UNIVERSIDAD DE COSTA RICA

Escuela de Ciencias de la Computación e Informática

Estándar de diseño

Elaborado por: Prof. Gabriela Salazar Bermúdez

II Semestre 2007

Índice

1. Alcance

Este documento describe un estándar para especificar el diseño de un proyecto o módulo de software. Especifica la información que debe contener el documento de diseño del software y recomienda una organización para éste. El *documento de diseño* es una representación del sistema que se utiliza para comunicar cualquier información del diseño del software.

Este estándar debe ser aplicado durante la etapa de diseño del software y está orientado hacia los desarrolladores de software.

2. Referencias

[1] ANSI/ISO/IEEE Std. 1016-1987, IEEE Recommended practice for software Design Descriptions.

[2] Pressman, Roger S. Ingeniería del Software, Un enfoque práctico. Editorial McGraw-Hill/Interamericana de España, S. A. 5ª edición. España, 2002

[3] Adaptable Process Model (APM), Pressman & Associates, info@rspa.com

[4] ANSI/ISO/IEEE Std. 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.

3. Definiciones

Las definiciones de términos no encontradas aquí pueden ser buscadas en [4].

Componente. Es una parte de un sistema que puede ser hardware o software y puede ser dividido en otros componentes. A menudo suelen utilizarse términos como: módulo, componente, unidad.

Entidad de diseño (ED). Es un elemento (componente) del diseño, estructural y funcionalmente distinto de los demás, y que es nombrado y referenciado. Por ejemplo, una consulta, una pantalla, una tabla de la base de datos.

Diseño de la arquitectura. Es el proceso de definir un conjunto de componentes de hardware y software y sus interfaces para establecer la estructura del desarrollo.

Descripción del diseño del software (DDS). Es un documento que se utiliza para comunicar información de diseño. Es un plano detallado que describe la arquitectura del software, sus componentes, interfaces y datos, y que es utilizado en la etapa de programación. El DDS muestra como el

software es estructurado para satisfacer los requerimientos funcionales. Cada requerimiento del software debe poder ser asociado a una o más entidades de diseño.

Prototipo. Versión preliminar de un software que sirve como modelo para las fases posteriores del desarrollo.

Requerimiento funcional. Requerimiento que especifica una función que el software o alguno de sus componentes debe ser capaz de realizar. Por ejemplo, el software debe permitir crear, modificar, y eliminar registros de una tabla de artículos.

Requerimiento de diseño. Requerimiento que especifica o restringe el diseño del software o de alguno de sus componentes.

IMEC (Insertar, Modificar, Excluir, Consultar). Administración de las entidades o tablas del sistema.

4. Consideraciones para producir un DDS

El diseño es el primer paso de la fase de desarrollo de cualquier producto de software. Puede definirse como "el proceso de aplicar distintas técnicas y principios con el propósito de definir un sistema de software con los suficientes detalles como para permitir su implementación".

Mediante el proceso diseño del software se traducen los requerimientos del software en una representación o modelo, que sirva como punto de partida para la etapa de programación.

5. Descripción del contenido del documento de diseño

Un documento de especificación de diseño debe incluir los puntos descritos en las Secciones 5.1 a la 5.11 y debe organizarse de acuerdo a la tabla de contenidos descrita en la sección 6 de este documento.

5.1. Portada

La portada debe seguir el formato descrito en la figura 1. Debe incluir al menos la siguiente información: nombre de la institución, el nombre del documento, el nombre de la aplicación, los autores del Plan, el número de versión y la fecha de aprobación.

Universidad de Costa Rica

Especificación del diseño

"Sistema Administración de Requerimientos"

Elaborado por: Juan Rojas Arias

Versión 1.0

Aprobado el 03 de marzo del 2005

Figura 1. Ejemplo de portada del documento "Especificación de Diseño".

5.2. Tabla de Contenido

Un documento de Descripción de Diseño debe estar organizado de acuerdo al formato que se presenta en la sección 6 de este documento.

5.3. Introducción

En esta sección se debe explicar el objetivo del documento de diseño, así como el contenido general del mismo y una breve reseña de cómo se usa. Además, se debe mencionar a manera de referencia, el método particular de diseño que se utilizará para realizar el diseño del software y cualquier herramienta, biblioteca de programas, notación, o sintaxis.

5.3.1 Alcance del Sistema

El alcance del sistema define los requerimientos del software sin entrar en detalles de implementación. Cada requerimiento debe presentarse con su correspondiente identificador numérico y su prioridad de implementación. En la tabla 1 se muestra un ejemplo de cómo se deben presentar.

Tabla 1. Ejemplo de requerimientos funcionales de acuerdo a su prioridad.

No.	Descripción : Mantenimiento Software	Prioridad
R1-1	Incluir nuevo software y nuevas versiones del mismo	Alta
R1-2	Modificar características de software ya incluido.	Alta
R1-3	Borrar software que ya no se utilice en la empresa	Alta
R1-4	Listar el software utilizado, con la posibilidad de filtrar información	Alta
R1-5	Enviar por correo lista de software a XXX	Baja

Además se deben incluir requerimientos que no se tomaron en cuenta durante la fase de análisis y explicar la razón por la cual no se incluyeron. Asimismo, deben de señalarse provisiones o planes futuros para nuevas versiones del software con respecto a estos requerimientos.

5.3.2 Definiciones y Términos.

Todos los términos específicos del software que se describen deben definirse, de tal manera que faciliten el entendimiento del documento de diseño.

5.3.3. Documentos de Referencia.

Debe contener todas las referencias bibliográficas y otras fuentes de información que se citan en el documento de diseño, incluyendo cualquier otro estándar y documento relacionado.

5.4. Integración con otros Sistemas.

Esta sección tiene como objetivo describir las relaciones de dependencia de datos del software con otros sistemas y las dependencias funcionales que existan entre el software y los otros sistemas y mostrarlas gráficamente. Solo si aplica se completa esta sección, si no aplica debe indicarse.

5.4.1 Dependencias Funcionales y de Datos.

Se describen las dependencias funcionales y de datos del software con *otros* sistemas. Una ***dependencia de datos*** existe cuando una de las funciones del software depende en alguna medida *de algún dato* proporcionado por otro sistema o módulo**.** Una ***dependencia funcional*** existe cuando una de las funciones del software depende en alguna medida *de alguna función* realizada por otro sistema o módulo.

Cada relación de dependencia o intercambio debe ser descrita brevemente en prosa, detallando el propósito y la naturaleza de los datos que se transfieren.

5.4.2 Diagrama de Integración.

Debe mostrar gráficamente las relaciones de intercambio de datos con otros sistemas.

5.5. Diseño de Datos.

El diseño de datos transforma los requerimientos establecidos durante la fase de análisis en las estructuras de datos que se necesitarán para implementar el software

5.5.1 Diagrama Detallado de la Base de Datos.

Se debe mostrar el Diagrama de Entidad Relación en donde se describen sus entidades. Cada entidad debe incluir sus atributos y las relaciones entre las entidades, incluyendo la cardinalidad.

5.5.2 Diagrama del Modelo Relacional

El objetivo de esta sección es brindar una descripción detallada de las tablas de la base de datos, suficientemente completa como para permitir su implementación en forma directa en algún lenguaje de definición de bases de datos, como por ejemplo SQL. Las tablas deben estar normalizadas.

5.5.3 Descripción de los Datos.

Para cada tabla que conforma la base de datos se deben especificar los siguientes aspectos:

5.5.3.1Nombre de la tabla. Cada tabla debe tener un nombre que la identifique unívocamente.

5.5.3.2 Propósito. Se debe justificar brevemente la razón de su existencia, especificando la manera como se satisfacen uno o más requerimientos.

5.5.3.3 ***Índices.*** Para cada tabla indicar los nombres de los campos que conforman la llave primaria y la(s) llave(s) foránea(s) indicando el nombre de la tabla a la que pertenecen las llaves foráneas.

5.5.3.4 Descripción de los atributos. Para cada atributo se deben describir brevemente los siguientes aspectos:

1) Identificación o nombre del atributo.
2) Tipo de dato (ejemplo: entero).
3) Longitud (ejemplo: 10 caracteres).
4) Restricciones (ejemplo: si acepta valores nulos o no).

A continuación se muestra un ejemplo:

Nombre de la tabla: Empleado

Propósito de la tabla: Satisfacer la necesidad de incluir, modificar, consultar y excluir los empleados de una organización.

Índices:

Llave primaria: Cédula

Llave foránea: NoDpto – **tabla a la que pertenece:** Departamento

Descripción de los atributos:

Identificación	Tipo de Dato	Longitud	Restricciones
Cédula	Hilera	15	No nulo
Nombre	Hilera	20	No nulo
Dirección	Hilera	50	-
NoDpto	Entero	4	No nulo

5.6. Diseño de la Arquitectura.

El objetivo de esta sección es brindar una visión general de la arquitectura del software. La arquitectura es como el mapa a seguir para desarrollar el software, El diseño de la arquitectura es el resultado de descomponer los requerimientos del software en componentes que puedan ser considerados, implementados, cambiados y probados con efectos mínimos sobre otros componentes. Ejemplos de componentes pueden ser: un subprograma, un módulo, un procedimiento, un proceso, un almacenamiento de datos, una interfaz.

5.6.1 Diagrama de la Arquitectura

Se debe proporcionar una descripción de cómo están organizados los diferentes componentes. Se debe presentar el diagrama de la arquitectura a utilizar y una justificación de la arquitectura seleccionada.

5.7. Diseño a Nivel de Subsistemas.

El objetivo es mostrar una descripción detallada de cada subsistema. Esto incluye la identificación del subsistema, las clases que los componen y el procesamiento y los datos de las clases, es decir los detalles necesarios que requieren los programadores antes de comenzar la codificación del software. Esta descripción también se puede utilizar en el plan de pruebas unitarias. En el anexo 1 de este documento se muestra un ejemplo de cómo elaborar esta sección.

Las secciones 5,7.1 a 5.7.3 se repiten por cada componente.

5.7.1 Identificación del Subsistema

Indicar el nombre del componente y presentar el diagrama de clases para el componente que se está describiendo mostrando los métodos de cada clase y si hay herencia la jerarquía de clases. No es necesario mostrar los atributos porque lo que se desea es destacar los métodos y el mostrar los atributos aumenta el tamaño del diagrama y dificulta el uso del documento.

5.7.2 Diseño de Mensajes

Describir las relaciones entre las clases del componente, indicando los métodos y atributos que posibilitan la comunicación. Se debe utilizar el diagrama de colaboración para mostrar los mensajes y hacer referencia a este archivo electrónico, el cual debe adjuntarse a este documento.

5.7.3 Diseño de Responsabilidades

Presentar una descripción detallada o el pseudocódigo del procedimiento de los métodos de cada clase.

5.8. Diseño de Mensajes entre Subsistemas.

5.8.1 Diagrama de Clases Global

Presentar el diagrama de clases global (que incluye todos los subsistemas) mostrando las relaciones entre ellos. En la figura 2 se muestra un diagrama en donde cada paquete represente un subsistema y a través de flechas discontinuas se muestra la relación entre los subsistemas.

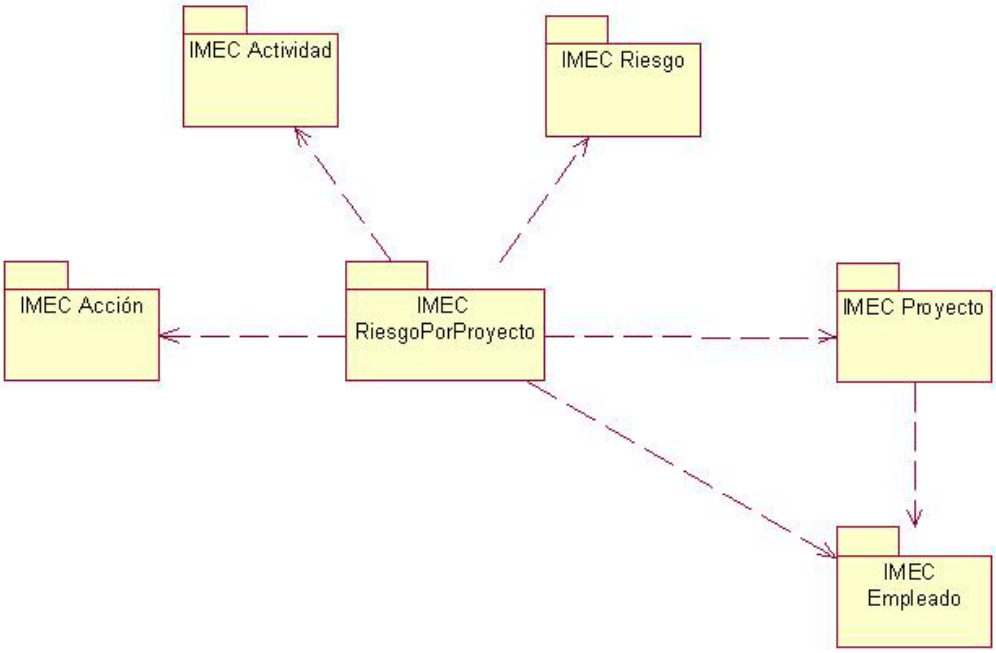


Figura 2. Ejemplo del flujo de navegación entre pantallas.

5.8.2 Descripción de los Mensajes entre los Subsistemas

Describir las relaciones de asociación entre los subsistemas, indicando los métodos que posibilitan la comunicación. En el anexo 1 de este documento se

muestra un ejemplo de cómo describir los mensajes entre los subsistemas.

5.9. Diseño de la Interfaz del Usuario.

El objetivo es dar una breve reseña, a manera de referencia, del prototipo final que se implementó. El prototipo final debe incluir los siguientes aspectos: *Pantallas*, *Filtros*, *Reportes*, *Navegación*, *Procesos* e *Interacción con el usuario*. El prototipo no implementa aspectos tales como: manipulación de la base de datos, validaciones de campos e implementación de los procesos y reportes. En el anexo 1 de este documento se muestra un ejemplo de cómo realizar esta sección.

5.9.1 Diagrama de Navegación de las Pantallas

Se presenta un diagrama que muestre la navegación de las pantallas del sistema.

5.9.2 Descripción de las Pantallas

Se identifica y presenta cada pantalla y se describen todos los objetos y acciones incluidas.

5.10 Aspectos de Pruebas

En esta sección se presenta la estrategia de pruebas y la especificación de casos de pruebas preliminares.

5.10.1 Clases de Pruebas

Se especifican las estrategias de prueba que se realizarán.

- La ***prueba de unidad*** se centra en cada módulo individual del software, tal como está implementado en código fuente.
- La ***prueba de integración*** donde el foco es el diseño y la construcción de la arquitectura del software.
- La ***prueba de validación*** se validan los requisitos establecidos (funcionales, de comportamiento y de rendimiento), comparándolos con lo que se ha construido.
- La ***prueba del sistema*** verifica que cada elemento encaja con la funcionalidad y el rendimiento del sistema total.

Además se debe indicar el tipo de prueba (caja blanca, caja negra) que se aplicará en cada estrategia.

5.10.2 Identificación de Componentes Críticos

Se identifican aquellos componentes críticos a los que hay que ponerles particular atención durante las pruebas.

5.10.3 Casos de Pruebas

Se deben definir los casos de prueba que se utilizarán durante la fase de pruebas del sistema. Se recomienda utilizar la siguiente plantilla.

1. **Identificación y nombre del Caso de Prueba:** Especificar la identificación y el nombre del caso de prueba. Estos deben coincidir con los del requerimiento que se está probando.

2. **Caso de uso asociado:** Indicar los nombres de los casos de uso asociados con el requerimiento que se está probando.

3. **Propósito:** Especificar el propósito de la prueba.

4. **Descripción General:** Se debe describir brevemente el requerimiento a probar.

5. **Precondición:** Se especifican los eventos que sucedan antes de que se despliegue la interfaz del requerimiento a probar.

6. **Post-condición:** Se especifica cualquier evento que suceda después de ejecutado el requerimiento.

7. **Entrada:** Especificar cualquier flujo de información o de control que se requiera para ejecutar el requerimiento.

8. **Flujo único de eventos:**

#	Acciones de algún actor	#	Acciones del sistema	#	Punto de verificación

#: Corresponde al número con que se identifica el evento.

Acciones de algún actor: Se debe indicar la acción de un actor al interactuar con el sistema.

Acciones del sistema: Corresponde a la respuesta del sistema ante la acción del actor.

Punto de verificación: A manera de pregunta se verifica si la acción esperada del sistema se dio exitosamente.

9. **Salida:** Se indican los flujos de información producto de la ejecución del requerimiento.

10. **Reporte:** Especificación de los resultados de la prueba.

5.11 Cambios a Sistemas Existentes.

Deben describirse todos los cambios que el diseño del software genere en otros sistemas y módulos (si aplica). Para cada cambio que se genere deben especificarse:

1. Descripción del cambio (ejemplo: modificar la tabla de clientes)
2. La razón o justificación por la que el cambio es necesario (ejemplo: permitir llevar un control cruzado entre clientes).
3. Si es necesario realizar el cambio en el otro sistema para el software que se está diseñando.

5.12 Anexos

Los anexos deben incluir cualquier otra información adicional que pueda ayudar a especificar mejor el documento de diseño.

6. Formato del Documento de Especificación de Diseño

1. Portada
2. Tabla de Contenido
3. Introducción
    - 3.1 Alcance del Sistema.
    - 3.2 Definiciones y Términos.
    - 3.3 Documentos de Referencia.
4. Integración con otros Sistemas.
    - 4.1 Dependencias funcionales y de datos.
    - 4.2 Diagrama de integración.
5. Diseño de Datos.
    - 5.1 Diagrama Detallado de la Base de Datos.
    - 5.2 Diagrama del Modelo Relacional.
    - 5.3 Descripción de los datos.
6. Diseño de la arquitectura.
    - 6.1 Diagrama de la arquitectura.
7. Diseño a nivel de subsistemas.
    - 7.1 Identificación del subsistema.
    - 7.2 Diseño de los mensajes.
    - 7.3 Diseño de responsabilidades.
8. Diseño de mensajes entre subsistemas.
    - 8.1 Diagrama de clases global.
    - 8.2 Descripción de los mensajes entre los subsistemas.
9. Diseño de la interfaz del usuario.
    - 9.1 Diagrama de navegación entre pantallas.
    - 9.2 Descripción de las pantallas.
10. Aspectos de pruebas.
    - 10.1 Clases de pruebas a realizar.

Anexo 1

Caso de ejemplo: Especificación del diseño para el sistema SAWMI

Hoja de portada

Universidad De Costa Rica

Especificación Diseño

Sistema de Contabilidad y Facturación SAWMI

Elaborador por:

Versión 1.0

Aprobado: 17 de octubre de 2006.

Tabla de contenido

1. Introducción

En el presente documento de especificación de diseño, se describirán de manera detallada todos los elementos que rodean la implementación del Sistema de Facturación para la empresa SAWMI.

Se presentarán inicialmente todos los requisitos que se deben cumplir en el sistema y se abarcarán las cuatro áreas correspondientes al diseño de la siguiente manera:

1. **Datos**: Se describe de manera detallada las clases del sistema y las correspondientes estructuras de datos utilizadas. Se presenta el esquema entidad relación. Además la estructura de las tablas (detalle para cada tabla de su llave primaria, foránea y sus restricciones) con sus correspondientes atributos (se detallan nombres, tipos y longitud de cada uno).
2. **Arquitectura**: Se describe la arquitectura que se utilizará para desarrollar la aplicación y se muestra el esquema de arquitectura correspondiente.
3. **Interfaz:** Se presentan las interfaces por cada módulo y la navegabilidad entre pantallas de la aplicación. Además se describen los componentes de cada pantalla.
4. **Componentes: S**e detalla la relación existente entre los principales elementos estructurales del sistema, la composición y la funcionalidad de cada componente.

1.1 Alcance del proyecto

El alcance determinado para el proyecto está dado por los requerimientos establecidos para el sistema SAWMI los cuales se presentan a continuación:

Lista de Requerimientos Funcionales de acuerdo a su prioridad

No.	Descripción: Administrar Factura	Prioridad
R1.1	Incluir una nueva Factura con el detalle de los productos comprados	Alta
R1.2	Anular una factura que ya existe	Alta
R1.3	Mostrar los datos de una determinada factura existente	Alta

No.	Descripción: Administrar Proforma	Prioridad
R2.1	Incluir una nueva Proforma con su datos correspondientes	Alta
R2.2	Modificar los datos de una Proforma por solicitud del cliente	Alta
R2.3	Borrar una Proforma existente	Alta
R2.4	Listar todos los productos y datos de una Proforma ya existente	Alta

No.	Descripción: Administrar Pedido Proveedor	Prioridad
R3.1	Incluir un nuevo Pedido a un Proveedor	Alta
R3.2	Modificar los datos de un Pedido antes de realizarlo	Alta
R3.3	Borrar un Pedido existente	Alta
R3.4	Listar los productos y datos de un determinado Pedido	Alta
R3.5	Realizar la recepción de un Pedido	Alta

No.	Descripción: Administrar Miembro (Cliente y Empleado)	Prioridad
R4.1	Incluir un nuevo Miembro con sus respectivos datos	Alta
R4.2	Modificar los datos correspondientes a un Miembro	Alta
R4.3	Borrar un Miembro existente	Alta
R4.4	Mostrar los datos de un Miembro	Alta

No.	Descripción: Administrar Proveedor	Prioridad
R5.1	Incluir un nuevo Proveedor con sus respectivos datos	Alta
R5.2	Modificar los datos de un Proveedor ya existente	Alta
R5.3	Borrar un Proveedor existente	Alta
R5.4	Mostrar todos los datos correspondientes a un Proveedor	Alta

No.	Descripción: Administrar Usuario	Prioridad
R6.1	Incluir un nuevo Usuario	Alta
R6.2	Modificar los datos de un determinado Usuario	Alta
R6.3	Borrar un Usuario Existente	Alta
R6.4	Consultar los datos de un Usuario	Alta
R6.5	Validar el Ingreso al Sistema de un Usuario	Alta

No.	Descripción: Administrar Producto	Prioridad
R7.1	Insertar un nuevo Producto	Alta
R7.2	Modificar los datos de un Producto existente	Alta
R7.3	Borrar un Producto existente	Alta
R7.4	Mostrar los datos de un Producto Existente	Alta

No.	Descripción: Administrar Consulta Complejas	Prioridad
R8.1	Mostrar todas las Facturas realizadas en un determinado rango de tiempo.	Alta
R8.2	Mostrar todo el historial de facturas correspondiente a un determinado cliente.	Alta
R8.3	Mostrar todos los productos con su correspondiente proveedor comprados por un Cliente específico.	Alta
R8.4	Mostrar todos los productos correspondientes a un Proveedor	Alta
R8.5	Mostrar todos los pedidos realizados por un determinado empleado con su correspondiente detalle, en un rango de fechas.	Alta
R8.6	Mostrar el historial de los pedidos realizados a todos los proveedores y el correspondiente promedio de tiempo de entrega de cada proveedor	Alta
R8.7	Mostrar todas las facturas en las cuales se incluya un determinado producto(el producto esta asociado a un único proveedor), en un rango de fechas	Alta
R8.8	Mostrar todas las ventas realizadas por un determinado empleado con su correspondiente detalle, en un rango de fechas.	Alta

Planes futuros para nuevas versiones de software: el sistema será diseñado para que en caso que sea necesario agregar un nuevo requerimiento o módulo este se agregue de manera independiente a todos los demás, por tanto el agregar una nueva funcionalidad al sistema no será ningún problema ya que no afectará las funciones existentes.

1.2 Definiciones y Términos

***Software**:* Programa o programas con una funcionalidad determinada, compuestos de un diseño, arquitectura, código fuente producto de su implementación, ejecutables generados y su correspondiente documentación.

***Componente**:* es una parte del software la cual presenta: funcionalidad, modularidad, independencia, encapsulamiento de una determinada función y contiene un conjuntos de interfaces para la comunicación con otros módulos.

Entidad de Diseño(ED): elemento estructural del diseño. Utilizada para mencionar aspectos como una tabla en la base de datos.

***Diseño de la arquitectura**:* Es el proceso de definición de la estructura del sistema para el desarrollo y su posterior mantenimiento. En él se definen los componentes y la relación entre estos. Además evalúa patrones de diseño y estilos arquitectónicos.

Descripción del diseño del software(DDS): Es un documento que se utiliza para comunicar información de diseño. Es un plano detallado que describe la arquitectura del software, sus componentes, interfaces y datos, y que es utilizado en la etapa de programación.

Prototipo: es un modelo preliminar del software que da una visión del sistema final al usuario. Es útil para la captura de requerimientos y para las fases posteriores de implementación.

Requerimiento Funcional: es la especificación de una función que el software debe ser capaz de realizar a través de un componente o conjunto de componentes

***IMEC** (Insertar, Modificar, Eliminar, Consultar):* Administración de las entidades o tablas del sistema.

1.3 Documentos de Referencia

[1] Pressman Roger. “Ingeniería del Software: Un enfoque práctico”. Mc Graw Hill/Interamericana de España, S.A. Sexta Edición, España 2005.

[2] Salazar Gabriela. “Estándar de Diseño”, Versión 1, 2007.

[3] ANS/ISO/IEEE std 1016-1987, “IEEE Recommended practice for software Design Descriptions”.

[4] Adaptable Process Model (APM), Pressman & Associates, info@rspa.com

[5] ANSI/ISO/IEEE Std. 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.

2. Integración con otros Sistemas

Esta sección no aplica al Sistema SAWMI porque el sistema es independiente y acorto plazo no se planea integrar con ningún otro.

3. Diseño de Datos

Durante esta fase se tomará el trabajo realizado en el análisis para establecer las clases y las estructuras de datos necesarias para la implementación del sistema. Seguidamente se presenta el análisis de los datos.

3.1 Diagrama Detallado de la Base de Datos

A continuación se muestra la descripción del diagrama Entidad Relación para el Sistema SAWMI con sus entidades, atributos, relaciones entre las entidades y la cardinalidad.

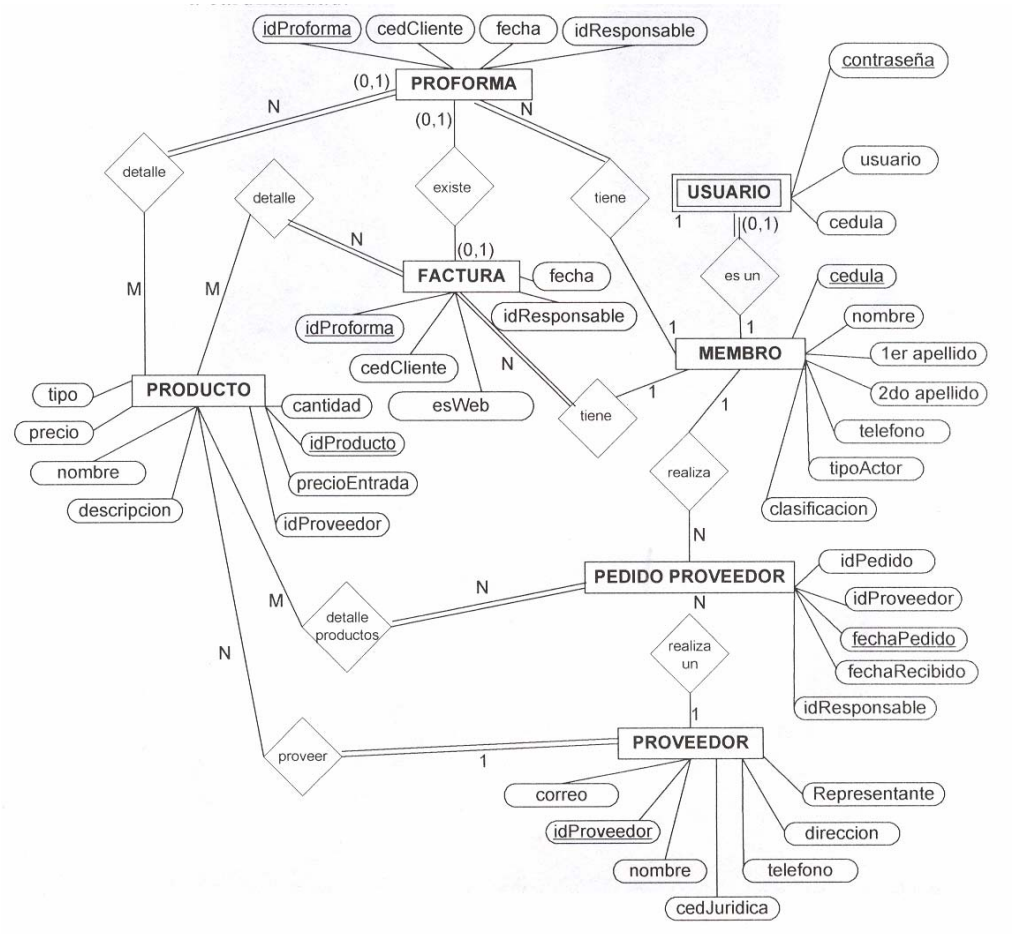
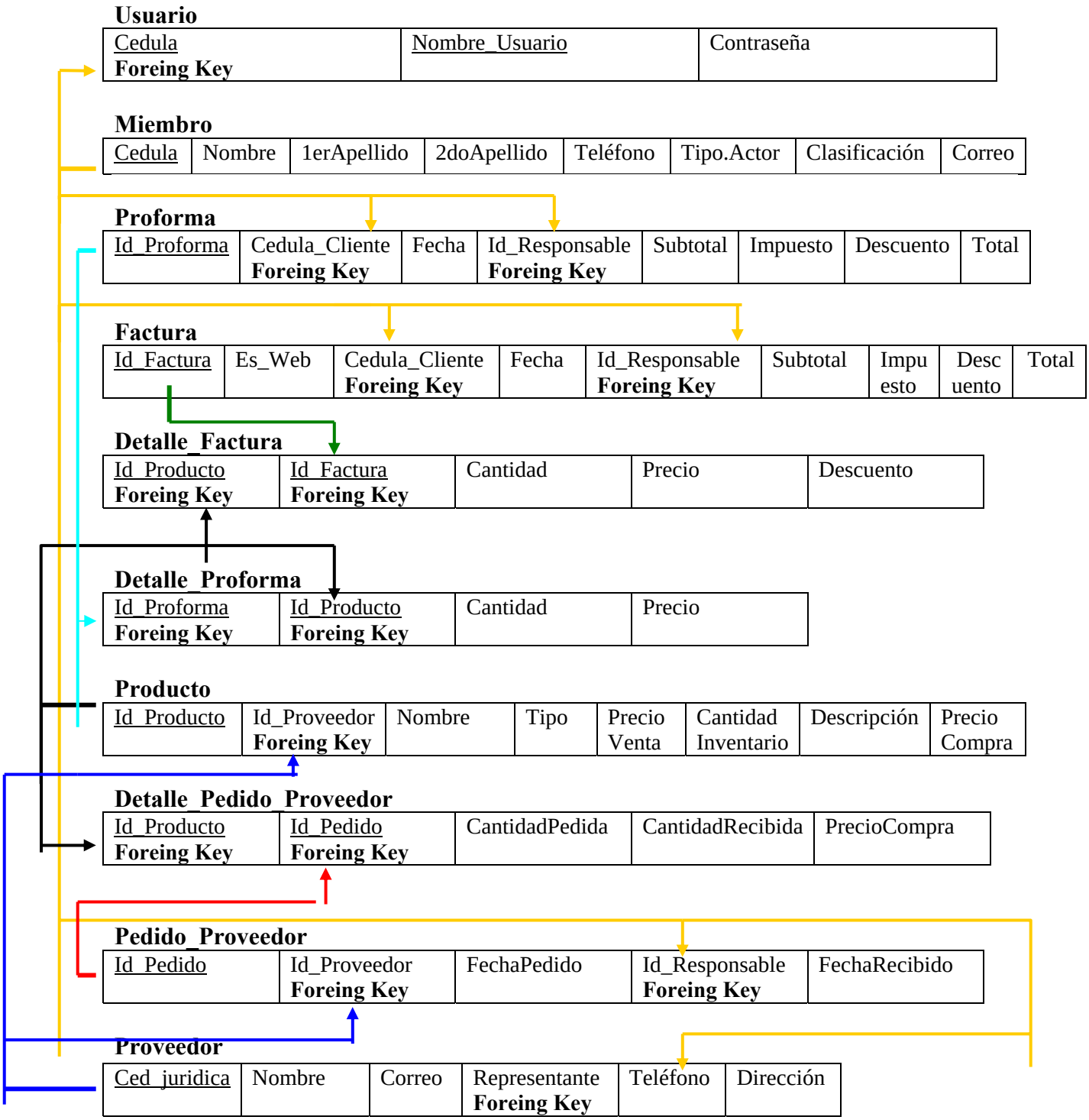


Figura 1. Representación del diagrama Entidad-Relación de todo el sistema.

3.2 Diagrama del Modelo Relacional

En esta sección se presenta el modelo relacional, con todas las tablas de la base de datos, con todos sus detalles (atributos, llaves primarias y foráneas) y la relación presente entre las tablas (presentada en el diagrama entidad-relación figura #1), pero ahora mostrando las tablas normalizadas.



3.3 Descripción de los Datos

Para ejemplificar esta sección a continuación se presenta la descripción de algunas de las tablas que se diseñaron para el Sistema SAWMI.

- **Nombre de la Tabla: Producto**

Propósito de la tabla: Administrar (insertar, modificar, eliminar, consultar) los datos de los productos que la empresa ofrece para la venta. Se requiere para controlar el inventario que se tiene, para saber cuando se debe enviar a comprar un producto y cuando el producto es traído por el proveedor se aumenta la cantidad en el Inventario, esto se hace para cada producto.
Índices:

Llave primaria: Id_Producto
Llave foránea: Id_Proveedor → Tabla a la que pertenece: Proveedor

Descripción de los atributos

Identificación	Tipo de Dato	Longitud	Restricciones
Id_Producto	Varchar	50	No nulo
Id_Proveedor	varchar	20	No nulo
Nombre	varchar	30	No nulo
Tipo	Int	2	No nulo
Precio_Venta	float	8	No nulo
Cantidad_Inventario	Int	5	No nulo
Descripción	varchar	100	-
Precio_Compra	float	8	No nulo

- **Nombre de la Tabla: Proveedor**

Propósito de la tabla: Administrar (insertar, modificar, eliminar, consultar) los datos de los proveedores que ofrecen productos a la empresa. Necesario para saber cuales productos ofrece cada proveedor (cada producto lo ofrece un único proveedor).
Índices:

Llave primaria: Ced_Juridica
Llave foránea: Representante→ Tabla a la que pertenece: Miembro

Descripción de los atributos

Identificación	Tipo de Dato	Longitud	Restricciones
Ced_Juridica	Varchar	20	No nulo
Nombre	varchar	25	No nulo
Correo	varchar	30	No nulo
Representante	Varchar	15	No nulo
Teléfono	Varchar	15	-
Dirección	Varchar	100	-

- **Nombre de la Tabla: Proforma**

Propósito de la tabla: Almacenamiento y administración (insertar, modificar, eliminar y consultar) de los datos de las preformas de los clientes. Se requiere además para poder generar una factura con una cotización antes hecha y estos datos se encuentran en la proforma.
Índices:

Llave primaria: Id_Proforma
Llave foránea: Cedula_Cliente → Tabla a la que pertenece: Miembro
Id_Responsable → Tabla a la que pertenece: Miembro

Descripción de los atributos

Identificación	Tipo de Dato	Longitud	Restricciones
Id_Proforma	serial		No nulo
Cedula_Cliente	Int	10	No nulo
Fecha	DataTime		No nulo
Id_Responsable	Int	10	No nulo

- **Nombre de la Tabla: Detalle Proforma**

Propósito de la tabla: manejo del detalle de todos los productos incluidos en una Proforma. Incluye la cantidad cotizada para cada producto, además el precio en el que se cotizó el producto (pues los precios pueden aumentar y se desea manejar el precio con el cual se cotizó el producto), además necesario para el manejo del historial de los detalles de las preformas.

Índices:

Llave primaria: Id_Producto Id_Proforma
Llave foránea: Id_Proforma → Tabla a la que pertenece: Proforma
Id_Producto → Tabla a la que pertenece: Producto

Descripción de los atributos

Identificación	Tipo de Dato	Longitud	Restricciones
Id_Producto	varchar	50	No nulo
Id_Proforma	Serial		No nulo
Cantidad	Int	5	No nulo
Precio	float	8	No nulo
Descuento	float	2	-

- **Nombre de la Tabla: Factura**

Propósito de la tabla: Almacenamiento y administración (insertar, anular y consultar) de los datos de las facturas de los clientes. Es necesario para manejar un historial de las ventas y registrar todas las ganancias de la empresa. Se requiere además controlar las ventas de los productos en general, llevar un control de las ventas y de los movimientos de la clientela dependiendo de la época.

Índices:

Llave primaria: Id_Factura
Llave foránea: Cedula_Cliente → Tabla a la que pertenece: Miembro
Id_Responsable → Tabla a la que pertenece: Miembro

Descripción de los atributos

Identificación	Tipo de Dato	Longitud	Restricciones
Id_Factura	serial		No nulo
Cedula_Cliente	Int	10	No nulo
Fecha	DataTime		No nulo
Id_Responsable	Int	10	No nulo

- **Nombre de la Tabla: Detalle Factura**

Propósito de la tabla: manejo del detalle de todos los productos incluidos en la factura. Incluye la cantidad comprada por cada producto, además el precio en el que se vende el producto (pues los precios pueden aumentar y se desea manejar el precio con el cual se vendió el producto), además necesario para el manejo del descuento que se le aplica a cada producto vendido.

Índices:

Llave primaria: Id_Producto Id_Factura

Llave foránea: Id_Factura → Tabla a la que pertenece: Factura

Id_Producto → Tabla a la que pertenece: Producto

Descripción de los atributos

Identificación	**Tipo de Dato**	**Longitud**	**Restricciones**
Id_Producto	Varchar	50	No nulo
Id_Factura	Int	8	No nulo
Cantidad	Int	3	No nulo
Precio	Float	8	No nulo
Descuento	Flota	2	-

4. Diseño de la Arquitectura

La arquitectura que se va a aplicar es la estratificada por capas. Este modelo arquitectónico consiste en crear diferentes capas, donde cada una de las capa realiza operaciones que progresivamente se aproximan más al cuadro de instrucciones de la máquina. En la capa externa los componentes sirven a las operaciones de interfaz de usuario, en la capa interna los componentes realizan operaciones de interfaz de usuario y en las capas intermedias proporcionan servicios de utilidad y funciones del software de aplicaciones. La organización jerárquica se realiza de tal manera que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de los servicios de la capa inmediatamente inferior.

4.1 Diagrama de la Arquitectura

La arquitectura se presente en la siguiente figura:

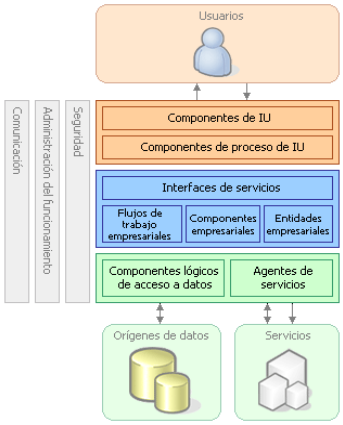


Figura 2. Esquema de la arquitectura por capas. Fuente ayuda de Microsoft co.

La descripción de las partes de la arquitectura base es la siguiente:

1 **Capa Interfaz:** Es la capa que se comunica con el usuario y está compuesta por:

Componentes de IU: actúan directamente con el usuario.

Componentes de Proceso de IU: son abstractos y son los intermediarios entre la capa de interfaz y la capa de negocios. Permiten la validación y flujo entre ventanas. Un beneficio muy importante es el que se presente menos código en las ventanas.

2 **Capa de Negocios:** es la encargada de manipular los servicios y está compuesta por:

Interfaces de Servicio: permite la comunicación con los servicios

Flujos de Trabajo Empresariales: encargada de manejar el flujo de las actividades (en muchas ocasiones lo único que se hace es el llamado a los componentes Empresariales) Por lo general están compuestos de componentes de negocio que realizan cada una de las actividades de proceso.

Componentes Empresariales o de Negocio: Implementan la lógica funcional de una aplicación. Actúan sobre una entidad de negocio, proveyendo todas las operaciones necesarias para manipular dicha actividad. Validan las reglas de negocio que aplican sobre las entidades de negocio.

Entidades Empresariales o de Negocio: Representan las estructuras de información que fluyen en un flujo de trabajo y que son manipuladas por los componentes de negocio.

3 **Capa de Acceso de Datos**

Componentes lógicos de acceso a datos: Proveen la lógica necesaria para realizar operaciones de consulta y acceso a la base de datos requerida por los componentes y las entidades de negocio.

Agentes de Servicios: Encargados de realizar toda la gestión y comunicación con los agentes externos. Son agentes que permiten la búsqueda de servicios en tiempo de ejecución (agentes inteligentes)

4 **Capa de Datos:** Permite el almacenamiento persiste de los datos (base o almacén de datos).

Las razones por las que se eligió la arquitectura N-capas para este sistema son las siguientes:

1. Permite la creación de componentes con una alta cohesión (encapsulan una función específica) y bajo acoplamiento (una mínima relación entre componentes únicamente la necesaria). Con estas características la arquitectura se presenta como sumamente modular.
2. Otra ventaja presente es el orden de acceso, los datos no pueden ser manipulados directamente por las aplicaciones clientes, esto colabora en el tema de seguridad de la aplicación. Permite una adecuada validación de los datos antes de ser almacenados.
3. Con esta arquitectura es posible que una vez definidas las tareas se realicen varias de éstas en paralelo, por parte del equipo desarrollador, debido a la independencia de los módulos.
4. Durante todo el proceso la arquitectura establece una serie de actividades paralelas como los son: seguridad, administración del funcionamiento y comunicación. Estas actividades colaboran con un proceso adecuado de ingeniería de software en todo el desarrollo.

5. Diseño al Nivel de Subsistemas

En esta sección se describirá algunos de los subsistemas que se diseñaron para el Sistema AWMI. Los subsistemas presentes en el sistema, con su correspondiente descripción, datos y detalle general de cada uno se presentan a continuación.

5.1 Identificación de los Subsistemas

Seguidamente se presenta a manera de ejemplo algunos de los subsistemas que se diseñaron como parte del sistema SAWMI:

- **Subsistema IMEC Producto**
  Clases involucradas:
  - Formulario Producto
  - Controladora Producto
  - Controladora Base de Datos
  - Controladora Proveedor
  - Producto

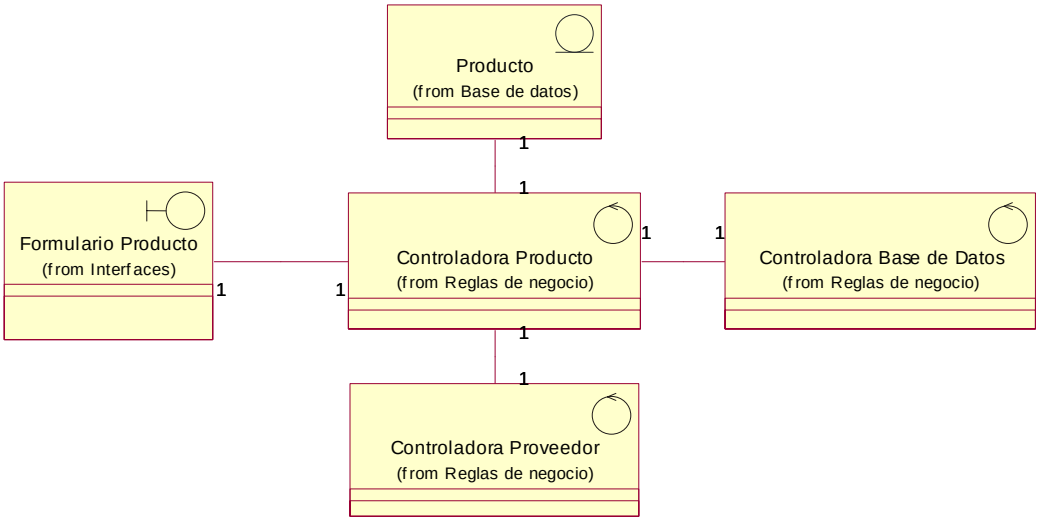


Figura 3. Diagrama de Clases del susbsistema IMEC Productos.

- **Subsistema IMEC Proveedor**
  Clases involucradas:
  - Formulario Proveedor
  - Controladora Proveedor
  - Controladora Base de Datos
  - Proveedor

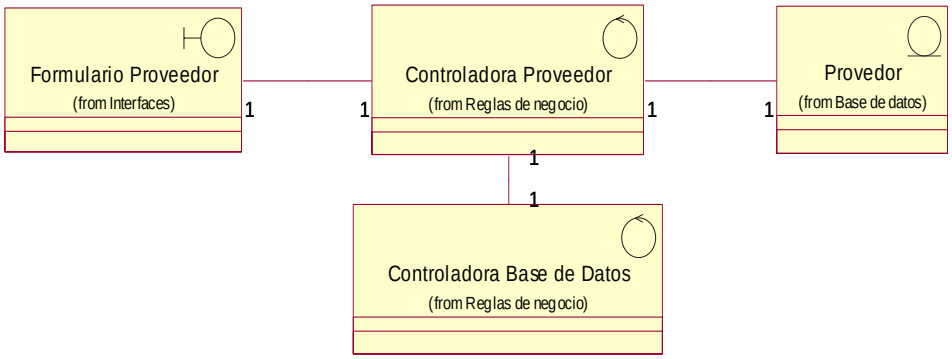


Figura 4. Diagrama de Clases del susbsistema IMEC Proveedor.

- **Subsistema IMEC Proforma**
  Clases involucradas:
  - Formulario Proforma
  - Controladora Proforma
  - Controladora Producto
  - Controladora Miembro
  - Controladora Base de Datos
  - Proforma
  - Detalle Proforma

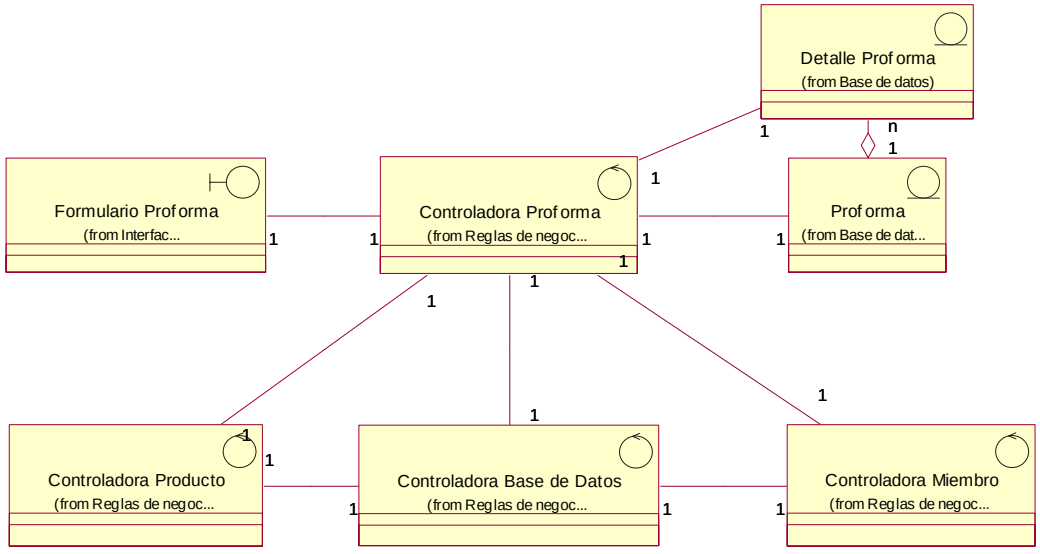


Figura 5. Diagrama de Clases del susbsistema IMEC Proformas.

- **Subsistema IMEC Factura**
  Clases involucradas:
  - Formulario Factura
  - Controladora Factura
  - Controladora Producto
  - Controladora Miembro
  - Controladora Base de Datos
  - Factura
  - Detalle Factura

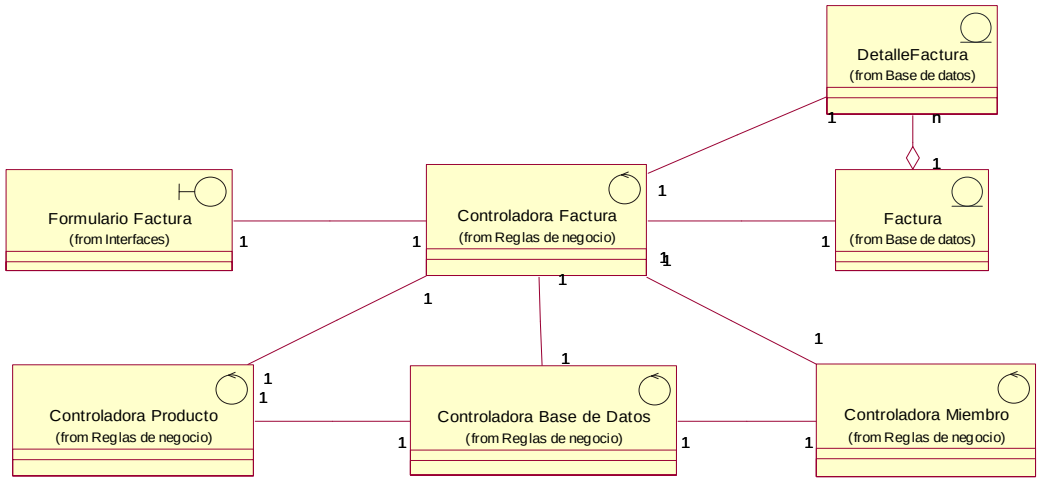


Figura 6. Diagrama de Clases del susbsistema IMEC Factura.

5.2 Diseño de Mensajes

A continuación se muestran los mensajes entre las clases de algunos de los subsistemas que se diseñaron para SAWMI.

- **Subsistema IMEC Producto**
  - Formulario Producto ←→ Controladora Producto
    - *Relación existente*: una relación uno a uno entre la clase Formulario Producto y la Controladora Producto. La clase Formulario Producto mediante el método solicitarDatos Producto (cedula) utiliza los datos que le devuelve la clase Controladora Producto para llenar el ComboBox con todos los Productos.
    - *Métodos*: la clase Controladora Producto utiliza los datos recibidos por la clase Formulario Producto para insertar, modificar, eliminar y consultar mediante los métodos: Guardar Producto (), Modificar Producto (), Eliminar Producto () y SolicitarDatos Producto () respectivamente. El método SolicitarDatos Producto () devuelve los datos del Producto consultado.
    - *Atributos*: el atributo necesario para consultar, modificar, eliminar es id_Producto.
  - Controladora Producto ←→ Controladora Base de Datos
    - *Relación existente*: una relación uno a uno entre la clase Controladora Producto y la Controladora Base de Datos.
    - *Métodos*: la clase Controladora Producto Utiliza los datos recibidos por la clase Formulario Producto y mediante los métodos: GuardarUsuario(), ModificarUsuario (), EliminarUsuario () y ConsultarUsuario ejecuta los llamados a la Controladora de Base de Datos. La Controladora de Base de Datos utiliza el metodo devuelveDatos Producto () para devolver los datos solicitados mediante una consulta y con el método devuelveMensaje(éxito Fracaso) responde a los llamados insertar, modificar y eliminar.
    - *Atributos*: el atributo necesario para consultar, modificar, eliminar es el id_Producto.
  - Controladora Producto ←→ Controladora Proveedor
    - *Relación existente*: una relación uno a uno entre la clase Controladora Producto y la Controladora Proveedor. Necesaria para la creación de un Producto ya que un Producto pertenece a un único Proveedor.
  - Controladora Producto ←→ Producto
    - *Relación existente*: una relación uno a uno entre la clase Controladora Producto y la clase constructora Producto.
    - *Métodos*: los métodos de la clase controladora Guardar Producto(), Modificar Producto(), Eliminar Producto() y SolicitarDatos Producto() trabajan sobre una sola instancia de un objeto de tipo Producto a la vez y utilizan los atributos de la clase Producto para enviar y recibir la información de cada Producto
    - *Atributos*: de la clase Producto son: id_Producto, Nombre, Id_Proveedor, Tipo, Precio, Cantidad_Inventario, Descripción, PrecioEntrada.
- **Subsistema IMEC Proveedor**
  - Formulario Proveedor ←→ Controladora Proveedor

- *Relación existente*: una relación uno a uno entre la clase Formulario Proveedor y la Controladora Proveedor. La clase Formulario Proveedor mediante el método solicitarDatos Proveedor(idProveedor) utiliza los datos que le devuelve la clase Controladora Proveedor para llenar el ComboBox con todos los Proveedores
- *Métodos*: la clase Controladora Proveedor Utiliza los datos recibidos por la clase Formulario Proveedor para insertar, modificar, eliminar y consultar mediante los métodos: Guardar Proveedor(), Modificar Proveedor(), Eliminar Proveedor() y SolicitarDatos Proveedor() respectivamente. El método SolicitarDatos Proveedor() devuelve los datos del Proveedor consultado.
- *Atributos*: el atributo necesario para consultar, modificar, eliminar e insertar es el id_Proveedor.

- Controladora Proveedor ←→ Controladora Base de Datos
  - *Relación existente*: una relación uno a uno entre la clase Controladora Proveedor y la Controladora Base de Datos.
  - *Métodos*: la clase Controladora Proveedor Utiliza los datos recibidos por la clase Formulario Proveedor y mediante los métodos: Guardar Proveedor(), Modificar Proveedor(), Eliminar Proveedor() y Consultar Proveedor() ejecuta los llamados a la Controladora de Base de Datos. La Controladora de Base de Datos utiliza el metodo devuelveDatos Proveedo() para devolver los datos solicitados mediante una consulta y con el método devuelveMensaje(éxito Fracaso) responde a los llamados insertar, modificar y eliminar.
  - *Atributos*: el atributo necesario para consultar, modificar, eliminar e insertar es el id_Proveedor.
- Controladora Miembro ←→ Miembro
  - *Relación existente*: una relación uno a uno entre la clase Controladora Proveedor y la clase constructora Proveedor.
  - *Métodos*: los métodos de la clase controladora Guardar Proveedor(), Modificar Proveedor(), Eliminar Proveedor() y SolicitarDatos Proveedor() trabajan sobre una sola instancia de un objeto de tipo Proveedor a la vez y utilizan los atributos de la clase Proveedor para enviar y recibir la información de cada Proveedor.
  - *Atributos*: de la clase Proveedor son id_Proveedor, Nombre, CedulaJuridica, Correo, Representante, Telefono, Direccion.

- **Subsistema IMEC Proforma**
  - Formulario Proforma ←→ Controladora Proforma
    - *Relación existente*: una relación uno a uno entre la clase Formulario Proforma y la Controladora Proforma. La clase Formulario Proforma mediante el método solicitarDatos Proforma (id_Proforma) utiliza los datos que le devuelve la clase Controladora Proforma para llenar el ComboBox con todos las Proformas.
    - *Métodos*: la clase Controladora Proforma utiliza los datos recibidos por la clase Formulario Proforma para insertar, modificar, eliminar y consultar mediante los métodos: Guardar Proforma(), Modificar Proforma(), Eliminar Proforma() y SolicitarDatos Proforma() respectivamente. El método SolicitarDatos Proforma() devuelve los datos de la Proforma consultada.

        - *Atributos*: el atributo necesario para consultar, modificar, eliminar es id_Proforma.
    - Controladora Proforma ←→ Controladora Base de Datos
        - *Relación existente*: una relación uno a uno entre la clase Controladora Proforma y la Controladora Base de Datos.
        - *Métodos*: la clase Controladora Proforma Utiliza los datos recibidos por la clase Formulario Proforma y mediante los métodos: GuardarUsuario(), Modificar Proforma(), Eliminar Proforma() y Consultar Proforma() ejecuta los llamados a la Controladora de Base de Datos. La Controladora de Base de Datos utiliza el metodo devuelveDatos Proforma() para devolver los datos solicitados mediante una consulta y con el método devuelveMensaje(éxito Fracaso) responde a los llamados insertar, modificar y eliminar.
        - *Atributos*: el atributo necesario para consultar, modificar, eliminar es el id_Proforma.
    - Controladora Proforma ←→ Controladora Miembro
        - *Relación existente*: una relación uno a uno entre la clase Controladora Proforma y la Controladora Miembro. Necesaria para la creación de una Proforma ya que a cada Proforma se le asocia un único Miembro (cliente). Además se le asocia un responsable empleado el cual también es Miembro.
    - Controladora Proforma ←→ Controladora Producto
        - *Relación existente*: una relación uno a n entre la clase Controladora Proforma y la Controladora Producto. Necesaria porque la Proforma contiene Productos y estos se encuentra en un detalleProforma.
    - Controladora Proforma ←→ Proforma
        - *Relación existente*: una relación uno a uno entre la clase Controladora Proforma y la clase constructora Proforma.
        - *Métodos*: los métodos de la clase controladora Guardar Proforma(), Modificar Proforma(), Eliminar Proforma() y SolicitarDatos Proforma() trabajan sobre una sola instancia de un objeto de tipo Proforma a la vez y utilizan los atributos de la clase Proforma para enviar y recibir la información de cada Proforma
        - *Atributos*: de la clase Proforma son: id_Proforma, CedulaCliente, Fecha, IdResponsable.
    - Controladora Proforma ←→ DetalleProforma
        - *Relación existente*: una relación uno a uno entre la clase Controladora Proforma y la clase constructora DetalleProforma. Se requiere relación pues cada Proforma tiene asociado un detalle de Productos.
        - *Atributos*: de la clase DetalleProforma son: IdProducto, Descuento, precio, idProforma, Cantidad.
- **Subsistema IMEC Factura**
    - Formulario Factura ←→ Controladora Factura
        - *Relación existente*: una relación uno a uno entre la clase Formulario Factura y la Controladora Factura. La clase Formulario Factura mediante el método solicitarDatos Factura(id_Factura) utiliza los datos que le devuelve la clase Controladora Factura para llenar el ComboBox con todos las Facturas.
        - *Métodos*: la clase Controladora Factura utiliza los datos recibidos por la clase Formulario Factura para insertar, modificar, eliminar y consultar mediante los

métodos: Guardar Factura(), Modificar Factura(), Eliminar Factura() y SolicitarDatos Factura() respectivamente. El método SolicitarDatos Factura() devuelve los datos de la Factura consultada.
    - *Atributos*: el atributo necesario para consultar, modificar, eliminar es id_Factura.
- Controladora Factura ←→ Controladora Base de Datos
    - *Relación existente*: una relación uno a uno entre la clase Controladora Factura y la Controladora Base de Datos.
    - *Métodos*: la clase Controladora Factura Utiliza los datos recibidos por la clase Formulario Factura y mediante los métodos: GuardarFactura(), Modificar Factura(), Eliminar Factura() y ConsultarFactura() ejecuta los llamados a la Controladora de Base de Datos. La Controladora de Base de Datos utiliza el metodo devuelveDatos Factura() para devolver los datos solicitados mediante una consulta y con el método devuelveMensaje(éxito Fracaso) responde a los llamados insertar, modificar y eliminar.
    - *Atributos*: el atributo necesario para consultar, modificar, eliminar es el id_Factura.
- Controladora Factura ←→ Controladora Miembro
    - *Relación existente*: una relación uno a uno entre la clase Controladora Factura y la Controladora Miembro. Necesaria para la creación de una Factura ya que a cada Factura se le asocia un único Miembro (cliente). Además se le asocia un responsable empleado el cual también es Miembro.
- Controladora Factura ←→ Controladora Producto
    - *Relación existente*: una relación uno a n entre la clase Controladora Factura y la Controladora Producto. Necesaria porque la Factura contiene Productos y estos se encuentra en un detalleFactura.
- Controladora Factura ←→ Factura
    - *Relación existente*: una relación uno a uno entre la clase Controladora Factura y la clase constructora Factura.
    - *Métodos*: los métodos de la clase controladora Guardar Factura(), Modificar Factura(), Eliminar Factura() y SolicitarDatos Factura() trabajan sobre una sola instancia de un objeto de tipo Factura a la vez y utilizan los atributos de la clase Factura para enviar y recibir la información de cada Factura
    - *Atributos*: de la clase Factura son: id_Factura, CedulaCliente, Fecha, IdResponsable.
- Controladora Factura ←→ DetalleFactura
    - *Relación existente*: una relación uno a uno entre la clase Controladora Factura y la clase constructora DetalleFactura. Se requiere relación pues cada Proforma tiene asociado un detalle de Productos.
    - *Atributos*: de la clase DetalleFactura son: IdFactura, IdProducto, Descuento, precio, Cantidad.

5.3 Diseño de Responsabilidades

Seguidamente se realiza una descripción de los métodos de cada clase (Controladora, Formularios, Entidades).

Clases Formularios

Métodos Generales Clases Formularios

- *validarDatos():* Se encarga de comprobar que los datos que el usuario colocó para algunas de las tareas de modificar, eliminar, consultar e insertar son válidos(es posibles colocarlos)
- *mensajeConfirmación(exitoFracaso):* se devuelve un mensaje al usuario indicando el resultado de la acción que realizó, ya sea exitoso o si se produjo un error indicada las condiciones del error y cómo se puede salir de este.
- *actualizarDataGrid(datos):* Actualiza los datos del DataGrid con los datos que se devuelven de la Controladora, la actualización se hace de acuerdo a los cambios que ocurren en la base de datos.
- *limpiaCampos():* Se encarga de dejar en blanco todos los campos que tenga el formulario, por motivo de realizar una nueva inserción. Este método se ejecuta una vez que se confirmó una inserción adecuada.
- *cargarDataGrid():* Se realiza una solicitud a la controladora correspondiente y luego se toma el resultado que esta controladora devuelve, estos datos deben ser colocados en un DataGrid para ser mostrados al usuario.

Métodos Específicos Clases Formularios

Formulario_Producto

- *SolicitaDatos Producto(Cedula):* Envía a realizar una consulta de los datos de un Producto, este método realiza el llamado a la controladora Producto para que ejecute la consulta.
- *guardar Producto(idProducto, nombre, idProveedor, Tipo, precio, CantidadInventario, Descripción, PrecioEntrada):* Realiza la ejecución de una inserción de un Producto, realiza el llamado a la controladora Producto para que ejecute la inserción.
- *eliminar Producto(idProducto):* Envía el llamado a la controladora Producto para que ejecute la eliminación del Producto que tiene el idProducto enviado como parámetros.
- *modificar Producto(, nombre, idProveedor, Tipo, precio, CantidadInventario, Descripción, PrecioEntrada):* Realiza el llamado a la Controladora Producto para que se ejecute la modificación del Producto seleccionado, se envían como parámetros los nuevos datos que se desea colocar para este Producto. Es requisito enviar el idProducto.

Formulario_Proveedor

- *SolicitaDatosProveedor(idProveedor):* Envía a realizar una consulta de los datos de un Proveedor, este método realiza el llamado a la controladora Proveedor para que ejecute la consulta.
- *guardarProveedor(idProveedor, nombre, CedulaJuridica, Corrreo, Representante, Teléfono, dirección):* Realiza la ejecución de una inserción de un Proveedor, realiza el llamado a la controladora Proveedor para que ejecute la inserción.
- *eliminarProveedor (idProveedor):* Envía el llamado a la controladora Proveedor para que ejecute la eliminación del Proveedor que tiene el idProveedor enviado como parámetros.

- *modificarProveedor(idProveedor, nombre, CedulaJuridica, Corrreo, Representante, Teléfono, dirección):* Realiza el llamado a la Controladora Proveedor para que se ejecute la modificación del Proveedor seleccionado, se envían como parámetros los nuevos datos que se desea colocar para este Proveedor. Es requisito enviar el idProveedor.

Formulario_Factura

- *SolicitaDatosFactura(id_Factura):* Envía a realizar una consulta de los datos de una Factura, este método realiza el llamado a la controladora Factura para que ejecute la consulta.
- *guardarDetalleFactura(id_factura, id_Producto, precio,descuento):* Realiza la ejecución del guardado del detalle de un producto que esta dentro de una factura.
- *guardarFactura(id_Factura, cedulaCliente, Fecha,idResponsable):* Realiza la ejecución de una inserción de una Factura, realiza el llamado a la controladora Factura para que ejecute la inserción de una factura.
- *eliminarFactura (id_Factura):* Envía el llamado a la controladora Factura para que ejecute la eliminación de la Factura que tiene el id_Factura enviado como parámetros.

Formulario_Proforma

- *SolicitaDatosProforma(idProforma):* Envía a realizar una consulta de los datos de una Proforma, este método realiza el llamado a la controladora Proforma para que ejecute la consulta.
- *guardarProforma(idProforma, cedulaCliente, Fecha, IdResponsable):* Realiza la ejecución de una inserción de una Proforma, realiza el llamado a la controladora Proforma para que ejecute la inserción.
- *eliminarProforma(idProforma):* Envía el llamado a la controladora Proforma para que ejecute la eliminación de la proforma que tiene el idProforma enviado como parámetros.
- *modificarProforma(cédula idProforma, cedulaCliente, Fecha, IdResponsable):* Realiza el llamado a la Controladora Proforma para que se ejecute la modificación de la Proforma seleccionado, se envían como parámetros los nuevos datos que se desea colocar para esta Proforma. Es requisito enviar el idProforma.

Clases Controladoras

Métodos Generales Clases Controladoras

- *SolicitaDatosObjeto*(idObjeto**):* Envía a realizar una consulta realizando un llamado a la ControladoraBaseDeDatos.
- *guardarObjeto*(datosObjeto***):* Realiza la ejecución de una inserción realiza el llamado a la controladora de Base de Datos para que ejecute la inserción.
- *eliminarObjeto*(idObjeto**):* Envía el llamado a la controladora Base de Datos para que ejecute la eliminación del objeto con el id enviado.
- *modificarObjeto*(datosObjeto***):* Realiza el llamado a la Controladora Base de Datos para que se ejecute la modificación del objeto seleccionado, se envían como parámetros los nuevos datos que se desea colocar para este objeto. Es requisito enviar el campos llave del objeto.
- *encapsularObjeto*(datosObjeto***):* Es un llamado a la clase entidad para que cree un objeto con los datos enviados como parámetros. Esto se hace cuando se realiza una consulta y contiene muchos campos.
- *crearObjeto(datosObjeto***):* Se realiza un llamado a la clase entidad para que cree un objeto con los datos enviados como parámetros. Esto se hace cuando se realiza una inserción antes de comunicarse con la base de Datos se crea el objeto y luego se envia a la Controladora de Base de datos.

Clases Entidad
Métodos Generales Clases Entidad

- *crearObjeto(datosObjeto***):* La entidad crea el objeto correspondiente con los datos que recibe como parámetros.
- *encapsularObjeto*(datosObjeto***):* Recibe un conjunto de datos y realiza la creación de un objeto a partir de los parámetros recibidos.
- devuelveObjeto*(Objeto*): Devuelve *el objeto que se creo.*

6. Diseño de Mensajes entre Subsistemas

6.1 Diagrama de Clases Global

A continuación se presenta el Diagramas de clases global para el Sistema SAWMI:

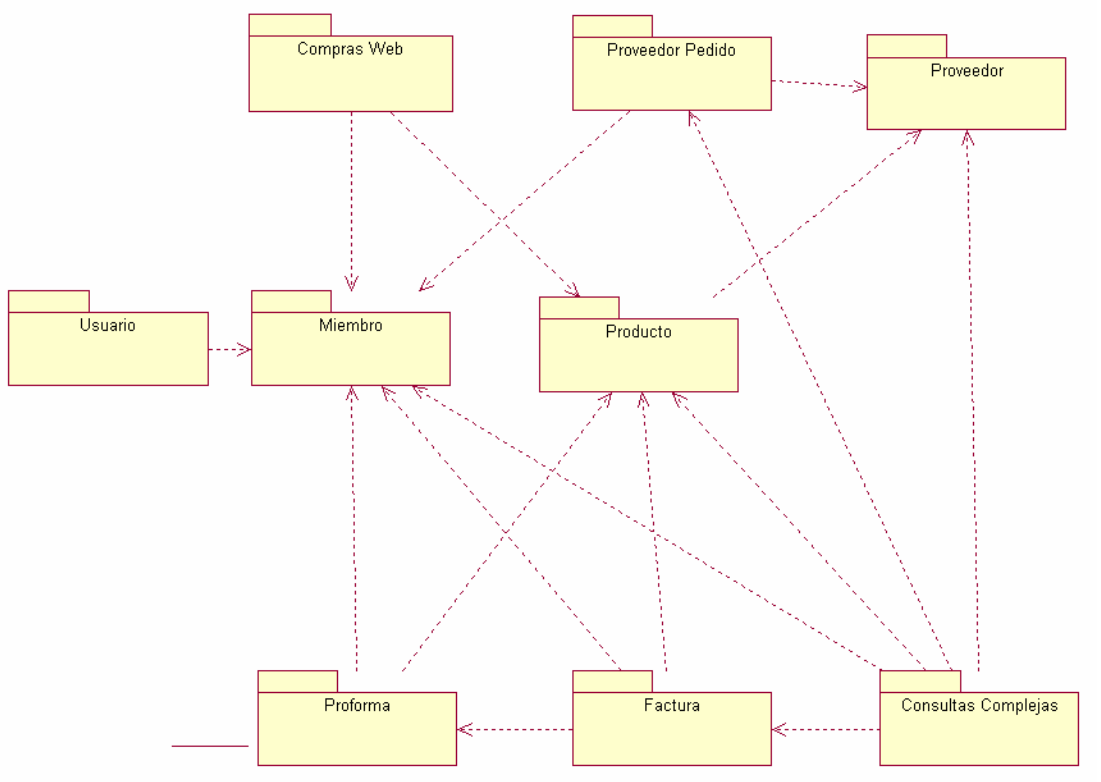


Figura 7. Diagrama de Clases global para el Sistema SAWMI.

6.2 Descripción de Mensajes entre los Subsistemas

Con base en el Diagrama de Clases Global mostrado en la figura 7 se describen los mensajes entre subsistemas:

Subsistema IMEC_Usuario ↔ Subsistema IMEC_Miembro
El subsistema IMEC_Usuario se comunica con el subsistema IMEC_Miembro con el fin de que en el momento en que se inserte o se modifique un miembro, a este se le pueda hacer un usuario para el sistema. Igualmente si le elimina el miembro, se elimina el usuario.

Subsistema IMEC_ConsultasComplejas ↔ Subsistema IMEC_Miembro.
El subsistema IMEC_ConsultasComplejas se comunica con el subsistema MEC_Miembro con el fin de que este subsistema tenga acceso al responsable en el caso de una factura, proforma o pedido, en el caso de las consultas 1, 2 Y 3, Y en el caso de la consulta 5 para llenar el combobox del cual se toma la referencia para hacer la consulta

Subsistema IMEC_ConsultasComplejas ↔ Subsistema IMEC_Producto
El subsistema IMEC_ConsultasComplejas se comunica con el subsistema IMEC_Producto con el fin de extraer los nombres de los productos (dado que estos se almacenan en la base de datos con el IdProduto) que se utilizan en las consultas 1,2,3, 5, 7 Y 8, Y en el caso de la consulta 7 para llenar el combobox del cual se toma la referencia para hacer la consulta

Subsistema IMEC_ConsultasComplejas ↔ Subsistema IMEC_Proveedor
El subsistema IMEC_ConsultasComplejas se comunica con el subsistema IMEC_Proveedor para extraer los nombres de los proveedores que se utilizan en las consultas 3 y 6, Y en el caso de las consultas 4, 5 Y 7 para llenar el combobox del cual se toma la referencia para hacer la consulta.

Subsistema IMEC _ ConsultasCom plejas ↔ Subsistema IMEC _Factura
El subsistema IMEC_ConsultasComplejas se comunica con el subsistema MEC_Factura con el fin de extraer los datos que se utilizan en los datagrid de las consultas 1, 2, 3, 7 Y 8, Y para hacer las comparaciones necesarias en el caso de las consultas 1, 2, 3, 7 Y 8.

Subsistema IMEC_ConsultasComplejas ↔ Subsistema IMEC_Pedido
El subsistema IMEC_ConsultasComplejas se comunica con el subsistema IMEC_Pedido con el fin de extraer los datos que se utilizan en los datagrid de las consultas 5 y 6, Y para hacer las comparaciones necesarias en el caso de estas mismas consultas.

Subsistema IMEC_Proforma.-. Subsistema IMEC_Miembro
El subsistema IMEC Profonna se comunica con el subsistema IMEC Miembro con el fin de extraer los datos que se utilizan para llenar el combobox de clientes y también en el momento de crear el detalle.

Subsistema IMEC_Proforma ↔ Subsistema IMEC_Producto
El subsistema IMEC Profonna se comunica con el subsistema IMEC Producto con el fin de extraer los datos que se utilizan para llenar el datagrid de los productos disponibles, también a la hora de crear el detalle local, y el detalle que se creará en la base de datos.

Subsistema IMEC Factura ↔ Subsistema IMEC Miembro
El subsistema IMEC Profonna se comunica con el subsistema IMEC Miembro con el fin de extraer los datos que se utilizan para llenar el campo del responsable (empleado) y

para extraer los datos necesarios (cedula, nombre, lapellido, 2apellido) para llenar el combobox de clientes.

Subsistema IMEC Factura ↔ Subsistema IMEC Proforma
El subsistema IMEC Factura se comunica con el subsistema IMEC Profonna con el fin de extraer los datos que se necesiten de la profonna en el caso de que la factura se lleve a cabo en función de una proforma (que esta se base en los datos de una proforma).

Subsistema IMEC_Factura ↔ Subsistema IMEC_Producto
El subsistema IMEC_Factura se comunica con el subsistema IMEC_Producto con el fin de extraer los datos que se utilizan para llenar el datagrid de los productos disponibles, también a la hora de crear el detalle local, y el detalle que se creará en la base de datos.

Subsistema IMEC_ComprasWeb ↔ Subsistema IMEC_Producto
El subsistema IMEC_ComprasWeb se comunica con el subsistema IMEC_Producto con el fin de extraer los datos que se utilizan para llenar el datagrid de los productos disponibles, también a la hora de crear el detalle local (carrito de compras), y el detalle que se creará en la base de datos.

Subsistema IMEC_ComprasWeb ↔ Subsistema IMEC_Miembro
El subsistema IMEC_ComprasWeb se comunica con el subsistema IMEC_Miembro con el fin de extraer los datos que se utilizan para la validación del usuario y para obtener los datos necesarios a la hora de crear la factura.

Subsistema IMEC_Producto ↔ Subsistema IMEC_Proveedor
El subsistema IMEC_Producto se comunica con el subsistema IMEC_Proveedor para llenar el combobox de proveedores y después para llenar el campo respectivo en la tabla de producto.

Subsistema IMEC_ ProveedorPedido ↔ Subsistema IMEC_Proveedor
El subsistema IMEC ProveedorPedido se comunica con el subsistema IMEC Proveedor con el fin de extraer los datos que se utilizan para llenar el combobox de los proveedores disponibles.

Subsistema IMEC_ ProveedorPedido ↔ Subsistema IMEC_Producto
El subsistema IMEC ProveedorPedido se comunica con el subsistema IMEC Producto con el fin de extraer los datos que se utilizan para llenar el datagrid de los productos disponibles, también a la hora de crear el detalle local, y el detalle que se creará en la base de datos.

Subsistema IMEC_ ProveedorPedido ↔ Subsistema IMEC_Miembro
El subsistema IMEC_ProveedorPedido se comunica con el subsistema IMEC_Miembro con el fin de extraer los datos que se utilizan para llenar el campo del responsable (empleado)

7. Diseño de la Interfaz de Usuario

7.1 Diagrama de Navegación de Pantallas

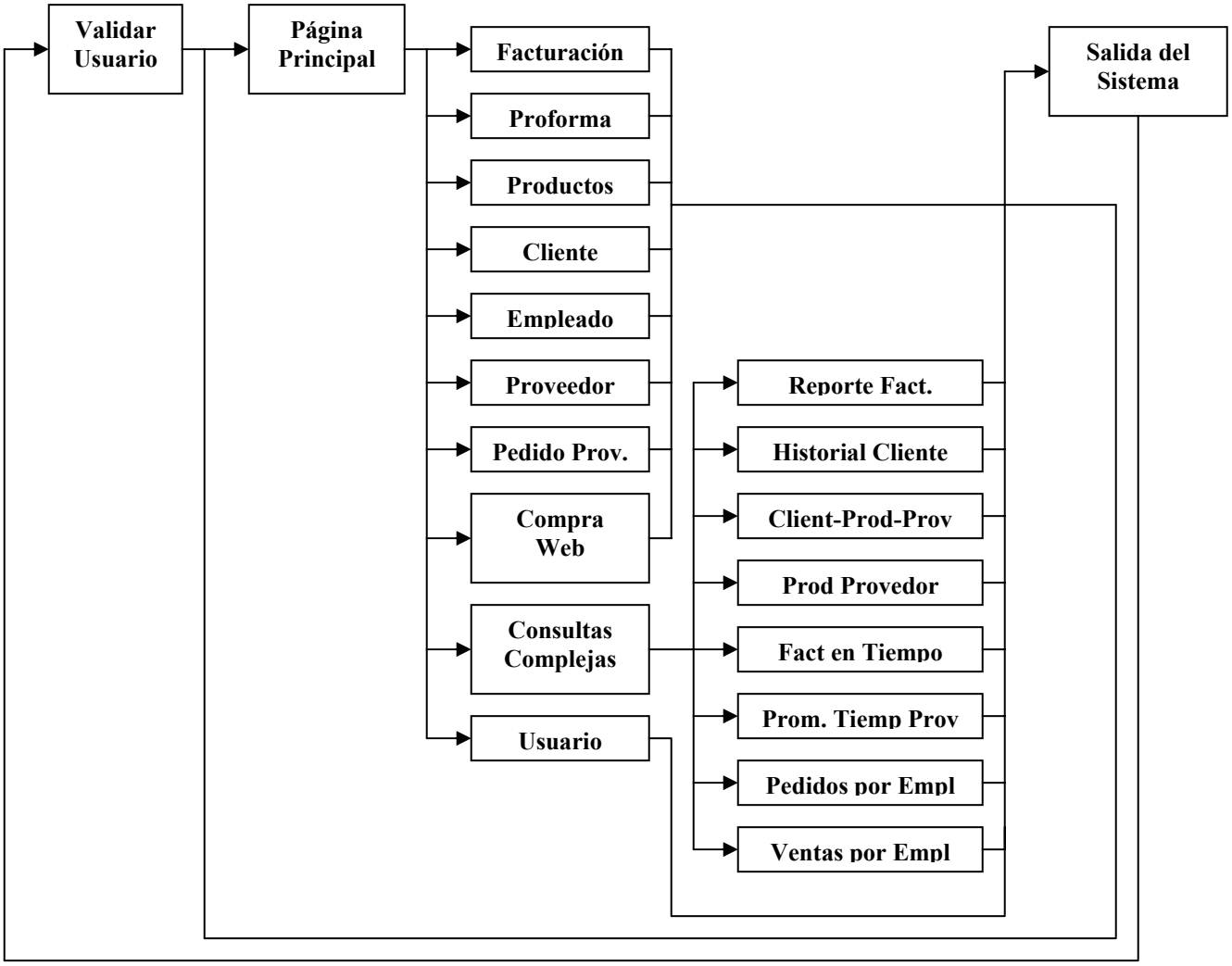


Figura 8. Diagrama de navegación entre pantallas para el Sistema SAWMI

7.2 Descripción de las Pantallas

A continuación se presentan a manera de ejemplo algunas de las pantallas que se diseñaron para el sistema SAWMI.

Pantalla de Validación:

Figura 9. Pantalla de validación del Sistema SAWMI.

Pantalla simple, que solicita el nombre de inicio y contraseña del usuario, cuenta con el botón aceptar (1), para introducir y chequear los datos (se verifican sean validos, o desplegará mensaje de error). Una vez realizado esto se llama a la pantalla principal del Sistema.

Menú de Navegación

Figura 10. Menú que permite la navegación entre los diferentes módulos del Sistema SAWMI.

El menú de navegación se mantiene a lo largo de las diversas pantallas del sistema, posee enlaces a los diversos módulos del sistema: Facturación, Preformas, Productos, Clientes,

Empleados, Proveedor, Pedidos a Proveedor, Compras Web, Consultas Complejas y Finalmente Usuarios del Sistema (1).

IMEC Producto

Figura 11. Pantalla del módulo Mantenimiento de Productos.

Existen tres botones en la parte superior derecha de la pantalla que permite llevar hacia las diversas funciones propias del Producto (2) – Insertar, Eliminar o Modificar -. Existe un datagrid (3) que permite visualizar los diferentes Productos existentes, a la vez que una serie de labels con los atributos del producto (4) propuestos para poder visualizar o editar la información según sea el caso. Finalmente se presentan los botones Aceptar y Cancelar en la parte inferior de la pantalla (1) para aprobar o rechazar las acciones realizadas y borrar los campos.

IMEC Proveedor

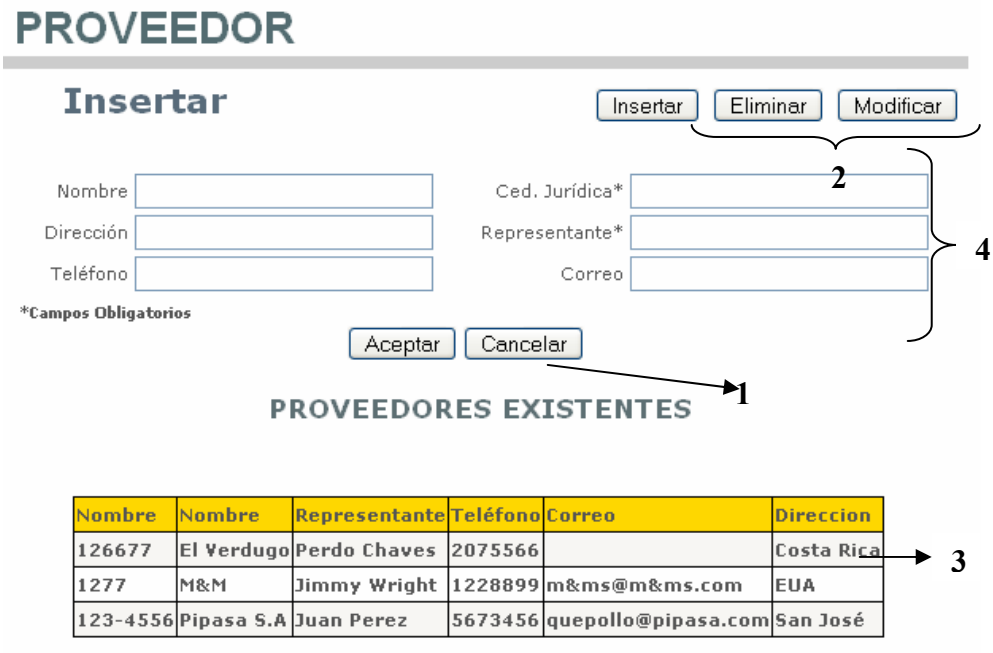


Figura 12. Pantalla del módulo Mantenimiento de Proveedores.

Existen tres botones en la parte superior derecha de la pantalla que permite llevar hacia las diversas funciones propias del Proveedor (2) – Insertar, Eliminar o Modificar -. Existe un datagrid (3) que permite visualizar los diferentes Proveedores existentes, a la vez que una serie de labels con los atributos del proveedor (4) propuestos para poder visualizar o editar la información según sea el caso. Finalmente se presentan los botones Aceptar y Cancelar en la parte inferior de la pantalla (1) para aprobar o rechazar las acciones realizadas y borrar los campos.

IMEC Factura

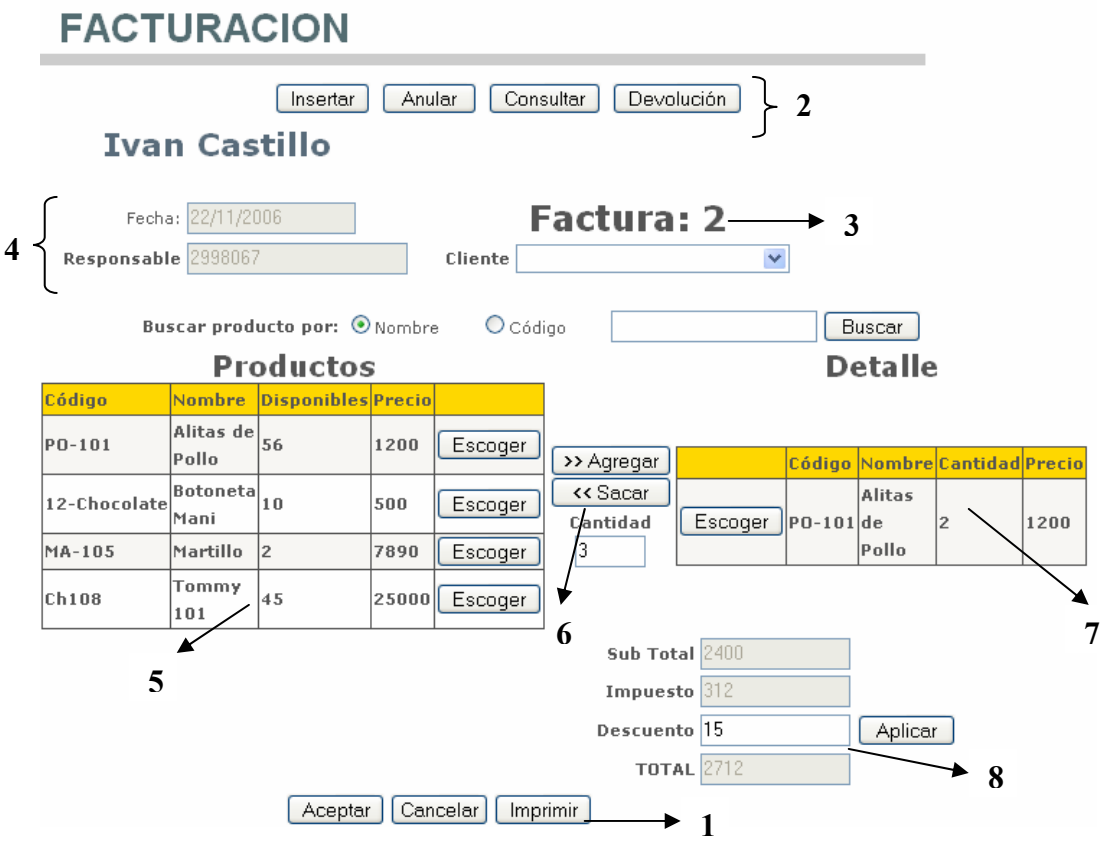


Figura 13. Pantalla del módulo Mantenimiento de Facturas.

Existen tres botones en la parte superior derecha de la pantalla que permite llevar hacia las diversas funciones propias la Factura (2) – Insertar, Anular o Consultar -. El número secuencial de la factura actual se muestra en la parte superior derecha de la pantalla (3). Una serie de labels con los atributos de la Factura (4) se presentan para poder visualizar o editar la información según sea el caso, al igual que decidir si la búsqueda se realizará por Nombre o Código del producto. En el datagrid de la parte izquierda de la pantalla (5) se presentan los diversos productos existentes que concuerdan con la búsqueda del usuario, en el de la derecha (7) los productos ya incluidos en la factura del usuario, para agregar o eliminar productos de la factura se utilizan los botones delimitados con lo símbolos >> (Agregar) o << (Eliminar) (6). En la parte derecha de la pantalla (8) se muestra el Subtotal, Descuento y Precio total de la compra actual; finalmente se presentan los botones Aceptar y Cancelar en la parte inferior de la pantalla (1) para aprobar o rechazar las acciones realizadas y borrar los campos.

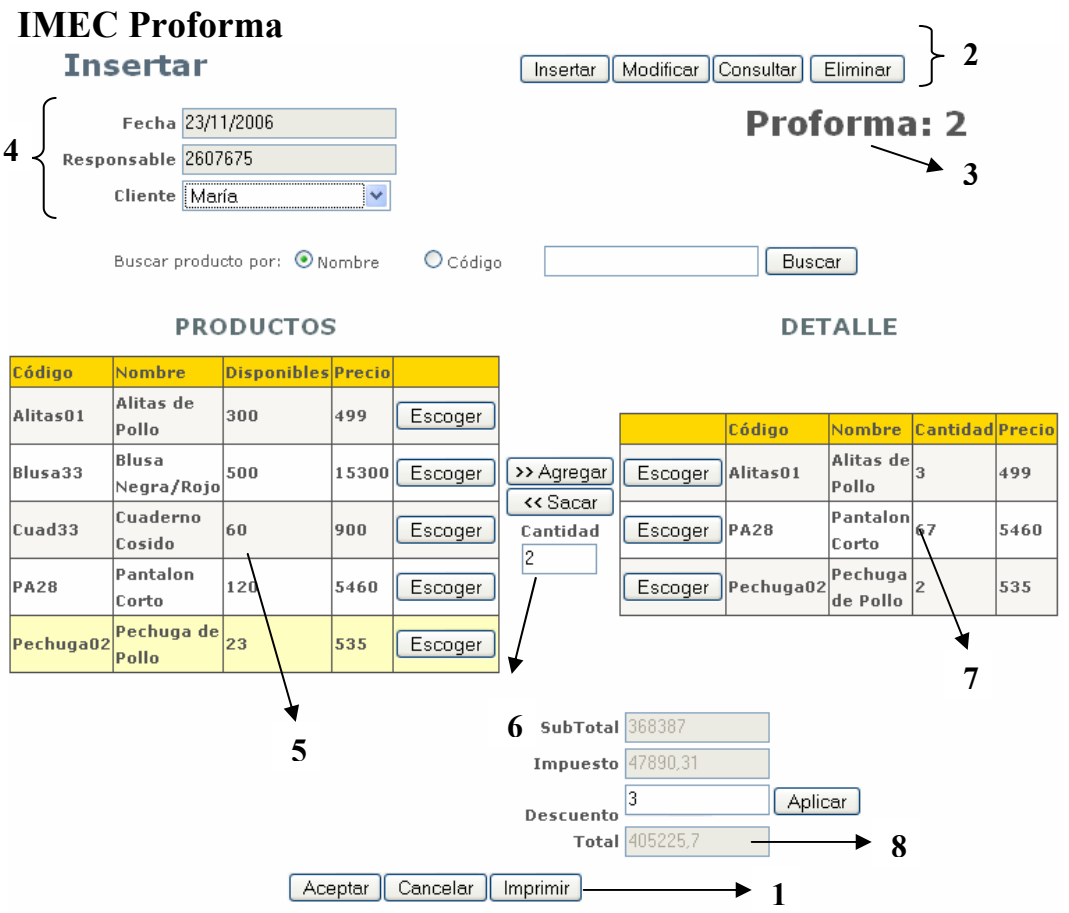


Figura 14. Pantalla del módulo Mantenimiento de Proformas.

Existen cuatro botones en la parte superior derecha de la pantalla que permite llevar hacia las diversas funciones propias la Proforma (2) – Insertar, Modificar, Eliminar o Consultar -. El número secuencial de la proforma actual se muestra en la parte superior derecha de la pantalla (3). Una serie de labels con los atributos de la Proforma (4) se presentan para poder visualizar o editar la información según sea el caso, al igual que decidir si la búsqueda se realizará por Nombre o Código del producto. En el datagrid de la parte izquierda de la pantalla (5) se presentan los diversos productos existentes que concuerdan con la búsqueda del usuario, en el de la derecha (7) los productos ya incluidos en la proforma del usuario, para agregar o eliminar productos de la proforma se utilizan los botones delimitados con lo símbolos >> (Agregar) o << (Eliminar) (6). En la parte derecha de la pantalla (8) se muestra el monto total actual de la proforma; finalmente se presentan los botones Aceptar y Cancelar en la parte inferior de la pantalla (1) para aprobar o rechazar las acciones realizadas y borrar los campos.

8. Aspectos de Pruebas

8.1 Clases de Pruebas

Para la realización de las pruebas se siguieron diferentes estrategias según sea el alcance deseado.

Pruebas de Unidad:

En estas pruebas se decidió seguir una estrategia de caja blanca, esto debido a que al ser los módulos individuales pareció correcto revisarlos más extenuantemente los códigos, debido a que de la revisión y correctitud de estos dependerá la eficacia del sistema una vez que se este integrando. Se opto por la estrategia de caja blanca determinada prueba flujo de datos, la cual selecciona caminos de prueba de acuerdo a la ubicación de las definiciones y los usos de las variables.

Pruebas de Integración:

Para las pruebas de integración se opto por utilizar pruebas de caja negra, ya que en la etapa anterior se revisó el código de las diferentes unidades. En esta etapa se da principal énfasis en la eficacia del sistema y no del código anteriormente escrito. En este caso se opto por utilizar pruebas valor límite, en la cual se utilizan valores límites para realizar pruebas sobre el sistema.

Pruebas de Validación

Para esta parte se decidió utilizar pruebas de caja negra. Se realizará una verificación por escenarios, con las cuales se van a verificar todos los requerimientos establecidos siguiendo el flujo normal del usuario, los cuales van a ejercitar varios subsistemas en una sola prueba

Pruebas de Sistema

Para esta parte se decidió utilizar pruebas de caja negra. Se utilizarán pruebas de estructura superficie, la cual es la estructura observable externamente de un programa orientado a objetos. Se le brindan objetos al usuario para que los manipule, y es realizada a través de la interfaz. Se basa en las tareas del usuario.

8.2 Componentes Críticos

Se estableció como componentes críticos los módulos de **compra** y **venta**, ya que son los que incluyen mayor cantidad de objetos relacionados y otorgan una mayor funcionalidad. Para cada uno de los elementos críticos se realizó anteriormente una descripción mediante diagramas de flujo.

8.3 Casos de Prueba

Para ejemplificar esta técnica a continuación se muestran algunos de los casos de pruebas que se diseñaron para el Sistema SAWMI.

1. **Identificación y nombre del Caso de Prueba:** Crear una factura.

2. **Caso de uso asociado:** Administrar Facturas.

3. **Propósito:** Llevar a cabo la prueba de la operación Crear Factura con el fin de verificar su correcto funcionamiento, detectando y corrigiendo algún error en la interfaz o en el código interno.

4. **Descripción general:** Se realizarán varias solicitudes de este tipo, con el fin de verificar el correcto funcionamiento de esta operación.

5. **Precondiciones:** Base de datos disponible. Además el usuario tiene que estar "autenticado" en el sistema para poder realizar la opción de Crear. Deben de existir al menos un empleado y un cliente en la base de datos, a su vez, una orden para relacionarla con la factura.

6. **Post-condición:** Ninguna.

7. **Entrada:** Todos los datos de la factura generados o que el usuario ingresó: numero*, numero de la orden*, cedula del cliente*, fecha*, fecha de la orden*, subtotal*, impuesto de ventas, total*, descuento y cedula del responsable*

   *Campo obligatorio

8. **Flujo único de eventos:**

#	Acciones de algún actor	#	Acciones del sistema	#	Punto de verificación
					Verificar precondiciones
1	E jefe de departamento o encargado de soporte presiona el botón *Crear*.	**2**	Carga el combo box de Ordenes		¿Se carga correctamente el comboBox de Órdenes?
3	Selecciona la orden	**4**	Realiza una consulta de la tabla de orden para obtener todos los datos que luego serán agregados a la factura. (Cliente, Fecha de la Orden, Subtotal, Descuento, Impuesto de Ventas y Total)		¿Realiza bien la consulta y obtiene los datos adecuados?
		5	Carga el datagrid con el detalle de la orden.		¿Carga bien el datagrid?
		6	Genera la fecha de la factura		¿Genera correctamente la fecha?
		7	Genera el numero de la factura		¿Genera correctamente el número de la factura?
		8	Obtiene la cedula del empleado loggeado		¿Obtiene correctamente la cedula del empleado?

9	Presiona el botón A*ceptar*.	**10**	El sistema despliega un mensaje para confirmar la creación		¿Muestra el mensaje solicitando la confirmación del usuario?
11	Confirma el mensaje de si está seguro de hacer la creación de factura.	**12**	Valida los datos.		¿Hace la validación correctamente? ¿Si alguno de los campos obligatorios no está lleno muestra un mensaje indicando esta situación?
		13	Guarda la información de la nueva factura en la base de datos.		¿Inserta la compra correctamente en la base de datos? ¿Muestra un mensaje indicando que la operación fue exitosa o de lo contrario que hubo un problema?
		14	Actualiza el inventario (Tabla Producto)		¿Actualiza correctamente el inventario?
14	Presiona OK en el mensaje de confirmación				
15	Presiona botón *Cancelar.*	**16**	Limpia los campos.		¿Limpia todos los campos?

9. **Salida:** El sistema muestra un mensaje indicando que la creación fue exitosa o de lo contrario que se presentó un problema y que lo intente de nuevo más tarde.

10. **Reporte:** Se creará un documento que contiene los resultados una vez realizada la prueba.

11. **Identificación y nombre del Caso de Prueba:** Modificar un cliente.
12. **Caso de uso asociado:** Administrar Clientes.
13. **Propósito:** Llevar a cabo la prueba de la operación Modificar Cliente con el fin de verificar su correcto funcionamiento, detectando y corrigiendo algún error en la interfaz o en el código interno.
14. **Descripción general:** Se realizarán varias modificaciones de este tipo, con el fin de verificar el correcto funcionamiento de esta operación.
15. **Precondiciones:** Para modificar un cliente, el jefe de departamento o encargado de soporte debe realizar una consulta (ver sección consultar).

    Base de datos disponible. Además el usuario tiene que estar "autenticado" en el sistema para poder realizar la opción de Modificar.
16. **Post-condición:** Ninguna.
17. **Entrada:** Todos los datos del cliente que el usuario ingresó: Cédula, Nombre, Tipo, Teléfono, Dirección, Correo Electrónico.

18. **Flujo único de eventos:**

#	**Acciones de algún actor**	#	**Acciones del sistema**	#	**Punto de verificación**
					Verificar precondiciones
1	E jefe de	**2**	Habilita los siguientes		¿Están habilitados cada uno de los

	departamento o encargado de soporte presiona el botón *Modificar*.		campos que se pueden modificar: Cédula, Nombre, Tipo (comboBox), dirección, teléfono, Correo Electrónico.		campos? ¿Se despliega la lista de tipos de cliente?
3	Modifica los datos necesarios.				
4	Presiona el botón A*ceptar*.	**5**	El sistema despliega un mensaje para confirmar la modificación		¿Muestra el mensaje solicitando la confirmación del usuario?
6	Confirma el mensaje de si está seguro de modificar el cliente.	**7**	Valida los datos.		¿Hace la validación correctamente? ¿Si alguno de los campos obligatorios no está lleno muestra un mensaje indicando esta situación?
		8	Guarda la información del cliente en la base de datos.		¿Modifica al cliente correctamente en la base de datos? ¿Muestra un mensaje indicando que la operación fue exitosa o de lo contrario que hubo un problema?
9	Presiona OK en el mensaje de confirmación	**10**	Despliega el datagrid de *Clientes Actuales en el Sistema* actualizado.		¿Actualiza correctamente el datagrid?
11	Presiona botón *Cancelar*.	**12**	Limpia los campos.		¿Limpia todos los campos?

19. **Salida:** El sistema muestra un mensaje indicando que la modificación fue exitosa o de lo contrario que se presentó un problema y que lo intente de nuevo más tarde.
20. **Reporte:** Se creará un documento que contiene los resultados una vez realizada la prueba.