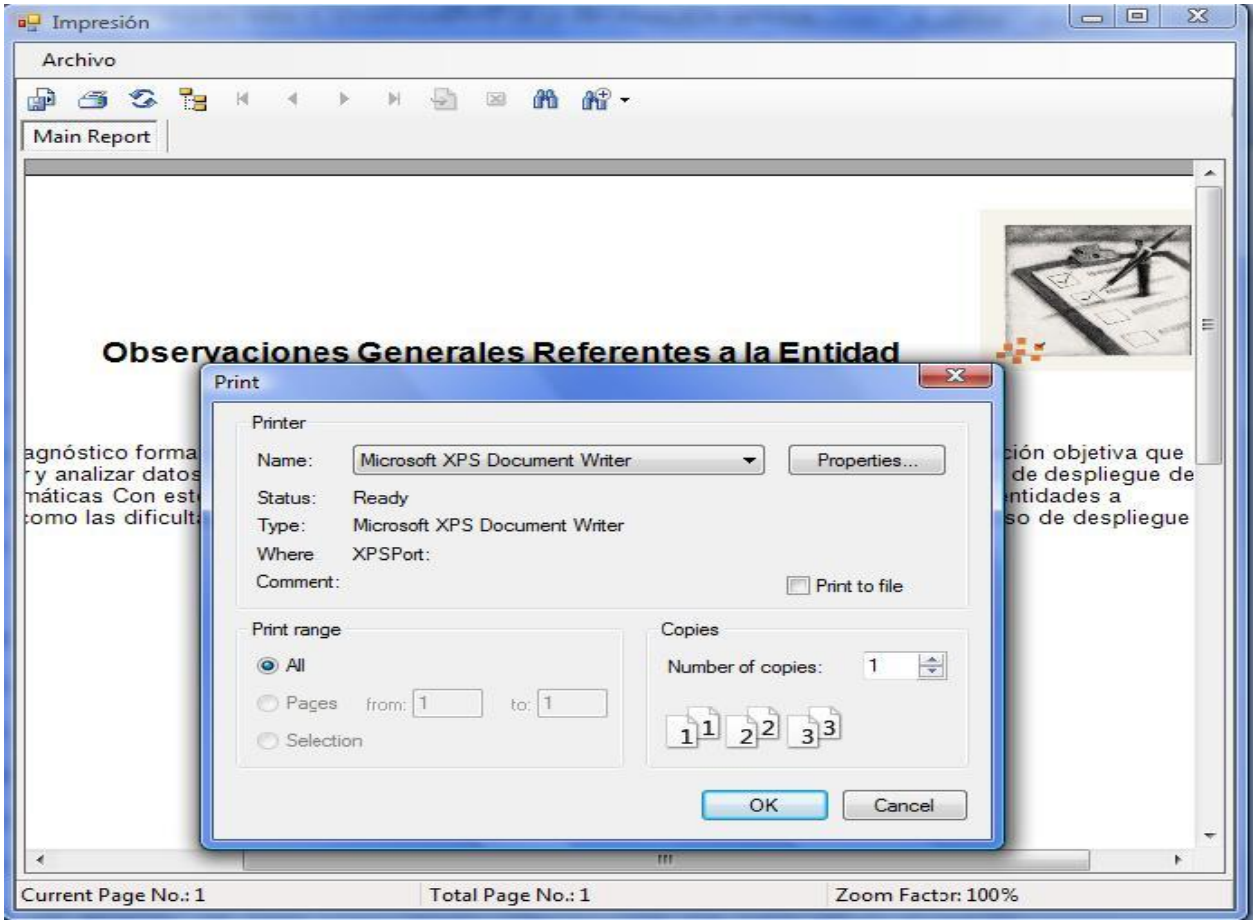
Flujo Alterno al paso 7“Operación cancelada”	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Cancelar”. 3. Selecciona la opción “Aceptar del mensaje”.	2. Muestra un mensaje de acción cancelada.
Post-condición	El sistema queda con nuevas planillas completadas y generadas en formato XML.

Tabla 17 Descripción del Caso de Uso del Sistema “CompletarPlanilla” de los Requisitos Opcionales.

Caso de Uso:	Imprimir.

Actor:	Procesador.
Resumen:	El caso de uso se inicia cuando el usuario Procesador se dispone a imprimir algún elemento seleccionado.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con el acceso a datos permitido. • El elemento a imprimir debe estar seleccionado.
Referencia	RF_32
Prioridad	Opcional
Complejidad	Medio
Nivel del caso de uso	Usuario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona el elemento a imprimir y luego la opción “Reporte”.	2. Muestra la interfaz para imprimir.
3. Selecciona la opción “Visualizar”.	4. Muestra el reporte.
5. Selecciona la opción “Imprimir”, configura la impresión y selecciona la opción “Aceptar”. 7. Selecciona la opción “Cerrar”.	6. Efectúa la impresión. 8. Cierra la interfaz. 9. El caso de uso termina.
Prototipo de Interfaz	



Flujo Alterno al paso 5“Operación cancelada”	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Cancelar”. 3. Selecciona la opción “Aceptar” del mensaje y regresa al paso 1 del flujo normal de eventos.	2. Muestra un mensaje de acción cancelada.

Post-condición	Impresión efectuada o cancelada.

Tabla 18 Descripción del Caso de Uso del Sistema “Imprimir” de los Requisitos Opcionales.

Caso de Uso:	AdicionarPlanilla.
Actor:	Procesador.
Resumen:	El caso de uso se inicia cuando el usuario Procesador se dispone a procesar la información recogida en las planillas.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con el acceso a datos permitido.
Referencias	RF_29, RF_30, CU Autenticar(incluido)
Prioridad	Opcional
Complejidad	Medio
Nivel del caso de uso	Usuario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción “Procesar Planilla”.	2. Muestra la interfaz para procesar la planilla, con las siguientes opciones. • Archivo.

	○ Cargar. ○ Cerrar. • Reporte.
3. Selecciona la opción “Cargar”.	4. Muestra la interfaz para cargar la planilla y pide nombre del archivo y dirección de la salva.
5. Introduce los datos y selecciona la opción “Aceptar”. 8. Selecciona la opción “Aceptar” del mensaje. 9. Selecciona la opción “Cerrar”.	6. Carga la planilla y almacena los datos en la base de datos indicada. 7. Muestra un mensaje de salva culminada satisfactoriamente. 10. Cierra la interfaz. 11. El caso de uso termina.
Prototipo de Interfaz 	
Flujo Alterno al paso 5“Operación cancelada”	
Acción del Actor	**Respuesta del Sistema**

1. Selecciona la opción “Cancelar”. 3. Selecciona la opción “Aceptar” del mensaje y regresa al paso 2 del flujo normal de eventos.	2. Muestra un mensaje de acción cancelada.
Flujo Alterno al paso 7“Error al salvar los datos”	
Acción del Actor	**Respuesta del Sistema**
	1. Muestra un mensaje de salva incompleta. Retorna al paso 2 del flujo normal de eventos.
Post-condición	• Datos guardados en la base de datos indicada.

Tabla 19 Descripción del Caso de Uso del Sistema “ProcesarPlanilla” de los Requisitos Opcionales.

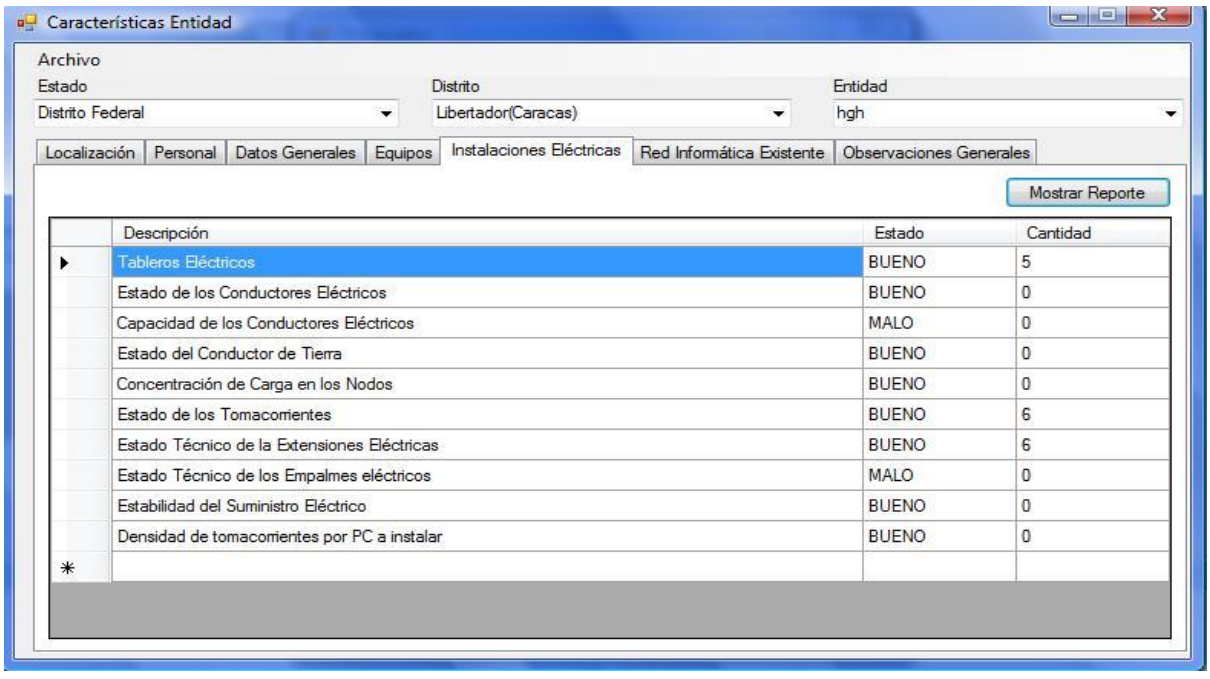
Caso de Uso:	Mostrar Reportes.
Actor:	Procesador.
Resumen:	El caso de uso se inicia cuando el usuario Procesador se dispone a visualizar algún reporte convenido.
Pre-condiciones:	• El sistema debe estar instalado y ejecutándose correctamente. • El actor debe estar autenticado con el acceso a datos permitido.
Referencias	RF_31, CU Autenticar(incluido), CU Imprimir (extendido)

Prioridad	Opcional
Complejidad	Simple
Nivel del caso de uso	Usuario

Flujo Normal de Eventos

Acción del Actor	Respuesta del Sistema
1. Selecciona la opción "Mostrar Reporte".	2. Muestra el reporte solicitado.
3. Selecciona la opción "Cerrar"	4. Cierra la interfaz.
	5. El caso de uso termina.

Prototipo de Interfaz

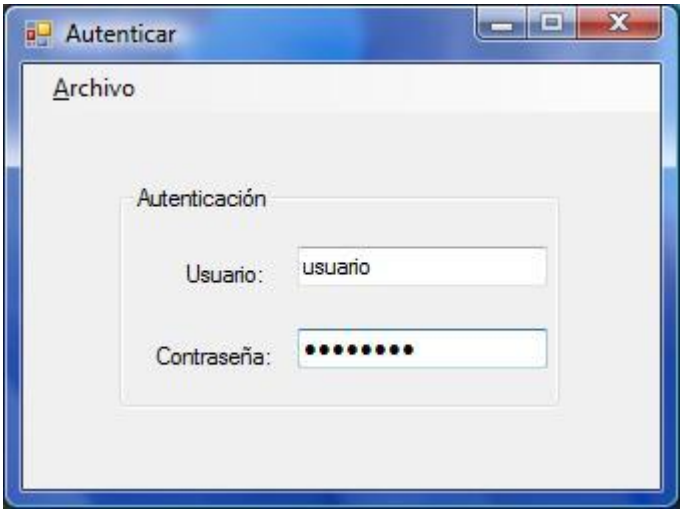


Flujo Alterno al paso 2"No hay reporte"

Acción del Actor	Respuesta del Sistema
2. Selecciona la opción “Aceptar” del mensaje.	1. Muestra un mensaje de no hay contenido para el reporte.
Post-condición	Reporte visualizado.

Tabla 20 Descripción del Caso de Uso del Sistema “Mostrar Reportes” de los Requisitos Opcionales.

Caso de Uso:	Autenticar.
Actor:	Procesador
Resumen:	El caso de uso se inicia cuando el procesador se dispone a autenticarse para obtener el acceso a datos necesario.
Pre-condición:	El sistema debe estar instalado y ejecutándose correctamente.
Referencia	RF_33
Prioridad	Opcional
Complejidad	Medio
Nivel del caso de uso	Usuario
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1. Accede a la opción “Autenticar”.	2. Muestra la interfaz de autenticación y pide nombre y contraseña.
3. Introduce los datos requeridos.	4. Valida las entradas. 5. Permite el acceso. 6. El caso de uso termina.
Prototipo de Interfaz 	
Flujo Alterno al paso 5“No hay acceso”	
Acción del Actor	**Respuesta del Sistema**
2. Selecciona la opción “Aceptar” del mensaje y retorna al paso 2 del flujo normal de eventos.	1. Muestra un mensaje de error de autenticación.

Post-condición	Acceso al sistema.

Tabla 21 Descripción del Caso de Uso del Sistema “Autenticar” de los Requisitos Opcionales.

Diagrama de Casos de Uso del Sistema.

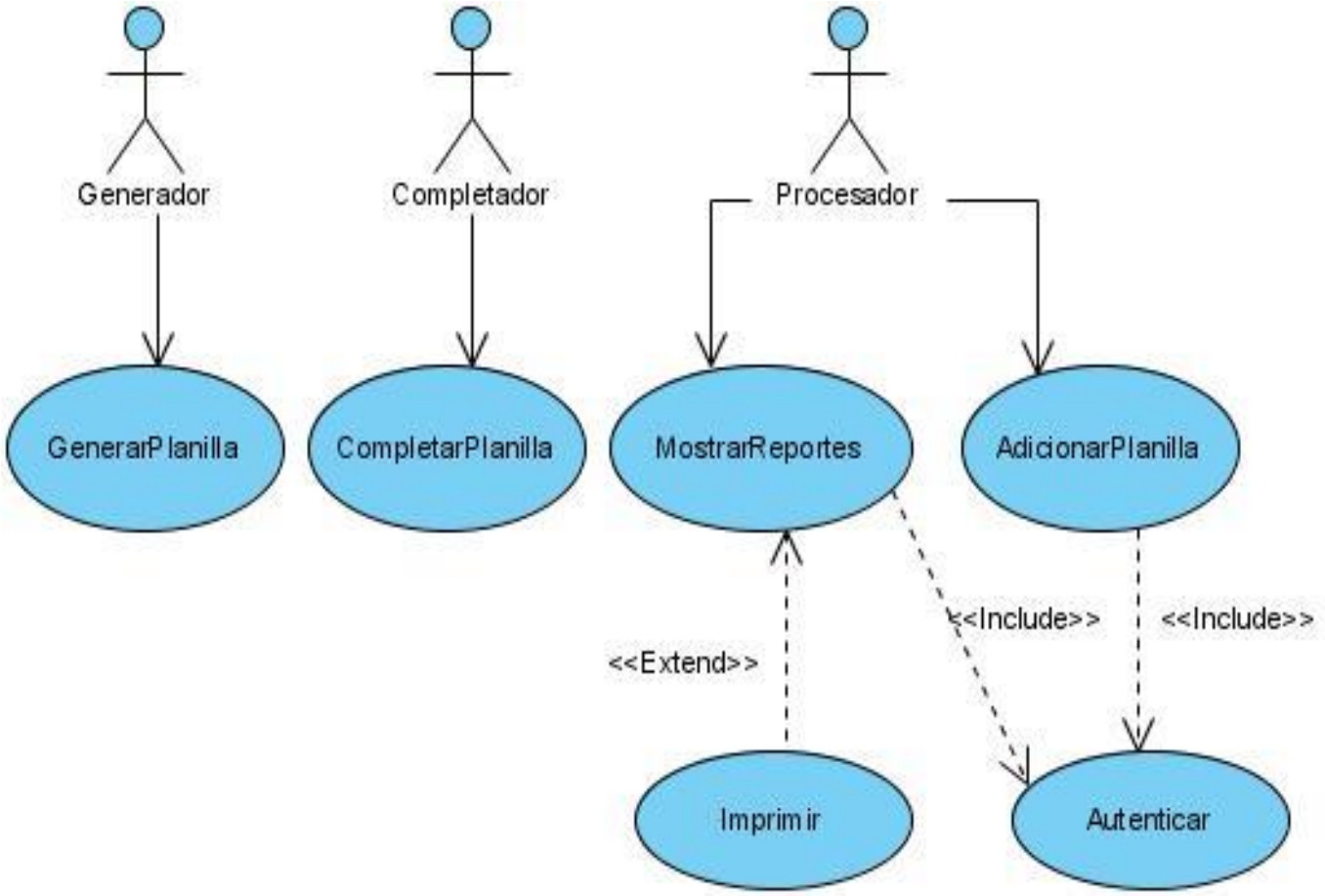


Figura 17 Diagrama de Casos de Uso del Sistema de los Requisitos Opcionales

Diagramas de Secuencia de los Casos de uso del Sistema perteneciente a los requisitos opcionales.

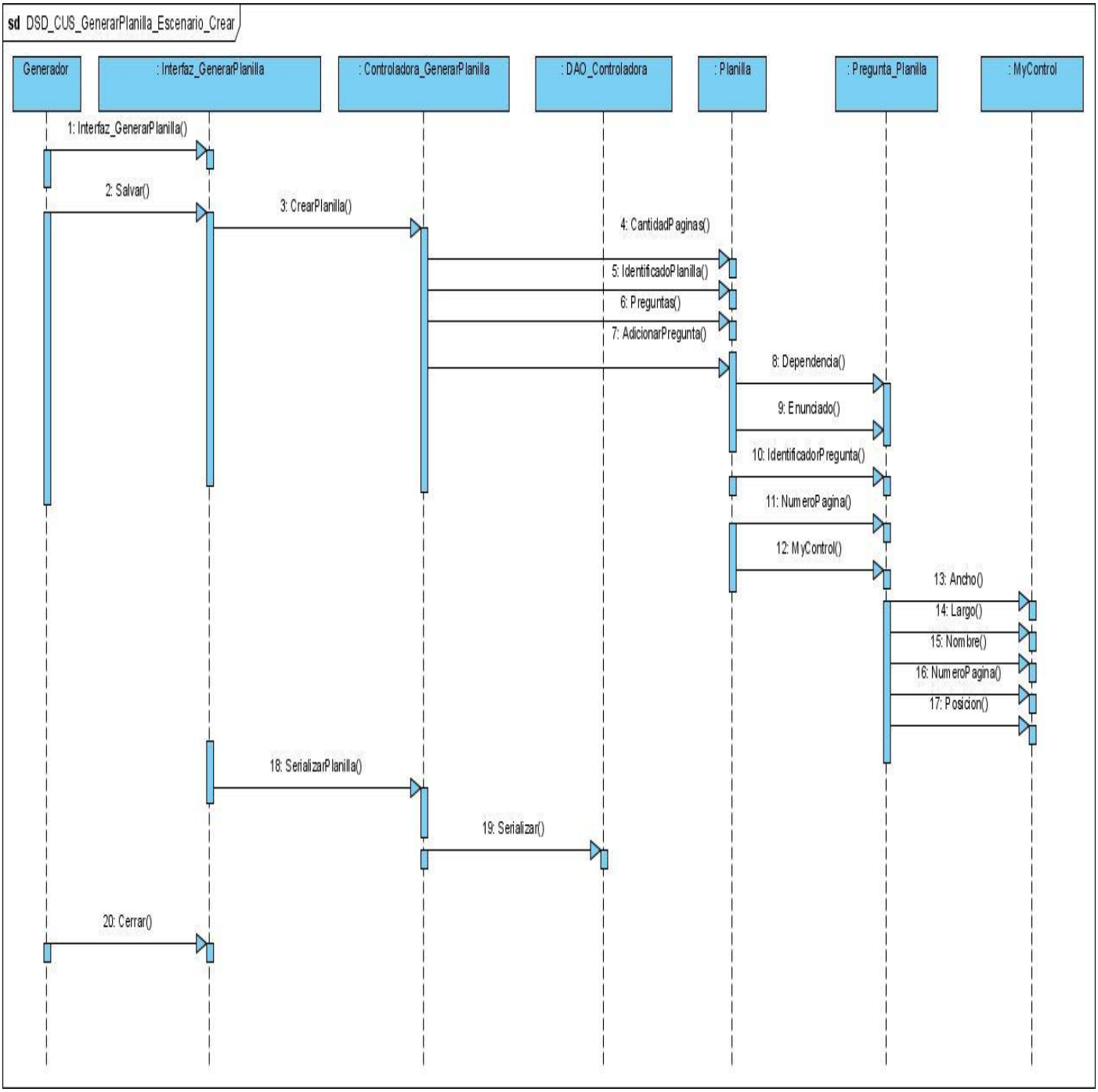


Figura 18. DSD_CUS_GenerarPlanilla_Escenario_Crear.

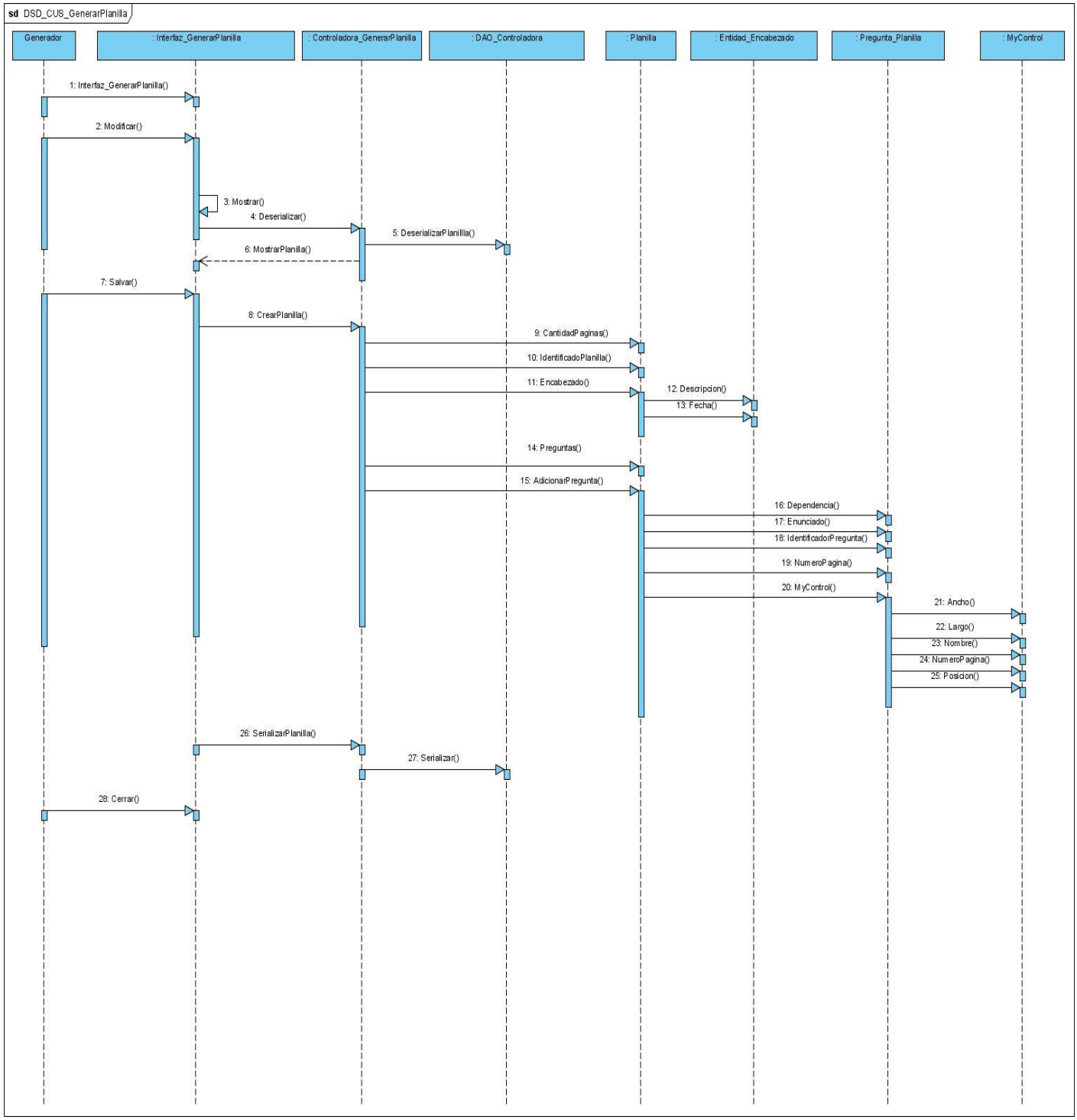


Figura 19.DSD_CUS_GenerarPlanilla_Escenario_Modificar

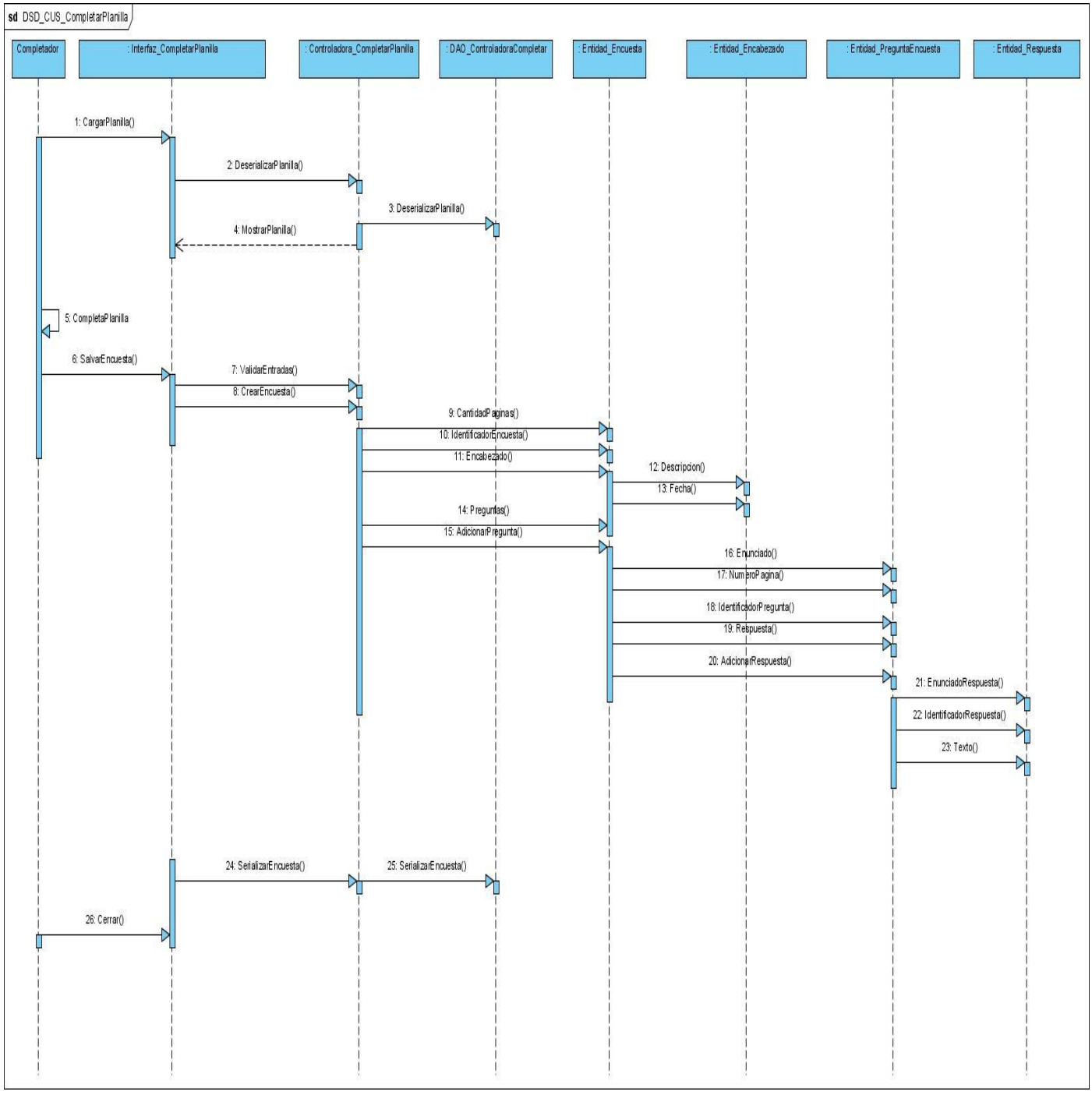


Figura 20. DSD_CUS_CompletarPlanilla.

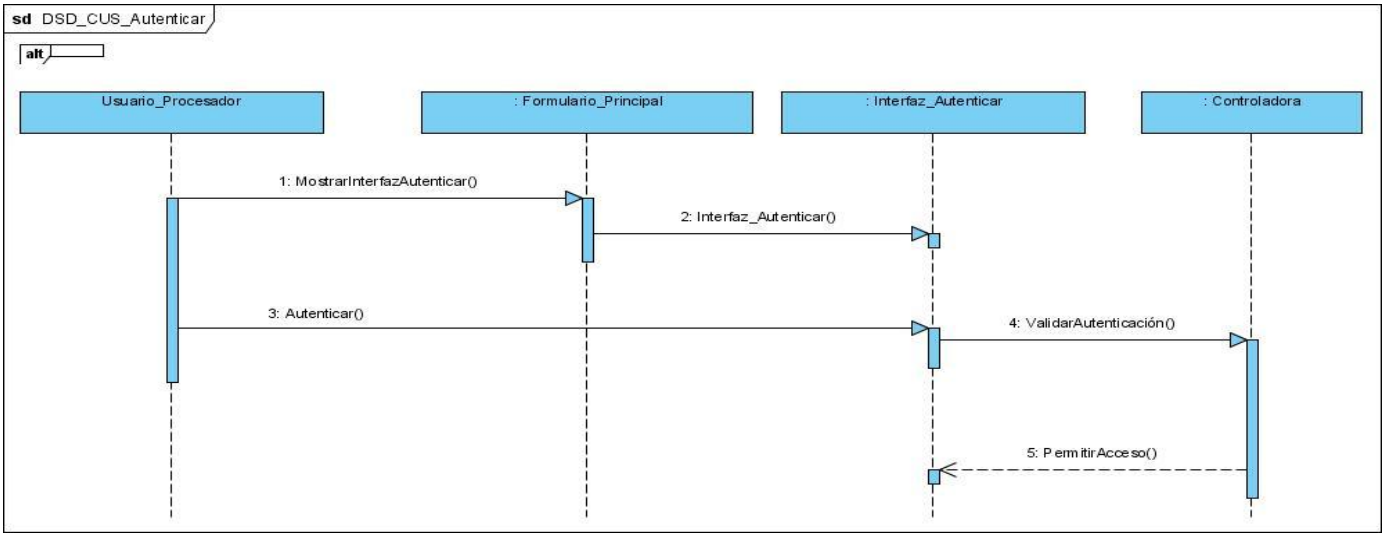


Figura 21. DSD_CUS_Autenticar.

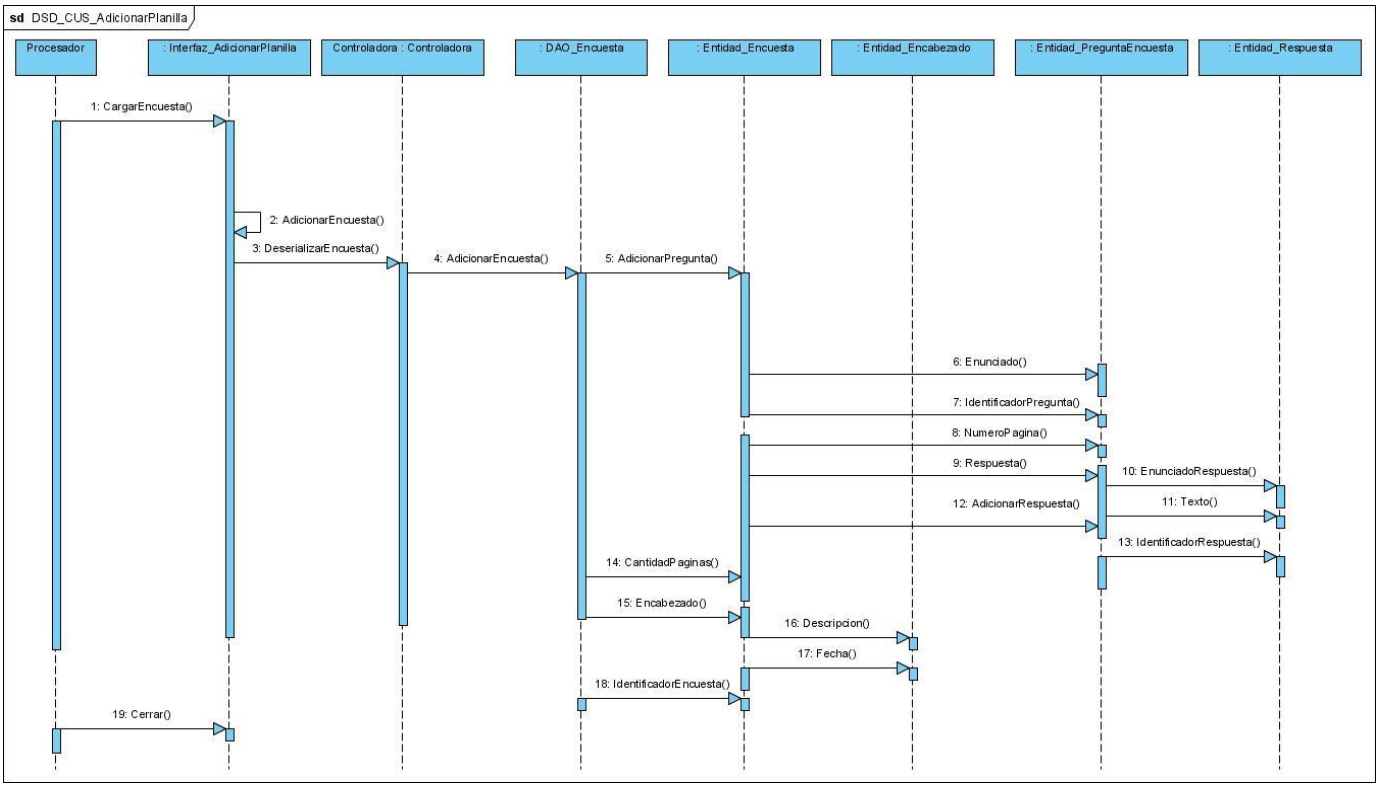


Figura 22. DSD_CUS_AdicionarPlanilla.

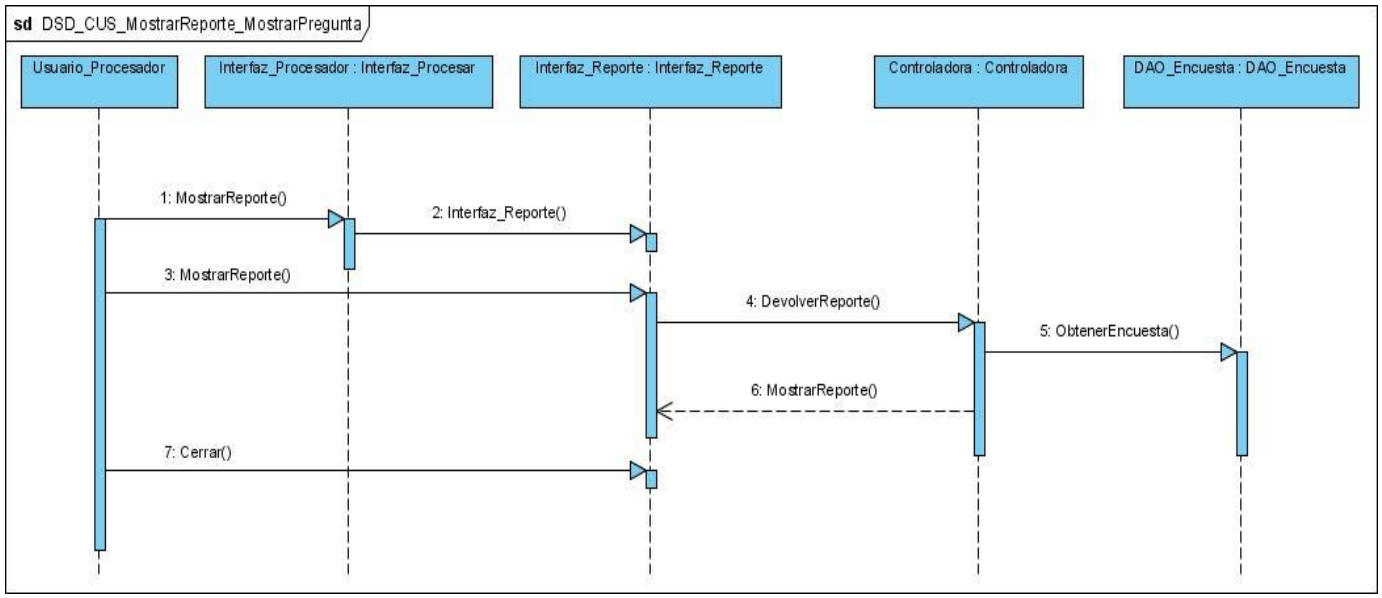


Figura 23. DSD_CUS_MostrarReporte.

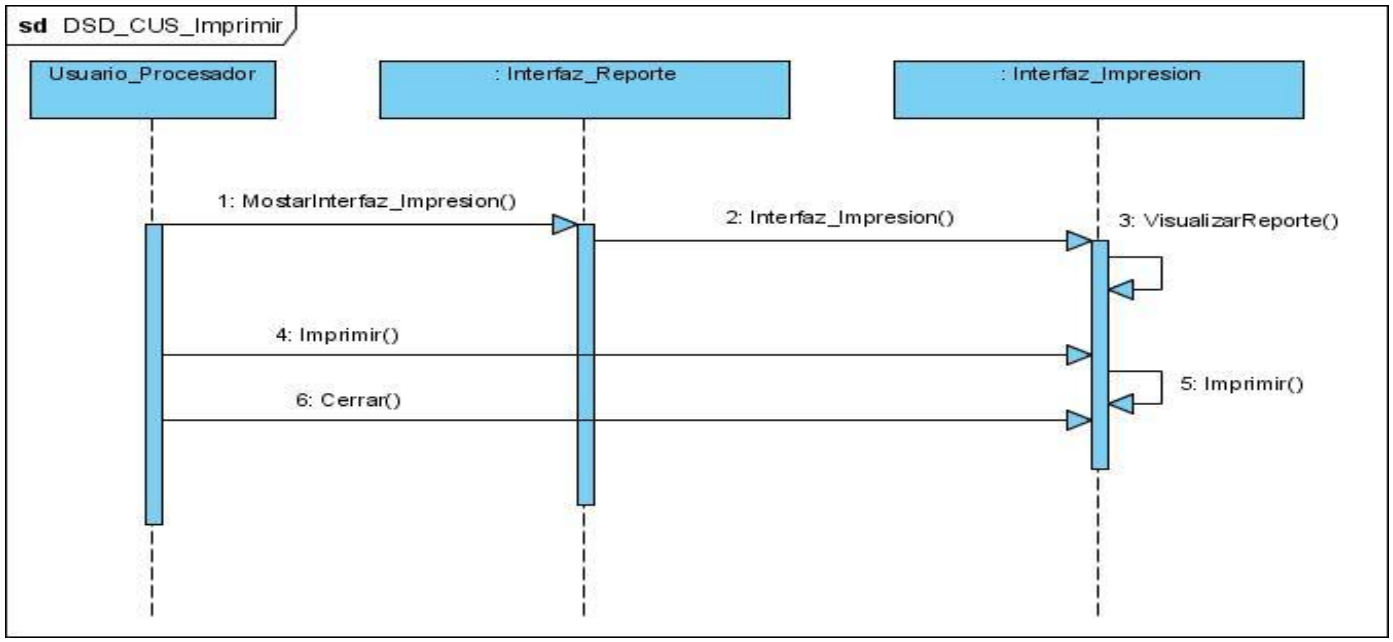


Figura 24. DSD_CUS_Imprimir.

Diagramas de clases de diseño de los Casos de Uso del Sistema de los requisitos opcionales.

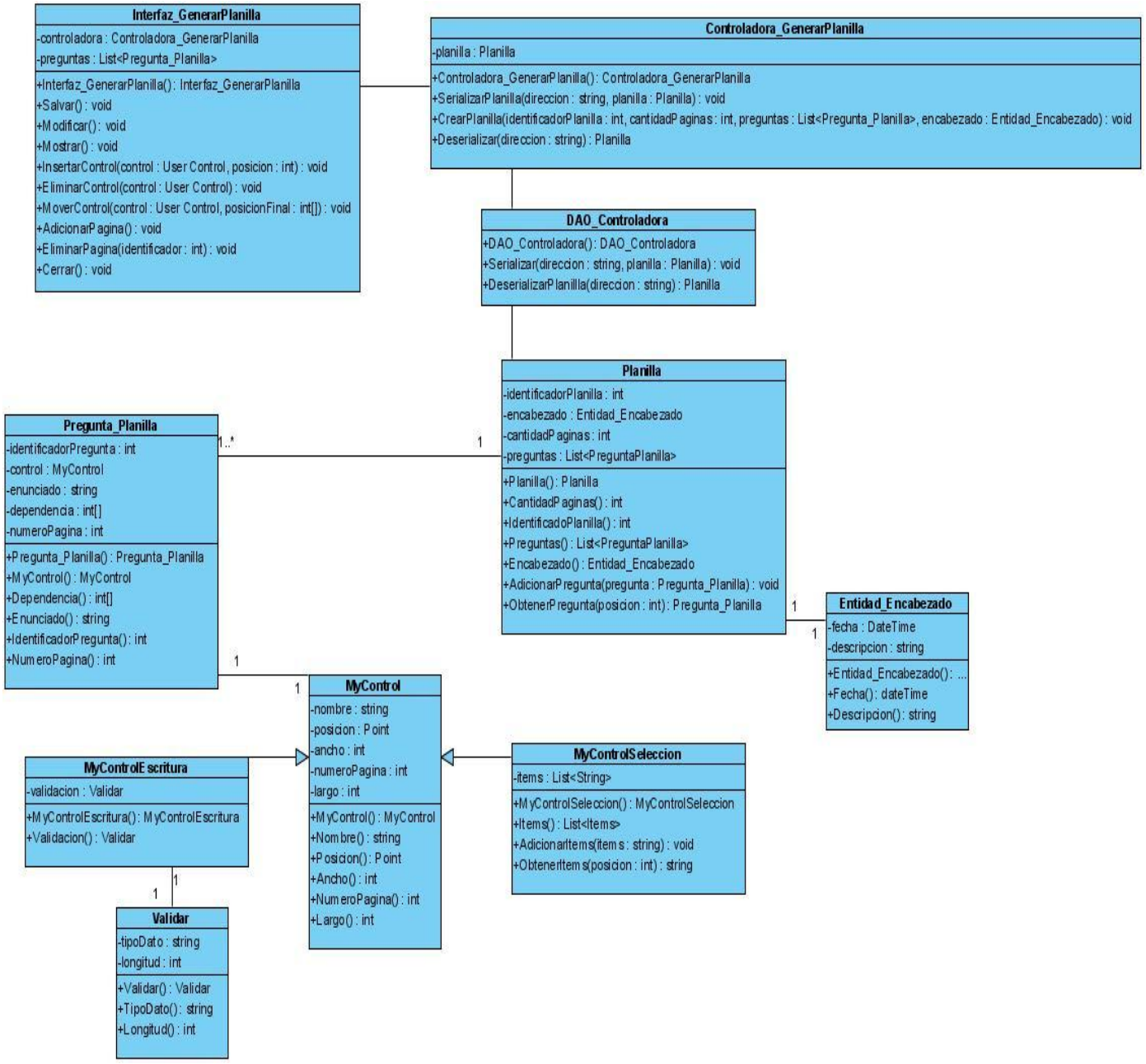


Figura 25.DCD_CUS_GenerarPlanilla.

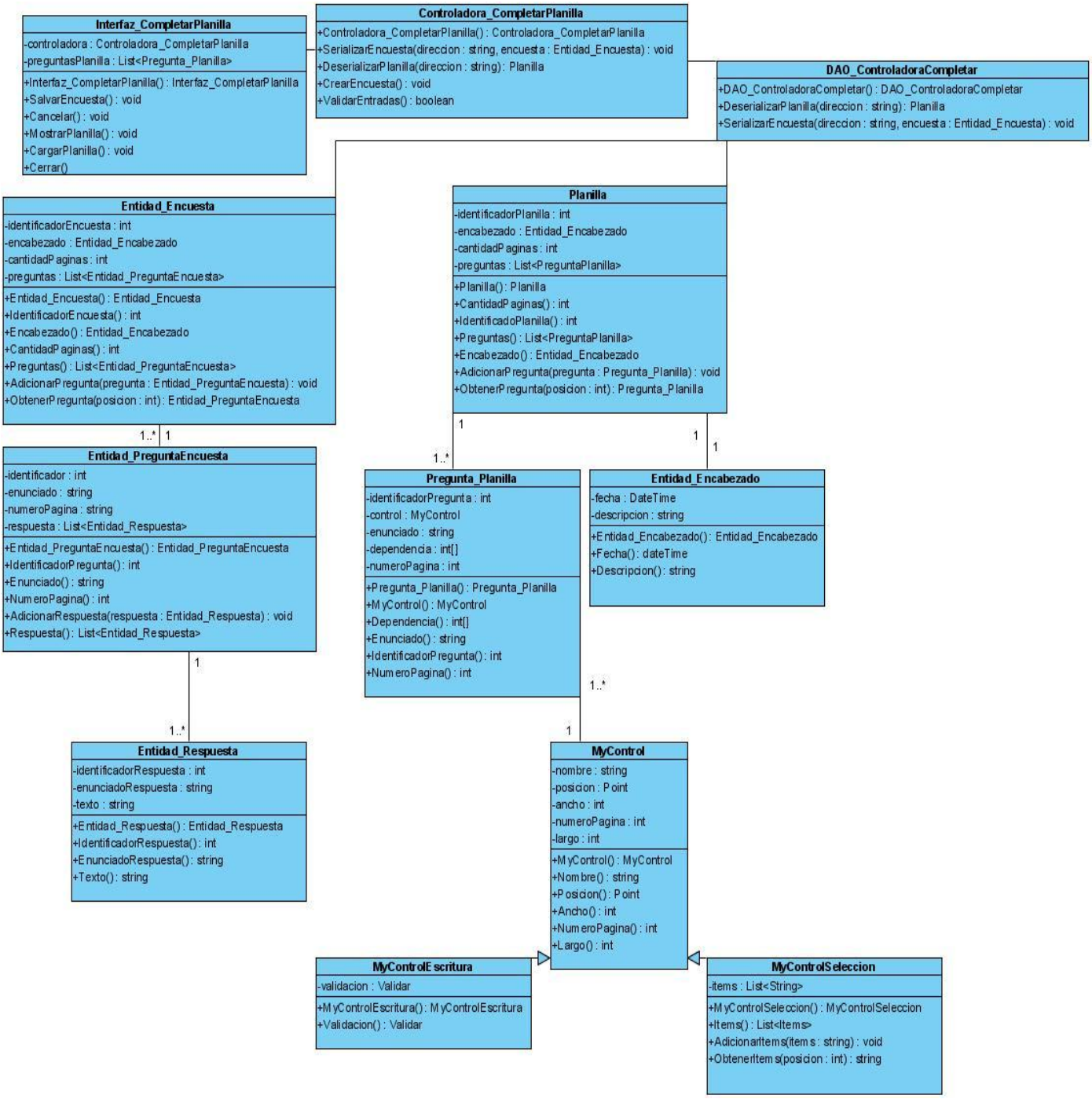


Figura 26.DCD_CUS_CompletarPlanilla.

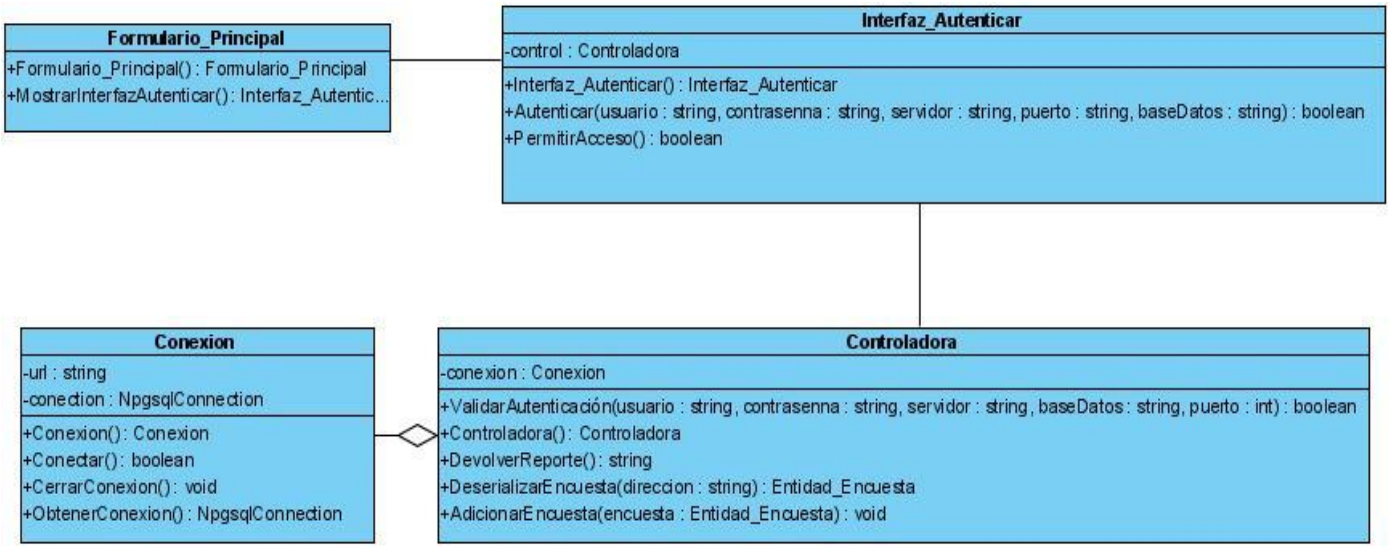


Figura 27.DCD_CUS_Autenticar

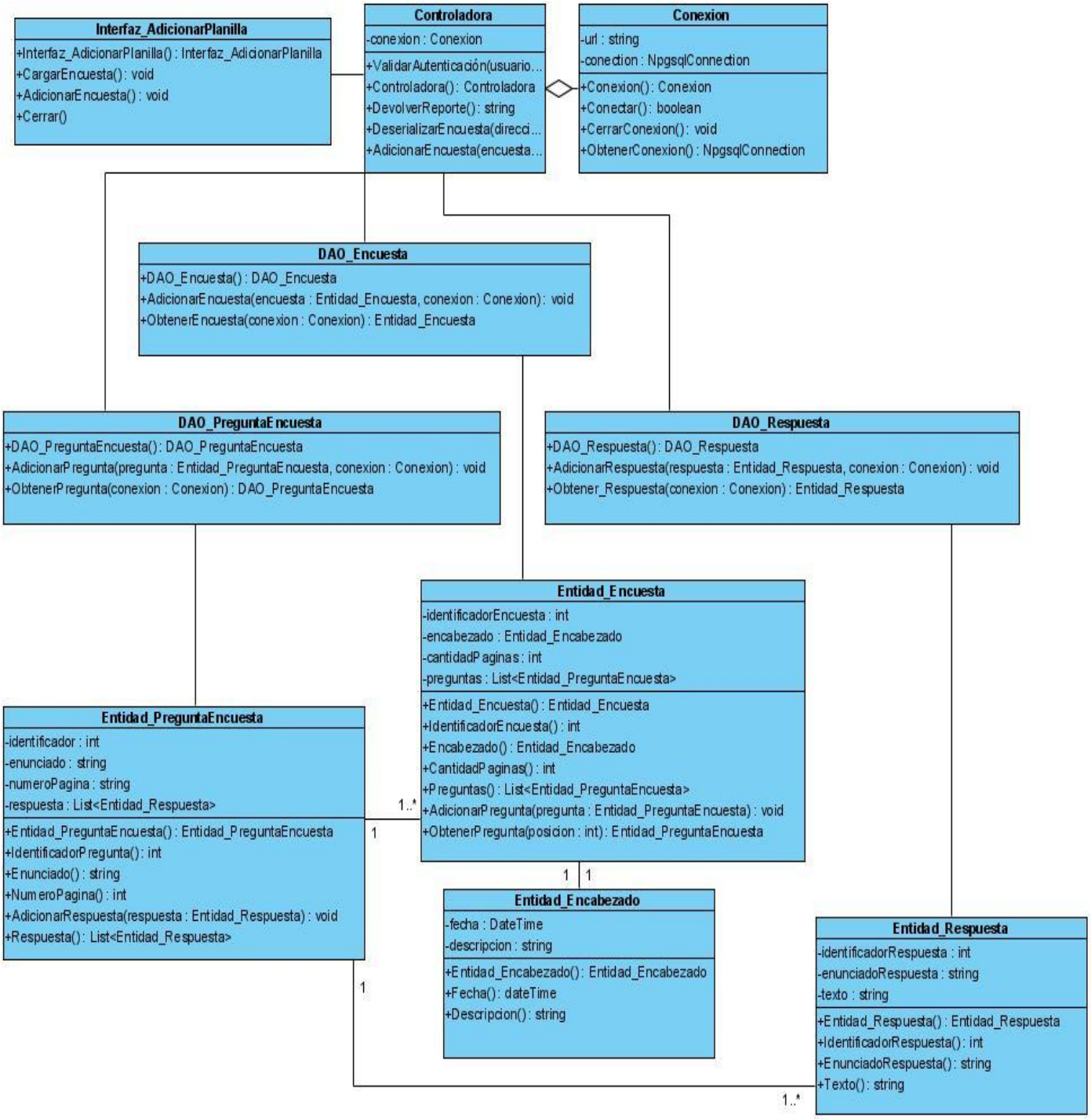


Figura 28.DCD_CUS_AdicionarPlanilla.

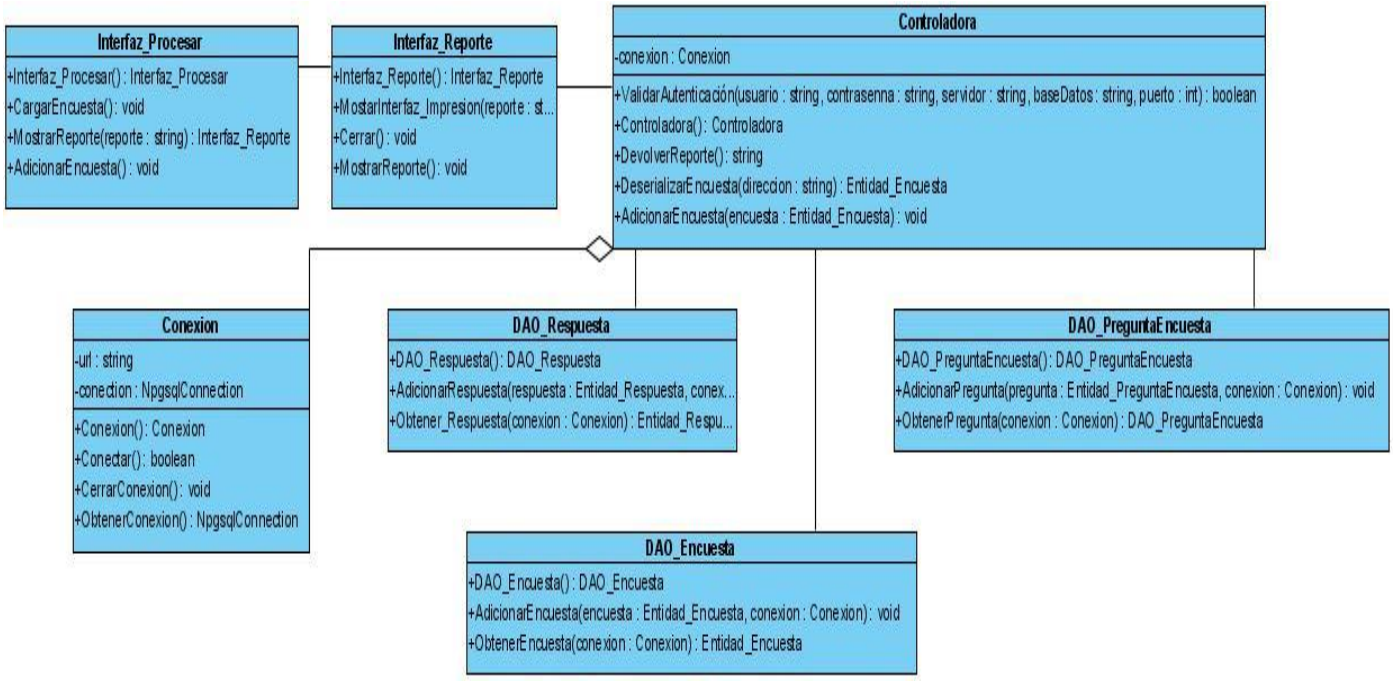


Figura 29.DCD_CUS_MostrarReporte.

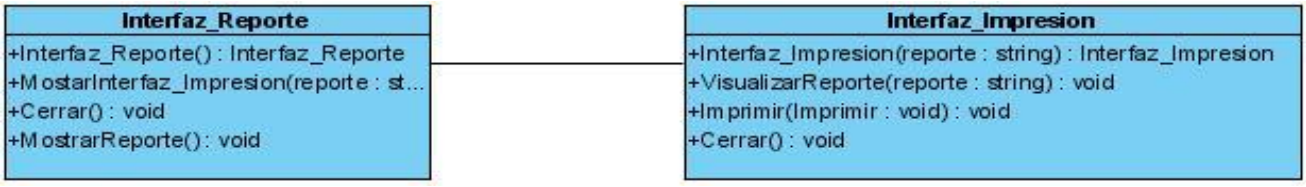


Figura 30.DCD_CUS_Imprimir.

Diagramas de Clases Persistentes

Para las funcionalidades opcionales se tuvo en cuenta, la necesidad de almacenar cualquier tipo de información en diferentes formatos. Se tiene una encuesta que puede tener una o muchas preguntas y un encabezado, por su parte cada pregunta puede tener una o muchas respuestas. Los diagrama de clases persistentes y el de entidad relación, muestran cada una de estas entidades con sus atributos, su tipo de datos y la relación entre las mismas.

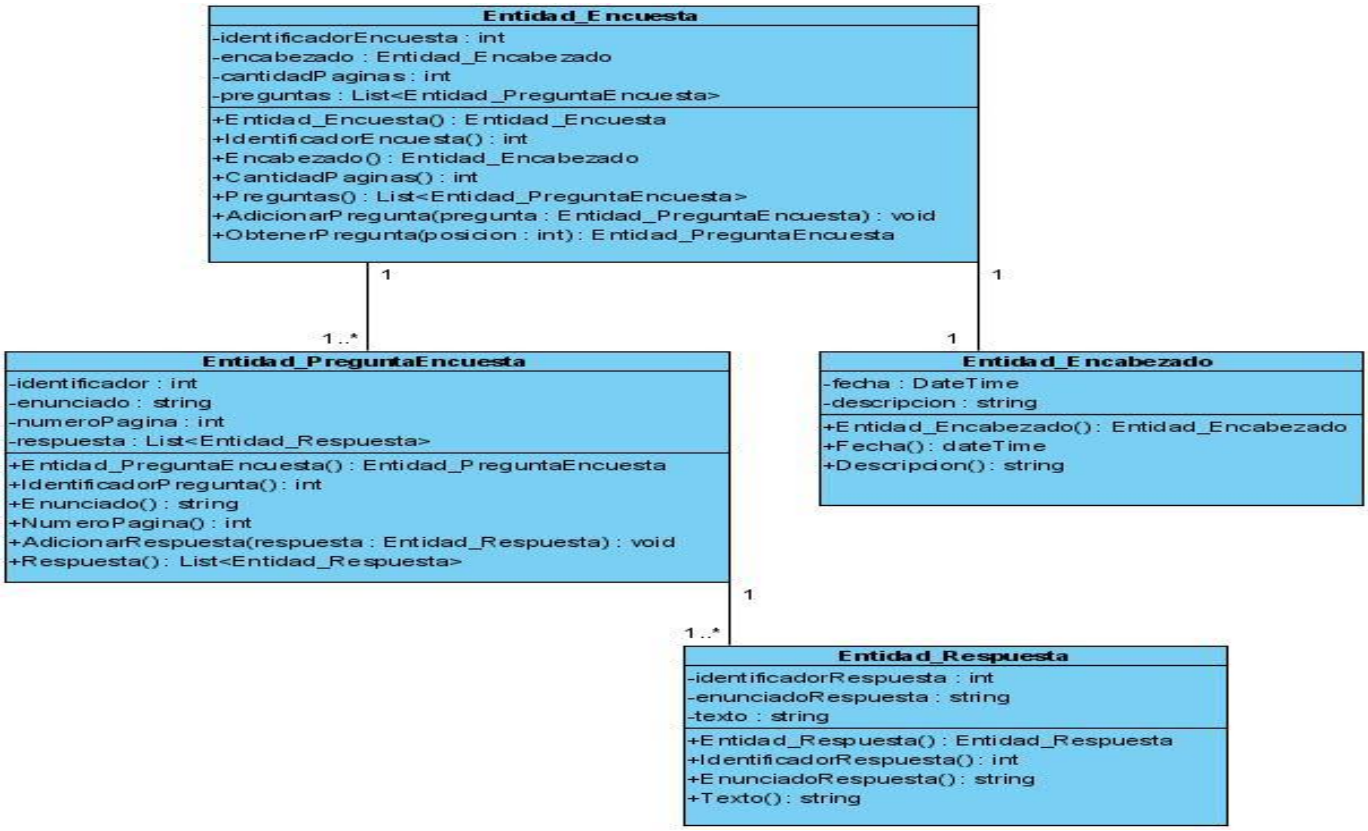


Figura 31. Diagrama de Clases Persistentes

Diagrama Entidad Relación

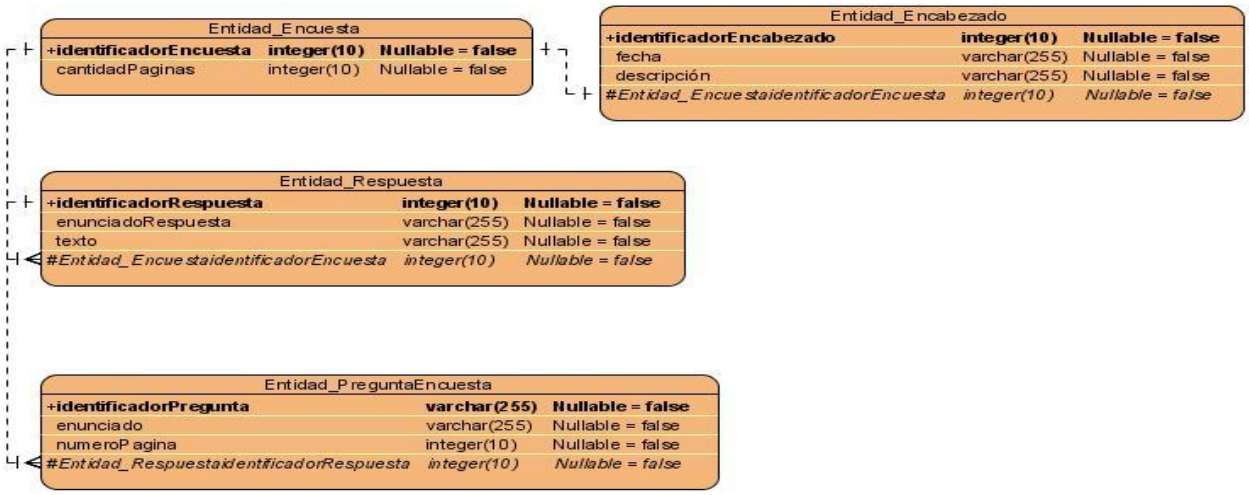


Figura 32. Diagrama Entidad Relación

3.2.7 Principios del diseño de la aplicación. [25]

Los 7 Principios del Diseño Universal, se centran en el diseño utilizable universalmente o por todos, pero hay que tener en cuenta que en el diseño intervienen otros aspectos, como el coste, la cultura en la que será usado, el ambiente; que tampoco pueden olvidarse. Estos Principios generales del diseño, son aplicables y de hecho se aplican en la arquitectura y la ingeniería.

En este apartado abarcaremos los principios utilizados, referente al sistema en cuestión.

1er Principio: Uso equiparable: El diseño es útil y vendible a personas con diversas capacidades.

Pautas para el Principio 1:

- ✓ Que proporcione las mismas maneras de uso para todos los usuarios: idénticas cuando es posible, equivalentes cuando no lo es.
- ✓ Que el diseño sea atractivo para todos los usuarios.

2do Principio: Uso flexible: El diseño se acomoda a un amplio rango de preferencias y habilidades individuales.

Pautas para el Principio 2:

- ✓ Que ofrezca posibilidades de elección en los métodos de uso.
- ✓ Que pueda accederse y usarse tanto con la mano derecha como con la izquierda.
- ✓ Que facilite al usuario la exactitud y precisión.
- ✓ Que se adapte al paso o ritmo del usuario.

3er Principio: Simple e intuitivo: El uso del diseño es fácil de entender, atendiendo a la experiencia, conocimientos, habilidades lingüísticas o grado de concentración actual del usuario.

Pautas para el Principio 3

- ✓ Que elimine la complejidad innecesaria.
- ✓ Que sea consistente con las expectativas e intuición del usuario.
- ✓ Que se acomode a un amplio rango de alfabetización y habilidades lingüísticas.
- ✓ Que dispense la información de manera consistente con su importancia.
- ✓ Que proporcione avisos eficaces y métodos de respuesta durante y tras la finalización de la tarea.

4to Principio: Información perceptible: El diseño comunica de manera eficaz la información necesaria para el usuario, atendiendo a las condiciones ambientales o a las capacidades sensoriales del usuario.

Pautas para el Principio 4

- ✓ Que use diferentes modos para presentar de manera redundante la información esencial (gráfica, verbal o táctilmente).
- ✓ Que proporcione contraste suficiente entre la información esencial y sus alrededores.
- ✓ Que amplíe la legibilidad de la información esencial.
- ✓ Que diferencie los elementos en formas que puedan ser descritas.

5to Principio: Con tolerancia al error: El diseño minimiza los riesgos y las consecuencias adversas de acciones involuntarias o accidentales.

Pautas para el Principio 5:

- ✓ Que disponga los elementos para minimizar los riesgos y errores: elementos más usados, más accesibles; y los elementos peligrosos eliminados, aislados o tapados.
- ✓ Que proporcione advertencias sobre peligros y errores.
- ✓ Que proporcione características seguras de interrupción.
- ✓ Que desaliente acciones inconscientes en tareas que requieren vigilancia.

6to Principio: Que exija poco esfuerzo físico: El diseño puede ser usado eficaz y confortablemente y con un mínimo de fatiga.

Pautas para el Principio 6:

- ✓ Que permita que el usuario mantenga una posición corporal neutra.
- ✓ Que utilice de manera razonable las fuerzas necesarias para operar.
- ✓ Que minimice las acciones repetitivas.
- ✓ Que minimice el esfuerzo físico continuado.

7mo Principio: Tamaño y espacio para el acceso y uso.

Que proporcione un tamaño y espacio apropiados para el acceso, alcance, manipulación y uso, atendiendo al tamaño del cuerpo, la postura o la movilidad del usuario.

Pautas para el Principio 7:

- ✓ Que proporcione una línea de visión clara hacia los elementos importantes tanto para un usuario sentado como de pie.
- ✓ Que el alcance de cualquier componente sea confortable para cualquier usuario sentado o de pie.
- ✓ Que se acomode a variaciones de tamaño de la mano o del agarre.

El diseño tiene que basarse en la mezcla de las ideas, deseos y necesidades del usuario y en los materiales de que dispone el programador, y en este caso se trata de usuarios que pueden ser expertos o inexpertos en el tema; algunos de ellos sin preparación en las cuestiones de la informática. Para ello, este sistema utiliza ciertos principios generales que garantizan la usabilidad en los diseños para estas aplicaciones:

- ✓ La interfaz no debe sobrecargar al usuario con complejidades innecesarias o distraerlo de su labor.
- ✓ Se debe dar al usuario un ambiente flexible para que pueda aprender rápidamente a usar la aplicación donde se ofrezcan posibilidades de elección en los métodos de uso, que facilite al usuario la exactitud y precisión, y se adapte al paso o ritmo del usuario.
- ✓ La interfaz debe ser consistente.
- ✓ Se debe considerar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, estas pérdidas de tiempo se pueden convertir en pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deberían tener las palabras claves del proceso.

En el desarrollo de la interfaz de esta aplicación se tuvieron en cuenta los principios anteriores de la siguiente manera:

- ✓ Se utilizaron elementos que resultaran agradables, logrando una mayor y adecuada comunicación entre la aplicación y el usuario.
- ✓ Se mantuvieron las alineaciones de textos así como la alineación de los botones.
- ✓ Se utilizaron elementos característicos para la descripción de acciones.

Se evidencia en la aplicación el cumplimiento de las exigencias fisiológicas con respecto al ser humano a la hora de recibir información digital, para que no se viera afectada su salud por un uso inadecuado, por

ejemplo: tamaño de letra, espaciamiento entre caracteres, tipografía, contraste figura_fondo de la información mayoritaria.

3.2.8 Tratamiento de Excepciones.

Es de vital importancia para lograr el correcto funcionamiento de cualquier sistema identificar y controlar los posibles errores que se pueden presentar a la hora de interactuar con el software. Para el correcto funcionamiento de la aplicación se tratarán estos errores de forma tal que las interacciones con la base de datos (inserción y reportes.) se realicen de forma correcta, para lograr esto se establecieron mecanismos de validación que comprueban la corrección de los datos a tratar; además en los formularios se insiste en que el usuario introduzca la menor cantidad posible de datos, aprovechando al máximo los campos calculables dentro del formulario, controles de selección; como: botones de opción (RadioButton), casillas de verificación (CheckBox), y listas de selección (ComboBox), entre otros. De esta forma el usuario selecciona entre opciones predefinidas lo que no da margen al error. En el caso de la entrada de datos por parte del usuario se utilizan las facilidades que brinda C# y .NET para la validaciones de los datos; de existir errores, se muestren mensajes que ilustren la incorrecta inserción, modificación o mala manipulación de datos en general con las posibilidades que brinda el lenguaje utilizado mostrando al usuario una información correcta y explicativa sobre el posible error cometido. Todos los mensajes de error se muestran en mensajes resaltados usando las facilidades de la tecnología, para la implementación de las funciones encargadas del control y validación de datos, y se implementa una segunda validación de los datos enviados por el usuario y recibidos por la base de datos para lograr minimizar a cero los posibles errores y lograr que el usuario interactúe con un sistema de calidad.

3.2.9 Estándar de codificación. [26]

Como dijo Martin Fowler “Cualquier tonto puede escribir código que entienden las computadoras. Los buenos programadores escriben código que entienden las personas.” El estándar de codificación centra en las guías de estilo para programación en lenguaje C# aunque puede ser utilizado en muchos otros lenguajes y entornos para establecer las convenciones a utilizar.

1. Organización de los ficheros
    1.1. Archivos fuente

Cada clase debe estar en un solo archivo, y un archivo no puede contener más de una clase.

1.2. Árbol de directorios y espacio de nombres

El espacio de nombres de los objetos se verá reflejado en el árbol de directorios del código fuente. Así, la clase Y10K.AyeAye.Switch, tendrá su código en Y10K/AyeAye/Switch.cs.

2. Indentación

2.1. Longitud de línea

No se mantendrá, siempre que sea posible, la longitud de la línea más allá de los 80 caracteres.

2.2. Espaciados de indentación

Dada la disparidad que hay sobre la cantidad de espacios de indentación, se utilizarán tabuladores, los cuales pueden usarse con muchos editores para representarse con distintas cantidades de espacios y, sobretodo, se convierten en una pulsación por indentación.

3. Declaraciones

3.1. Número de declaraciones por línea

Por definición, nunca se utilizará una misma línea para más de una definición, la cual cosa facilita los comentarios relativos al elemento declarado.

3.2. Declaración de clases e interfaces: Para definir las clases e interfaces, se seguirán las siguientes reglas:

No se usará ningún espacio entre el nombre de un método y el paréntesis de abertura de la lista de parámetros. La llave de abertura que contiene el código se escribe sola en la línea siguiente a la definición del prototipo. Igualmente, la llave de cierre correspondiente se escribe sola en la última línea.

4. Sentencias

4.1. Sentencias simples

Cada línea contendrá no más de una sentencia, aunque, según lo explicado en el apartado 3, una sentencia puede estar en más de una línea.

4.2. Sentencias de retorno

La sentencia return no utiliza paréntesis.

4.3. Sentencia de selección básica(if/if...else/if...else if...else)

Las llaves de inicio de un bloque de código se ponen en la siguiente línea de la sentencia if, else o else if. Las llaves de cierre van en líneas independientes.

4.4. Sentencias de bucle for o foreach

De forma similar a las sentencias de selección, la llave de abertura de bloque se colocará en la siguiente línea de la sentencia de definición del for, y la de cierre en una línea independiente.

4.5. Sentencias de selección múltiple (switch)

Respecto a lo que concierne a las llaves, se colocarán como en el resto de los casos. Respecto al código y la línea break, éstas irán con una indentación más que la línea case correspondiente, que irá con una indentación más que la línea de switch.

4.6. Sentencias de captura y tratamiento de excepciones (try/catch/finally)

El tratamiento de las llaves de bloque será como en el resto de las sentencias.

5. Espacios en blanco

5.1. Separaciones entre términos

En una lista de parámetros, se pondrán espacios tras las comas, pero nunca tras o antes de los paréntesis. En las asignaciones, se pondrán espacios antes y después del =. En las expresiones, se utilizarán espacios sólo cuando sean estrictamente necesarios y para separar las distintas expresiones.

6. Nomenclatura

6.1. Nomenclatura para las clases

El nombre de una clase debe componerse de sustantivos. Se utilizará capitalización Pascal. No se utilizará ningún prefijo de clase.

6.2. Nomenclatura para parámetros y atributos normales:

Se utilizarán nombres descriptivos, destacando el significado antes que la tipología del parámetro. En este caso se utiliza capitalización Camel.

6.3. Nomenclatura para variables:

Se utilizará Camel. Cuando se utilicen contadores triviales, se utilizarán nombres como i, j, k, m, n,...

6.4. Nomenclatura para métodos

Los métodos se nombrarán con verbos, en Pascal.

7. Prácticas de programación

7.1. Visibilidad

Preferentemente se codificarán los atributos de clase y de instancia como privadas, de modo que existirán métodos de acceso.

8. Comentarios

Se utilizará solamente el formato //, nunca se utilizará el /*...*/. Cuando haya que comentar un bloque de líneas, se utilizarán tantos // al principio de línea como convenga.

3.3 Validación.

3.3.1 Métricas de verificación del diseño.

En la etapa del diseño la calidad aumenta en la medida en que se realiza una alta especificación de los procesos y se propone una estrecha tolerancia a la modificación, estableciendo los métodos correctivos a las desviaciones ocurridas. [27]

Existen varios tipos de métricas orientadas a objetos, las cuales son las que se acercan al interés del presente trabajo, entre ellas se encuentran:

- Métricas Orientadas a las Operaciones.
- Métricas para Pruebas Orientadas a Objetos.
- Métricas sobre el tamaño del software.
- Métricas Orientadas a Clases.

En este último caso existen varios tipos de series aplicadas en este sentido, se pueden citar algunas como las de Chidamber y Kemner (CK), Lorentz y Kidd (LK), Li y Henry, Henderson-Sellers. En el diseño desarrollado se aplicaron algunas de las métricas propuestas por las series CK y LK, por ser las más reconocidas y las que más se ajustan al diseño desarrollado.

Serie CK

- Métodos ponderados por Clase (MPC).
- Árbol de Profundidad de Herencia (APH)

- Número de Descendientes (NDD)
- Acoplamiento entre Clases Objeto (ACO)
- Respuesta Para una Clase (RPC)
- Carencia de Cohesión de los Métodos (CCM)

Serie LK

- Tamaño de Clase (TC)
- Número de Operaciones redefinidas por una subclase (NOR)
- Número de Operaciones Añadidas por una subclase (NOA)
- Índice de Especialización (IE)

Para la verificación del diseño del software se escogieron algunas de estás métricas de la primera serie (CK):

Árbol de Profundidad de Herencia (APH): Se define como la máxima longitud del nodo a la raíz del árbol. [28]

Número de Descendientes (NDD): Las subclases inmediatamente subordinadas a una clase en la jerarquía de clases se denominan sus descendientes. [28]

De la segunda serie (LK):

Tamaño de Clases (TC): Las clases pueden medirse determinando el total de operaciones, tanto heredadas como privadas de la instancia que se encapsulan dentro de una clase, más el total de atributos, atributos tanto heredados como privados de la instancia encapsulados por la clases. [29]

A continuación se explicará el análisis de resultados del diseño planteado anteriormente por cada una de las métricas explicadas, haciendo énfasis en las clases de diseño del Componente Diagnóstico puesto que son las clases de mayor peso en la aplicación, no haciendo un estudio para el segundo componente (Procesador) puesto que posee aproximadamente la misma lógica del negocio del primero.

Métricas de verificación del diseño para el Componente Diagnóstico.

Árbol de profundidad de Herencia

Altos niveles de herencia indican objetos complejos, los cuales pueden ser difíciles de testear y mantener, esto es debido a que cuanto más profunda se encuentra una clase en la jerarquía, mayor será la probabilidad de heredar un mayor número de métodos. Por el lado positivo, los valores de APH grandes implican un gran número de métodos que se reutilizarán. [28]

Según la bibliografía consultada, varios autores, entre los que se encuentran Lorenz y Kidd, definen como una profundidad negativa, el nivel 6.

Resultado: Esta métrica arrojó como resultado que el nivel más alto de herencia entre las clases del diseño es 2, esto determina que el diseño no es complejo, es fácil de testear, adaptar y modificar.

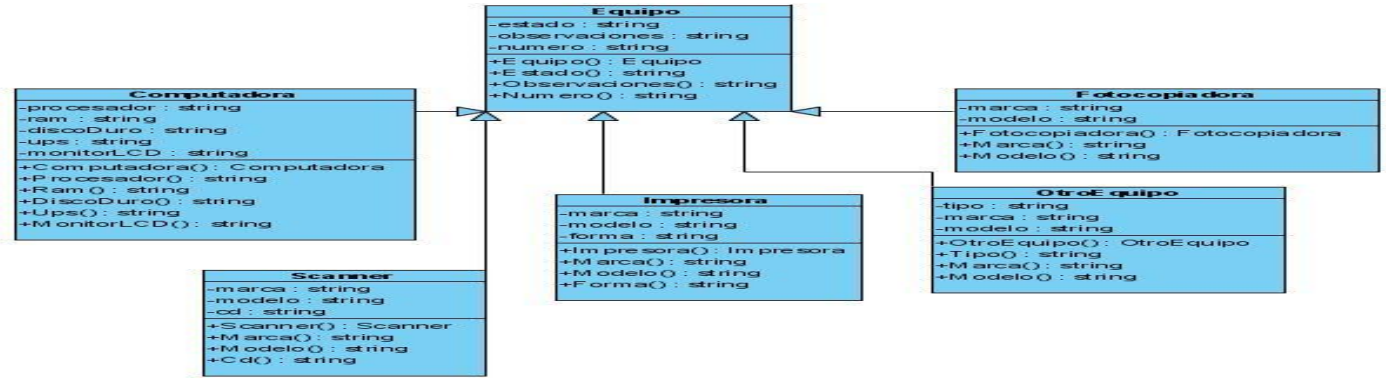


Figura 33. APH_Clases de Negocio del Componente Diagnóstico

Número de Descendientes (NDD)

Un mayor número de hijos representa una mayor reutilización de código, pero presenta inconvenientes como una mayor probabilidad de usar incorrectamente la herencia, creando abstracciones erróneas, es decir existe una posibilidad de que algunos descendientes no sean miembros, realmente apropiados, de la clase predecesora. Además, dificulta la modificación de esta clase, ya que afecta a todos sus hijos. Todo esto trae consigo que se requiera mayor número de recursos para testear.

Resultado: Esta métrica arrojó como resultado que el número máximo de descendientes es 5. Este es un resultado positivo, pues permite la reutilización, favorece la abstracción, permite la modificación moderada y facilita las pruebas en cuestión.

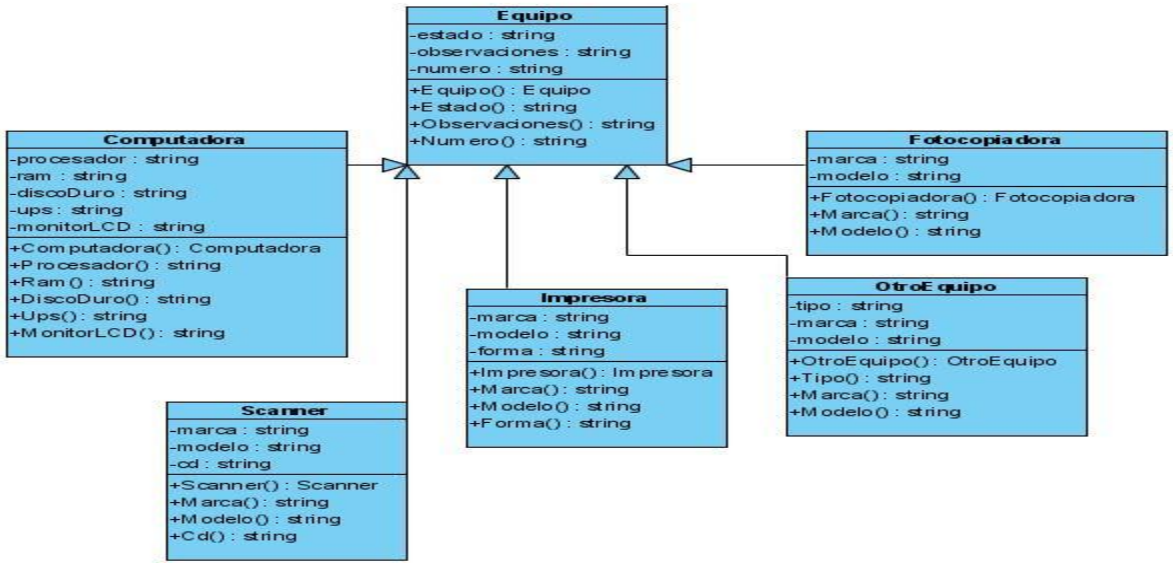


Figura 34. NDD_Clases de Negocio del Componente Diagnóstico

Tamaño de Clases (TC)

Los valores grandes para esta métrica, indican que la clase posee bastante responsabilidad y complejidad, esto reduce la reutilización de esta clase y dificulta las pruebas sobre la misma.

Para el funcionamiento de esta métrica se pueden calcular los promedios para el número de atributos y operaciones de clase.

Resultado: Teniendo en cuenta las medidas de referencia en lo que respecta al número de operaciones y/o atributos de las clases se establece que un tamaño de clase pequeño es aquel que tiene un valor menor o igual que 20, un tamaño de clase medio es aquel cuyos valores exceden a 20 y son menores o incluyen a 30 y un tamaño de clase grande es aquel que es mayor que este último valor (30). Después de aplicar esta métrica al diseño de la solución propuesta se llegó a las siguientes conclusiones.

Número	Clases	No. Atributos	No. Operaciones
1	ExistenciasEquiposRed	4	6
2	InstalacionesElectricas	4	6

3	F_01_05_Negocio	1	3
4	Resumen	9	11
5	CantEquipo	3	5
6	Equipo	3	5
7	Computadora	8	12
8	Scanner	6	10
9	Impresora	6	10
10	OtroEquipo	6	10
11	Fotocopiadora	5	11
12	Funciones	6	8
13	TipoDoc	1	3
14	Entidad	25	28
15	DatosGenerales	14	16
16	AdquisicionEquipo	3	5
17	ConocimientoEC	4	6
18	DatosEmpleado	5	7
19	TipoEmpleado	3	5
20	RecomendacionesExi	4	6
21	RecomendacionesInst	4	6

22	RiesgosEquip	1	3

Tabla 22. Diagnóstico. Resultados del TC para las clases de Negocio del Componente

Tomando como referencia las clases de la lógica de negocio de dicho componente, se cuenta con un total de 22 clases, con un promedio de cantidad de atributos aproximadamente de 5 y un promedio de cantidad de operaciones aproximadamente de 8.

Los valores de tamaño (Atributos + Operaciones) quedan distribuidos de la siguiente manera:

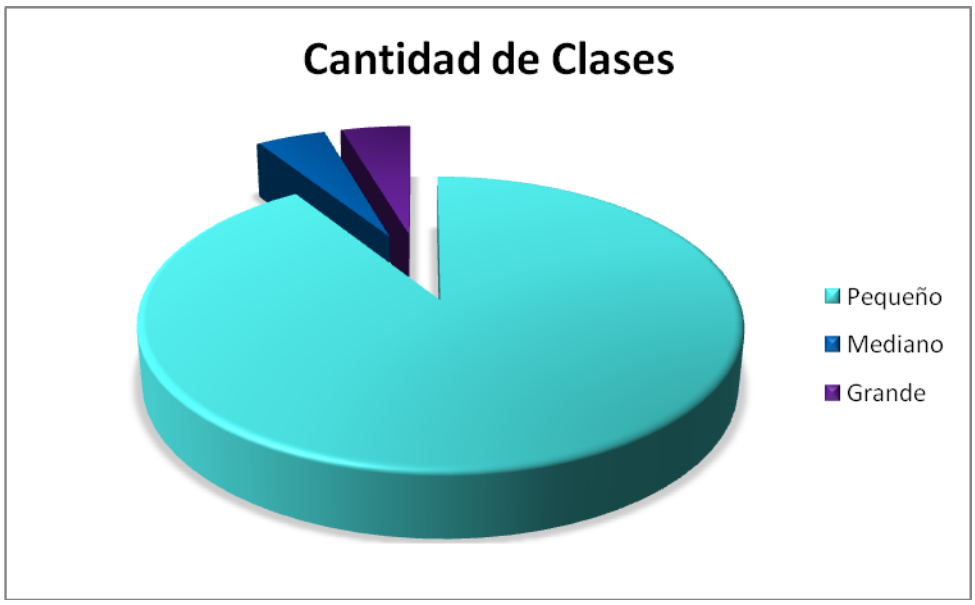


Figura 35. Tamaño de las Clases

Como se puede observar en las tablas que muestran los resultados, de 22 clases que contiene la solución hay una sola clase de tamaño grande, una de tamaño mediano y el resto son clases pequeñas, lo cual brinda un resultado positivo, debido a que la mayoría de las clases son pequeñas, proporcionando un diseño de baja responsabilidad y complejidad, reutilizable y fácil de probar.

3.3.2 Pruebas de Caja Negra.

El equipo de calidad del proyecto SAREN encargado de realizar las pruebas pertinentes al presente sistema en cuestión, tuvo en cuenta para validar la funcionalidad y calidad del mismo el uso de las pruebas unitarias de validación: caja negra, las cuales se realizan sobre las interfaces del sistema para demostrar que cada funcionalidad es operativa y encontrar de esta forma, funciones incorrectas o inexistentes, errores relativos a las interfaces; en estructuras de datos o en acceso a base de datos externas, debidos al rendimiento o de inicialización y terminación. Las pruebas de caja negra se centran en los requisitos funcionales del software

Entre las técnicas empleadas por el equipo se encuentran:

Partición equivalente: Divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. [28]

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. La prueba de partición equivalente se basa en evaluar las clases de equivalencia para una condición de entrada.

Análisis de valores límite: Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites, como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límites.

El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, este lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el mismo obtiene casos de prueba también para el campo de salida. [28]

Resultados: Para cada uno de los componentes se diseñaron 6 casos de prueba, obteniendo resultados satisfactorios en cada uno de los mismos, razón que avala la aceptación funcional del sistema por parte del equipo de calidad del proyecto SAREN. (Ver Anexo 12).

3.4 Conclusiones del Capítulo.

Con la culminación de este capítulo se ha logrado completar al 100% las funcionalidades previstas en el Product Backlog, cumpliendo de esta forma con las necesidades del cliente, lo cual fue validado a través de pruebas pertinentes, en este apartado. Por la importancia que trae consigo, al ser una de las mejores prácticas para el desarrollo de software, se describe y modela cada Sprint desarrollado, tomando para esto los artefactos generados por cada ciclo y de acuerdo a las características de cada uno, entre ellos se encuentran los diagramas de las clases de diseño, clases persistentes y modelo entidad_relación, teniendo un trato especial para los requisitos opcionales, debido a que su modelación servirá de base a una futura implementación de los mismos.

Como conclusión parcial se puede destacar que el sistema obtenido está apto para ser probado en un entorno real basado en los resultados satisfactorios de las pruebas de caja negra realizadas. Se analizaron de manera general las características de las métricas para verificar el diseño de sistemas informáticos. Se aplicaron algunas de las métricas de las series CK y LK; unas de las más conocidas internacionalmente. La aplicación de las mismas arrojó como resultado que el diseño de la solución presenta un bajo acoplamiento entre las clases, permite la reutilización, mantiene la abstracción representada por la clase predecesora, presenta una adecuada jerarquía, es un diseño de fácil mantenimiento, y el software será fácil de probar y modificar sin afectar el tiempo de los cambios.

CONCLUSIONES

El desarrollo de este trabajo de pregrado arriba a las siguientes conclusiones:

- Los conceptos “Proceso de Diagnóstico” y “Sistema de Gestión de la Información” están estrechamente vinculados, con el fin de apoyarse el uno en las ventajas y peculiaridades del otro.
- Un Sistema de Gestión de la Información es un elemento útil para el proceso de toma de decisiones, pues debido a la importancia que tiene la información en este último, su gestión es imprescindible.
- Un Proceso de Diagnóstico aporta beneficios como parte de un Proceso de Desarrollo de Software, su mayor utilidad consiste en el conocimiento pleno de la naturaleza de las organizaciones que serán automatizadas.
- Todo Sistema de Gestión de la Información es configurable y adaptable a las características y funcionalidades de la entidad que lo necesite o utilice, es por esto que el estudio de la factibilidad de estos sistemas en Cuba y el mundo, arrojó la decisión de desarrollar un Sistema de Gestión de la Información para satisfacer las necesidades del Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela.
- Las metodologías ágiles son muy eficaces en proyectos de corta duración y poco personal, pero su posibilidad de configurarse y adaptarse, sin perder el núcleo de su efecto, ha hecho que muchos equipos de desarrollo con características completamente diferente, las utilicen con resultados satisfactorios.
- El estudio de las diferentes herramientas, lenguajes y las diferentes terminologías necesarias para desarrollar un software, determinó el uso de las que más se acercaban a las necesidades del sistema en cuestión.
- Se realizó el diseño y la implementación de una herramienta que potencia la toma de decisiones en el Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela, cumpliendo con las regulaciones previstas por el cliente, incluyendo la funcionalidad de identificar

los procesos que se realizan en dichas entidades, con el objetivo de contribuir al proceso de desarrollo de software de la fase II del proyecto SAREN de la facultad 3.

- Se obtuvieron en la etapa de validación, resultados satisfactorios en cuanto al Sistema de gestión de la Información, para el Proceso de Diagnóstico a entidades legales de la República Bolivariana de Venezuela.

RECOMENDACIONES

Se recomienda basado en los resultados de la investigación:

- La implantación y uso del sistema para probar la eficiencia del mismo al potenciar la toma de decisiones en el proceso de diagnóstico a entidades legales de la República Bolivariana de Venezuela y su contribución en el proceso de desarrollo del software de la fase II del proyecto SAREN de la facultad 3.
- Migrar el sistema desarrollado hacia algún entorno de desarrollo integrado de código abierto o libre como el estudiado en el presente trabajo, Monodevelop, con aras de adentrar el software en la "Revolución del Software Libre" y garantizar la independencia tecnológica del mismo.
- Se recomienda la implementación de los requisitos opcionales del "Product Backlog", tomando como base la modelación del diseño de los mismos, que se ofrece en este trabajo.

REFERENCIAS BIBLIOGRÁFICAS

[1]. **Worley, Currey.** *Diagnóstico.* 2001.

[2]. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Person Education S.A, 2001.

[3]. **Sommerville, I.** *Ingeniería de Software.* s.l. : Pearson Educación, 2002.

[4]. **Pressman, R. S.** *Ingenería de Software. Un enfoque práctico.* 1999.

[5]. **Martínez, Alejandro y Martínez, Raúl.** *Guía a Rational Unified Process – Universidad de Castilla la Mancha.* 2001.

[6]. **Archer, T.** *C# a fondo.* 2001.

[7]. **Muñoz, A.** *Java del grano a su mesa.* 2000.

[8]. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Person Education S.A, 2000.

[9]. **Teknoda.** *¿Cómo conformar un entorno de programación java?* 2003.

[10]. **Javier García de Jalón, J. I.** *Aprenda Java como si estuviera en primero.* 2000.

[11]. **UCI.** *Ingeniería del software 1.Fase de Inicio. Flujo de trabajo de requerimientos.* 2006-2007.

[12]. **José H. Canós, P.** *Metodologías ágiles en el desarrollo de software.* 2003.

[13]. **Schwaber, y. o.** *Scrum.* 2003.

[14]. **Takeuchi.** *Scrum.* 1999.

[15]. **Tellez, Rolando.** *Explicando Scrum en 10 minutos.* 2007.

[16]. **Takeuchi.** *Scrum.* 2000.

[17]. **Newkirk.** *Extreme Programing.* 2007.

[18]. **Muñoz, A.** *Java: del Grano a su Mesa.* 2006.

[19]. **José F. Vélez Serrano, Á. A.** *Java.* 2000.

[20]. **Javier García de Jalón, J. I.** *Aprenda Java como si estuviera en primero.* 2001.

[21]. **Craig, Larman.** *UML y PATRONES. Introducción al análisis y diseño orientado a objetos.* 2004.

[22]. **Cooper, J.** *Introduction to Design Patterns in C#.* 2002.

[23]. **Buschmann, F. M.** *Pattern-Oriented Software Architecture: A System of Patterns.* 1996.

[24]. **Massachusetts Institute of Technology.** Massachusetts Institute of Technology. *Massachusetts Institute of Technology.* [En línea] 2003. [Citado el: 24 de Febrero de 2008.] http://mit.ocw.universia.net/6.170/6.170/f01/related-resources/java-qa.html.

[25]. **Bettye Rose Connell, M. J.** *Principios de Diseño.* 1997.

[26]. **Ignasi Fosch Alonso, J. A.** *Guía de Estilo de Codificación.* 2006.

[27]. **Vigo, Departamento de Informático de la Universidad e.** Sitio Web del Departamento de Informático de la Universidad e Vigo. *Sitio Web del Departamento de Informático de la Universidad e Vigo.* [En línea] 2003. [Citado el: 24 de Marzo de 2009.] http://www.lsi.uvigo.es/lsi/erosello/imo/articulos/rendimiento.pdf.

[28]. **Pressman, R. S.** *Ingenería de Software. Un enfoque práctico.* 2001.

[29]. **Mutis., A. E.** *Diseño e implementación de funcionalidades que se llevan a cabo en los registros mercantiles: solicitudes de expedientes, copias de documentos y sellado de libros.* La Habana : s.n., 2008.

BIBLIOGRAFÍA

1. **Alan.** Incubaweb. *Incubaweb.* [En línea] 14 de Junio de 2008. [Citado el: 12 de Marzo de 2009.] http://www.incubaweb.com/4987/herramientas/monodevelop-10-un-buen-editor-grafico-para-c-y-otros-lenguajes/.

2. **Alarcón, J. M.** IDG. *IDG.* [En línea] 1 de Diciembre de 2005. [Citado el: 23 de Febrero de 2009.] http://www.idg.es/pcworld/Nuevo--look--para-Visual-Studio-2005_Programacion-/art172522.htm.

3. **Ambler, S. W.** *Agile Modeling.* 2007.

4. **Arregui, J. J.** *Revisión Sistemática de Métricas de Diseño Orientado a Objetos.* 2005.

5. **Bulnes, A.** URIELLU .NET. *URIELLU .NET.* [En línea] 17 de Mayo de 2007. [Citado el: 14 de Marzo de 2009.] http://urriellu.net/es/articles-software/csharp-advantages.html.

6. **Corral, R.** *Por qué me gusta Scrum. Por qué me gusta Scrum.* 2006.

7. **Harrison, D. R.** *Medición en la Orientación Orientada a Objetos.* 2002.

8. **Joe Krebs, E. P.** *RUP en Diálogo con Scrum.* 2005.

9. **Kniberg, H.** *Scrum y XP desde las trincheras. Cómo hacemos Scrum.* 2007.

10. **Martínez, A.** Linux, Computación y más. *Linux, Computación y más.* [En línea] 20 de Marzo de 2008. [Citado el: 12 de Marzo de 2009.] http://www.antoniomtz.org/?q=comenzando-mono.

11. **Martínez, Alejandro y Martínez, Raúl.** *Guía a Rational Unified Process – Universidad de Castilla la Mancha.* 2001.

12. **Martínez, C. C.** *Consideraciones sobre la gestión de la información en función de la toma de decisiones en el sector cultural . Consideraciones sobre la gestión de la información en función de la toma de decisiones en el sector cultural.* 2006.

13. **Microsoft. (2009).** Centro de Información de Productos de Microsoft. *Centro de Información de Productos de Microsoft.* [En línea] 2009. http://www.microsoft.com/products/info/product.aspx?view=44&pcid=0a05620b-d256-487f-88d7-ceaa334cf95a&type=ovr.

14. **Palacio, J.** *Flexibilidad con Scrum.* 2008.

15. **Pressman, R. S.** *Ingenería de Software. Un enfoque práctico.* 2001.

16. **Rojas, D.** Icomparable. *Icomparable.* [En línea] 30 de Noviembre de 2008. [Citado el: 12 de Abril de 2009.] http://icomparable.blogspot.com/.

17. **Sommerville, I.** *Ingeniería de Software.* s.l. : Pearson Educación, 2002.

GLOSARIO DE TÉRMINOS

Bytecodes: Es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.

Capitalización Pascal: Estilo de codificación donde se capitaliza o se escribe en mayúscula, la primera letra de cada palabra.

Capitalización Camel: En este otro estilo se capitaliza la primera letra de cada palabra excepto la primera en palabras compuestas.

Capability Maturity Model Integration o Modelo de Capacidad y Madurez Integrado(CMMI): es un modelo para la mejora de procesos que proporciona a las organizaciones los elementos esenciales para obtener resultados efectivos y favorables.

Componente Diagnóstico: En este componente se recoge la información necesaria para diagnosticar cada entidad.

Componente Procesador: En este componente se procesa toda la información recogida a través de las planillas en formato XML que se generan en el primer componente y se visualizan los reportes necesarios para auxiliar la elaboración de los informes finales y potenciar la toma de decisiones.

Diagnóstico: Identificación de la naturaleza o esencia de una situación o problema y de la causa posible o probable del mismo o es el análisis de la naturaleza de algo.

European Computer Manufacturers Association (ECMA): Es una organización internacional basada en membrecías de estándares para la comunicación y la información.

Extensible Markup Language o Lenguaje de Marcas (**XML)**: Es un metalenguaje extensible de etiquetas, permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

Instituto de Ingenieros Eléctricos y Electrónicos (IEEE): Es una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación, ingenieros en informática e ingenieros en telecomunicación.

Interfaz de Programación de Aplicaciones (API): Es el conjunto de funciones o procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Licencia Pública General de GNU o GNU General Public License: Es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Licencia Distribución de Software Berkeley: Esta licencia se utiliza para identificar un sistema operativo derivado del sistema Unix nacido a partir de los aportes realizados a ese sistema por la Universidad de California en Berkeley.

Proceso de diagnóstico: Razonamiento dirigido en la determinación de la naturaleza, causas u origen de un fenómeno.

Protocolo de Capa de Conexión Segura o Secure Sockets Layer (SSL): Son protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet.

Servicio Autónomo de Registros y Notarías: Proyecto de la facultad 3 de la Universidad de las Ciencias Informáticas de Cuba, vinculado a Venezuela, con el objetivo de informatizar las entidades legales de este hermano país.

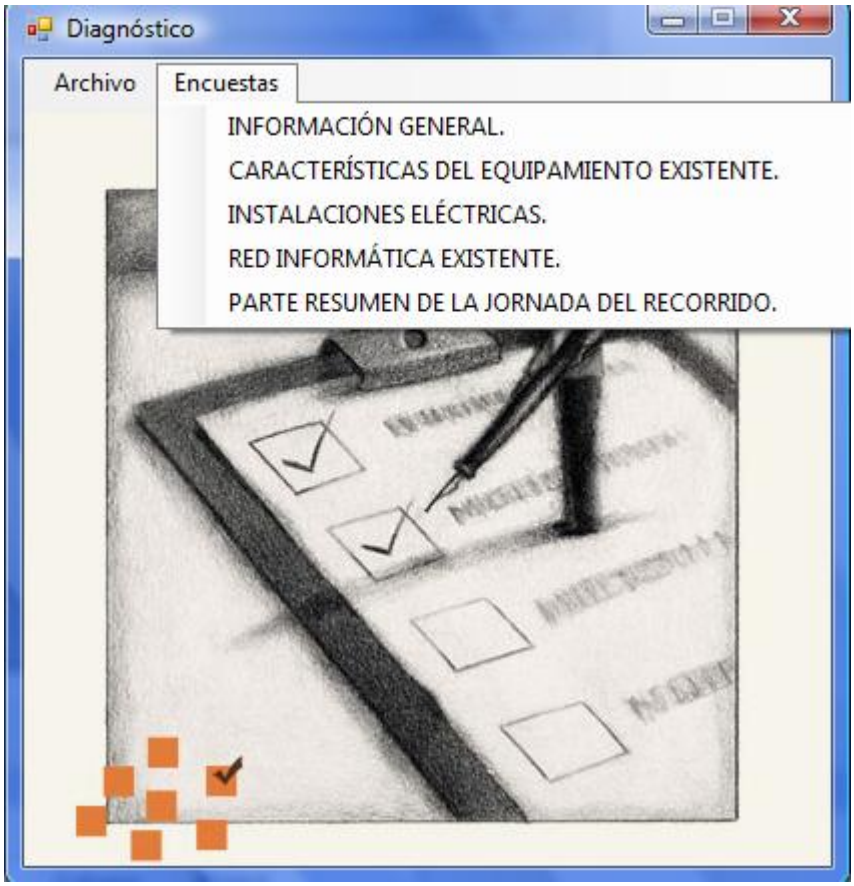
Sistema de Gestión de Información: Conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades que se realizan en una organización o para automatizar los procesos de trabajo que se efectúan dentro de la misma.

TableAdapter: Comunican la aplicación con una base de datos. Más específicamente, un TableAdapter se conecta con una base de datos, ejecuta consultas o procedimientos almacenados, y devuelve una nueva tabla de datos rellena con los datos devueltos o rellena una DataTable existente con los datos devueltos. Los TableAdapters también se utilizan para devolver los datos actualizados desde la aplicación a la base de datos.

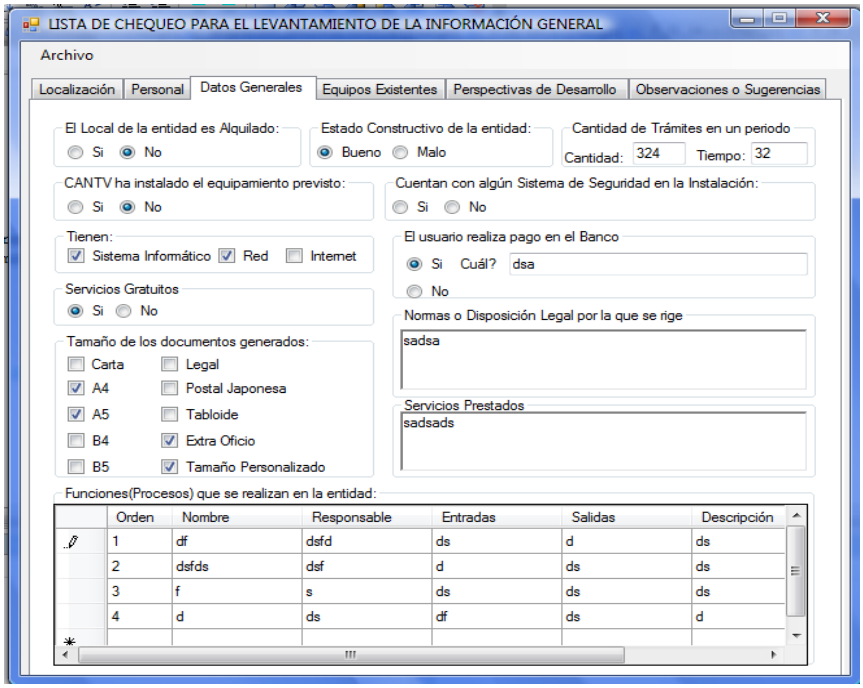
Toma de decisiones: Proceso de transformación de la información en acción.

XML Schema o Lenguaje de Esquema (XSD): es utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción.

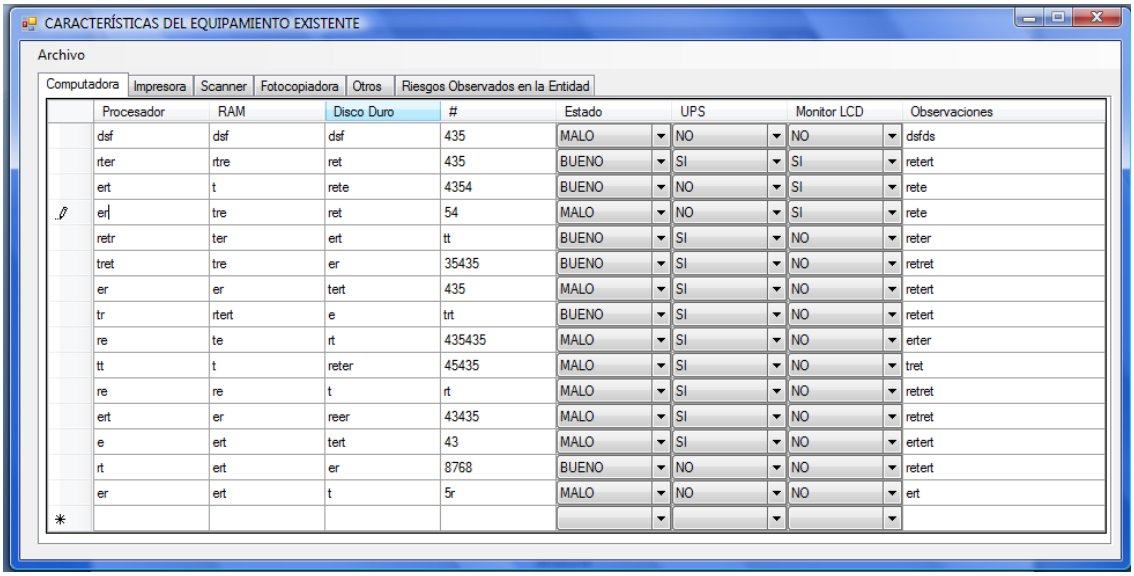
ANEXOS



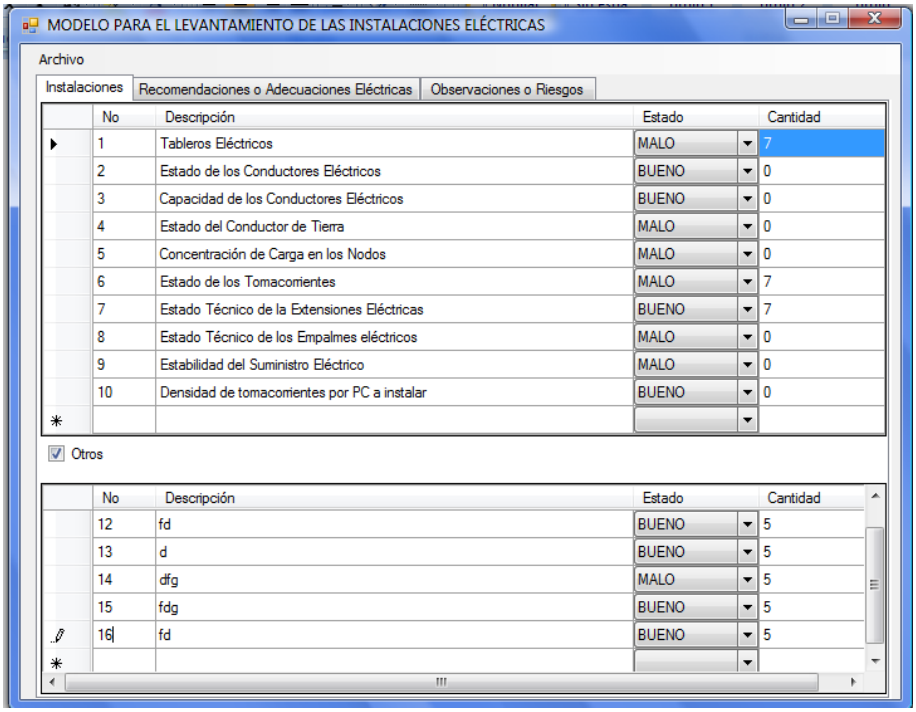
Anexo1. Sprint II



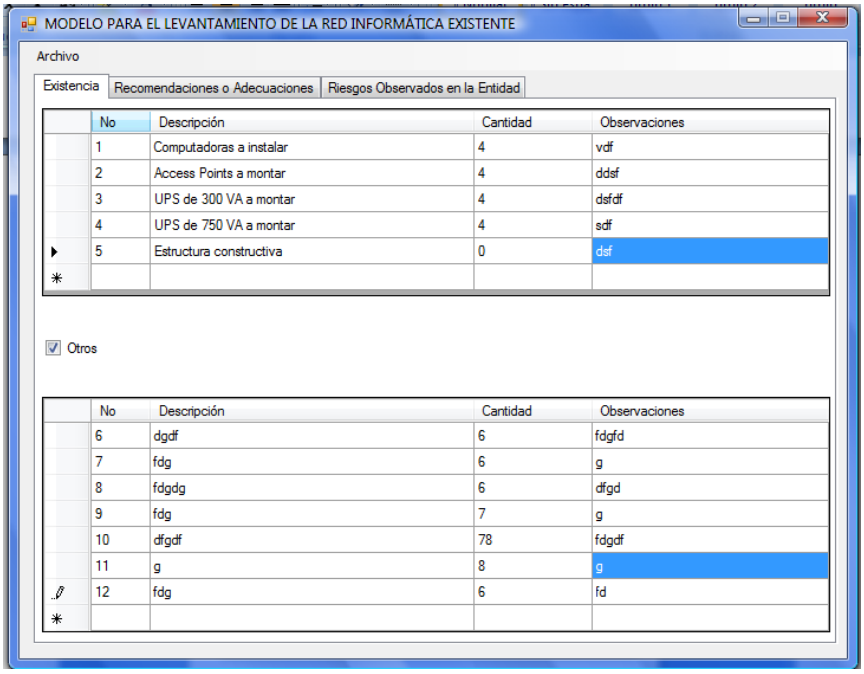
Anexo2. Sprint II



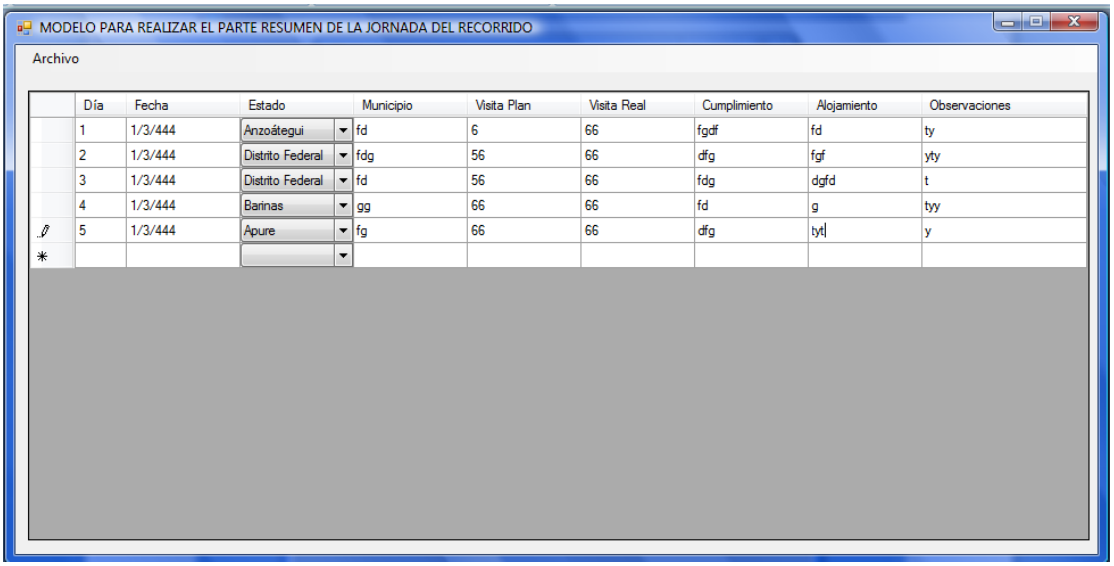
Anexo3. Sprint II



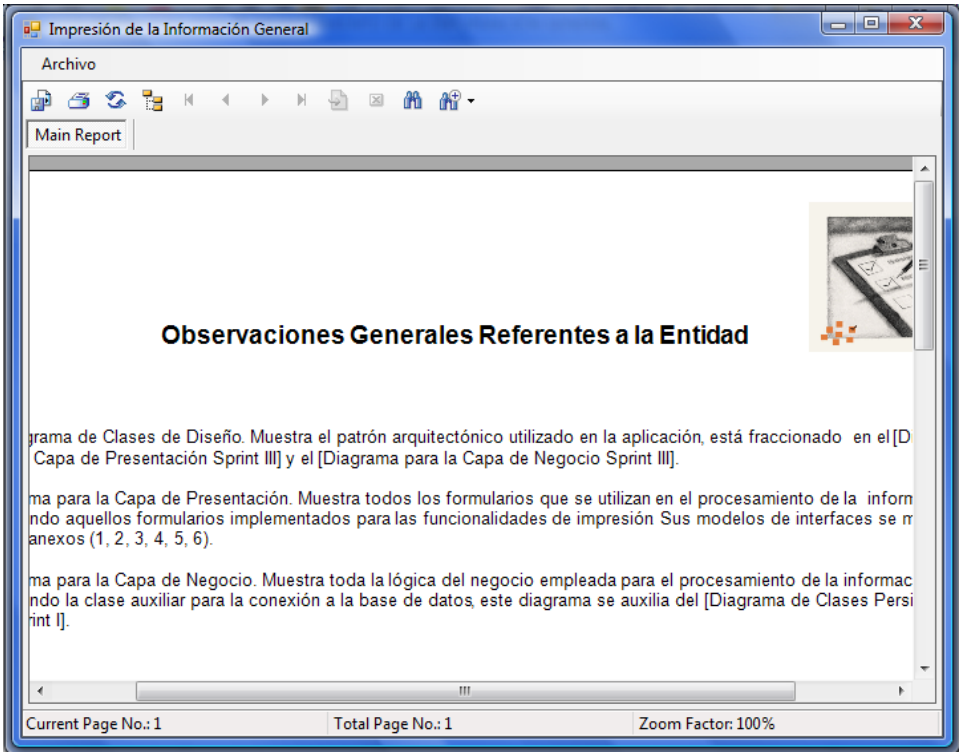
Anexo4. Sprint II



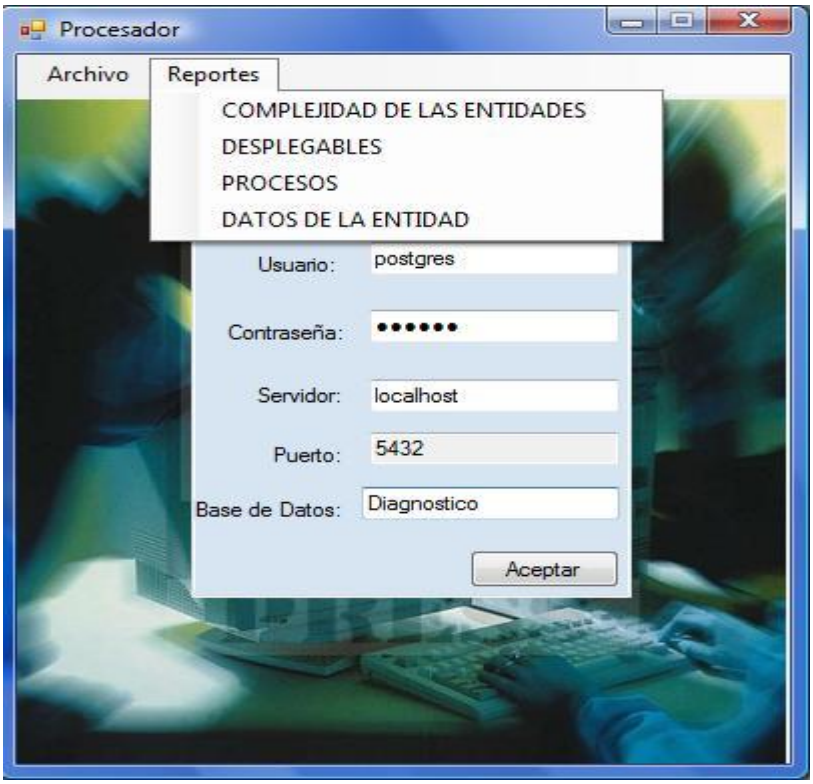
Anexo5. Sprint II



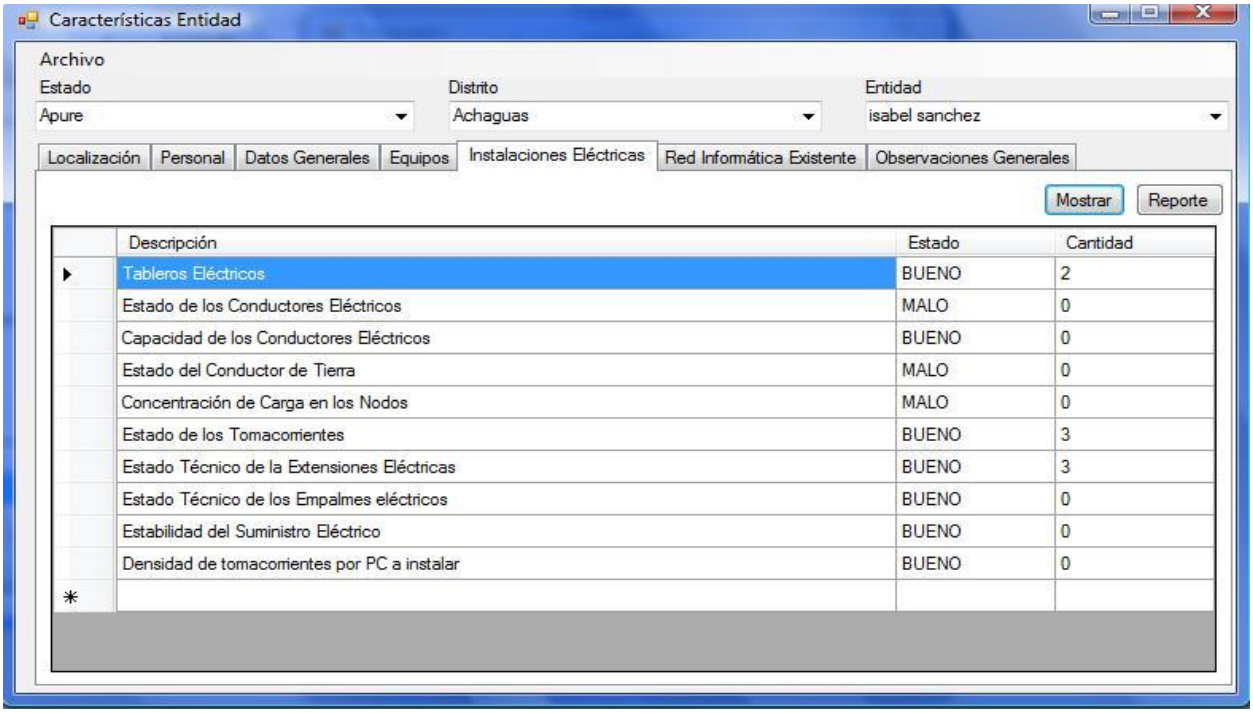
Anexo 6. Sprint II



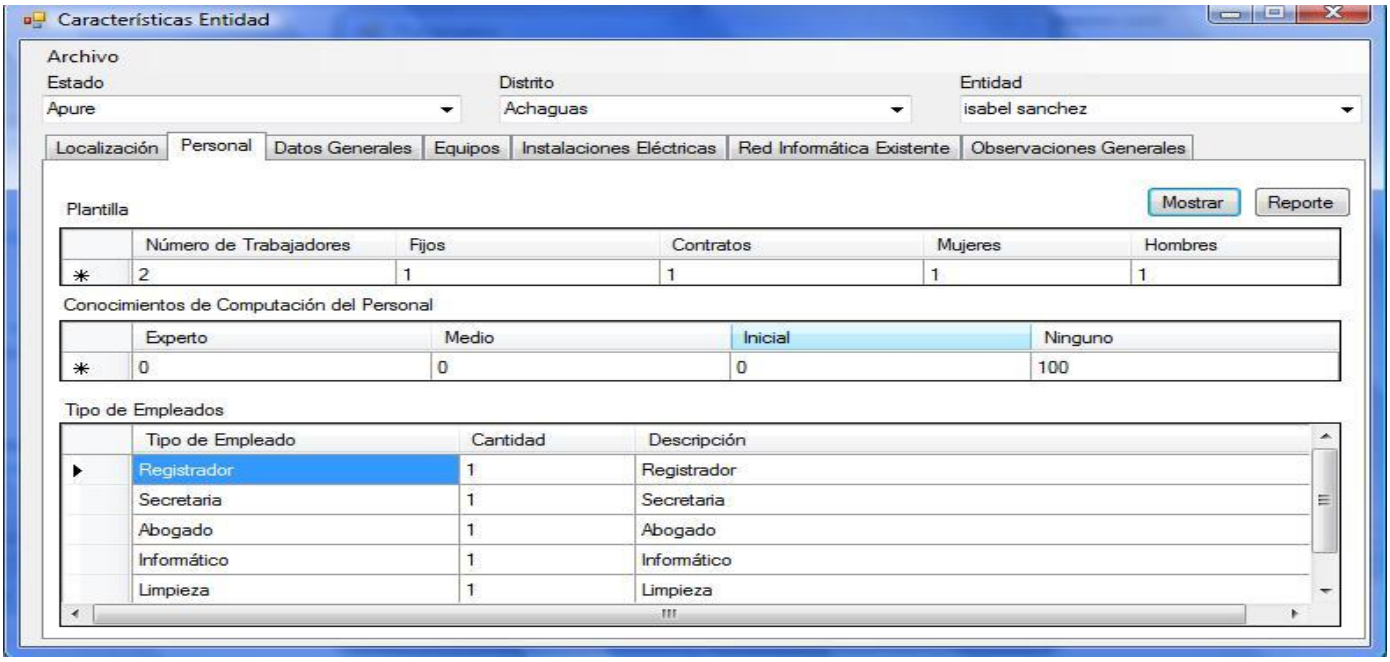
Anexo7. Sprint II



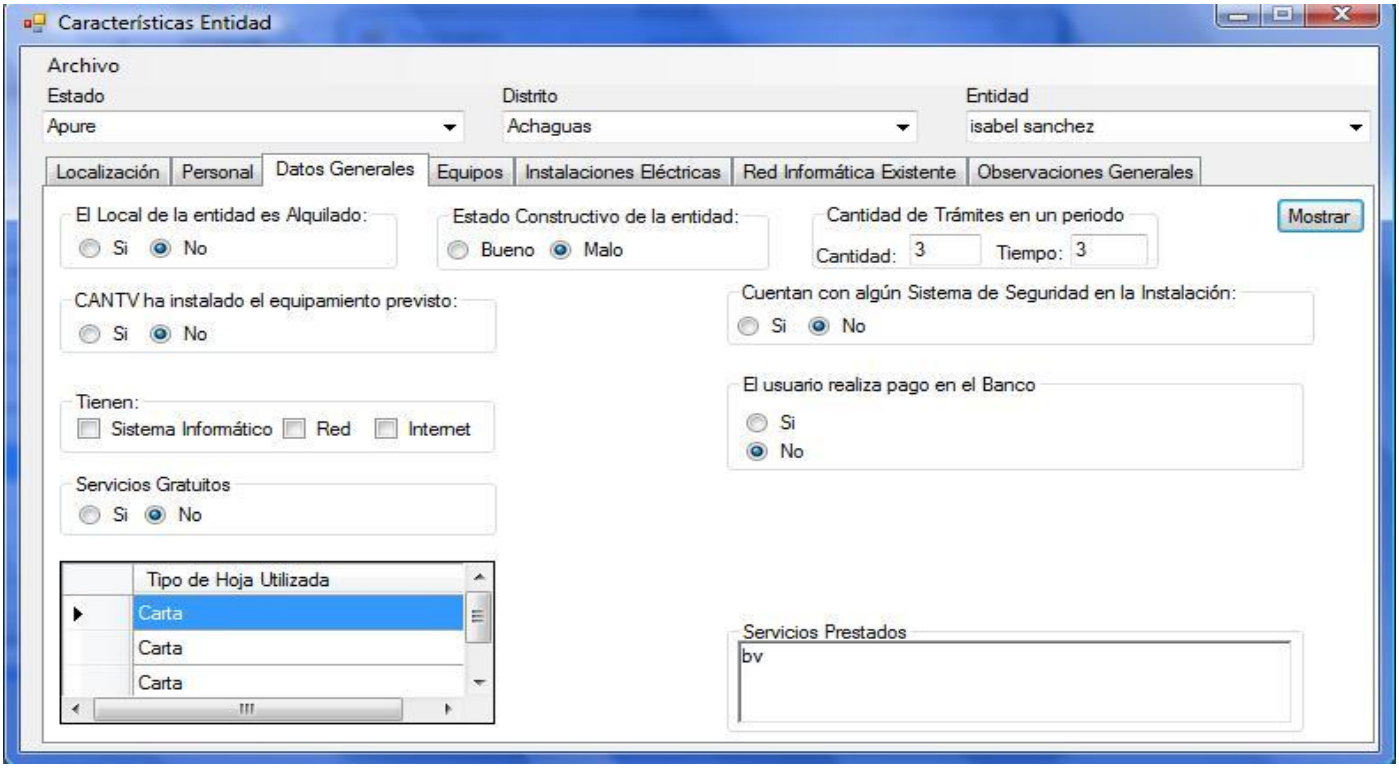
Anexo8. Sprint III



Anexo9. Sprint III



Anexo10. Sprint III



Anexo11. Sprint III

Algunas de las secciones del Casos de Prueba Complejidad del Componente Procesador.

Id del escenario	Escenario	*Clasificación*	*Reporte*	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	*Complejidad*	*V*	*V*	*El sistema verifica si los datos son válidos y guarda la información.*	*Satisfactoria*

Sección #2 Visualizar Pequeñas

Id del escenario	Escenario	Visualizar	Respuesta del Sistema	Resultado de la Prueba
EC 2	*Visualizar Pequeñas.*	*V*	*El sistema verifica si los datos son válidos y visualiza la información.*	*Satisfactoria*
		I	*El sistema muestra un mensaje de error informativo.*	*Satisfactoria*

Sección #2 Visualizar Medianas

Id del escenario	Escenario	Visualizar	Respuesta del Sistema	Resultado de la Prueba
EC 2	*Visualizar Medianas.*	*V*	*El sistema verifica si los datos son válidos y visualiza la información.*	*Satisfactoria*
		I	*El sistema muestra un mensaje de error informativo.*	*Satisfactoria*

Anexo12. Sprint III