Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: "Desarrollo del subsistema Transferencias de Quarxo"

Autor: Wilfredo José Guerra Quesada.

Tutor(es): Ing. José Antonio Sanchez Imbert.
Ing. Adrian Veranes Fonseca.

La Habana, Junio del 2012

"Año del 54 Aniversario de Triunfo de la Revolución"

Declaración de Autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

_____ _____

Wilfredo José Guerra Quesada **Ing. José Antonio Sanchez Imbert**

_____

Ing. Adrian Veranes Fonseca

Agradecimientos

A Emilio por ser mi guía y amigo que tanto me ha ayudado, que sin él nada de esto podía haberse cumplido.

A mi oponente Yesenia que ha sido una persona muy especial y que he podido contar con ella en todo momento.

A mis hermanos aquí en la UCI que tanto hemos compartido: Adrian, Danis, Bode, Vladimir, Nelson, Lian, Leonardo, Efrain, Héctor, que de una forma u otra han estado presente cuando los he necesitado.

A todos mis amigos del grupo, en especial a Dailiana, Lucrecia, por ser como si fueran mis hermanas.

A todas las personas que me han ayudado en el proyecto y que merecen ser reconocidos como: Manuel, Jorge, Jorgito, Dariel, Annier, Evelio, Matías.

A mis tutores que han estado presentes en todos los cortes, y que me han ayudado de una manera u otra.

Al tribunal por señalarme en todos los cortes lo que debía mejorar.

A Kamelia, una persona que ha sido capaz de estar conmigo en los momentos buenos y malos aquí en la Universidad.

Dedicatoria

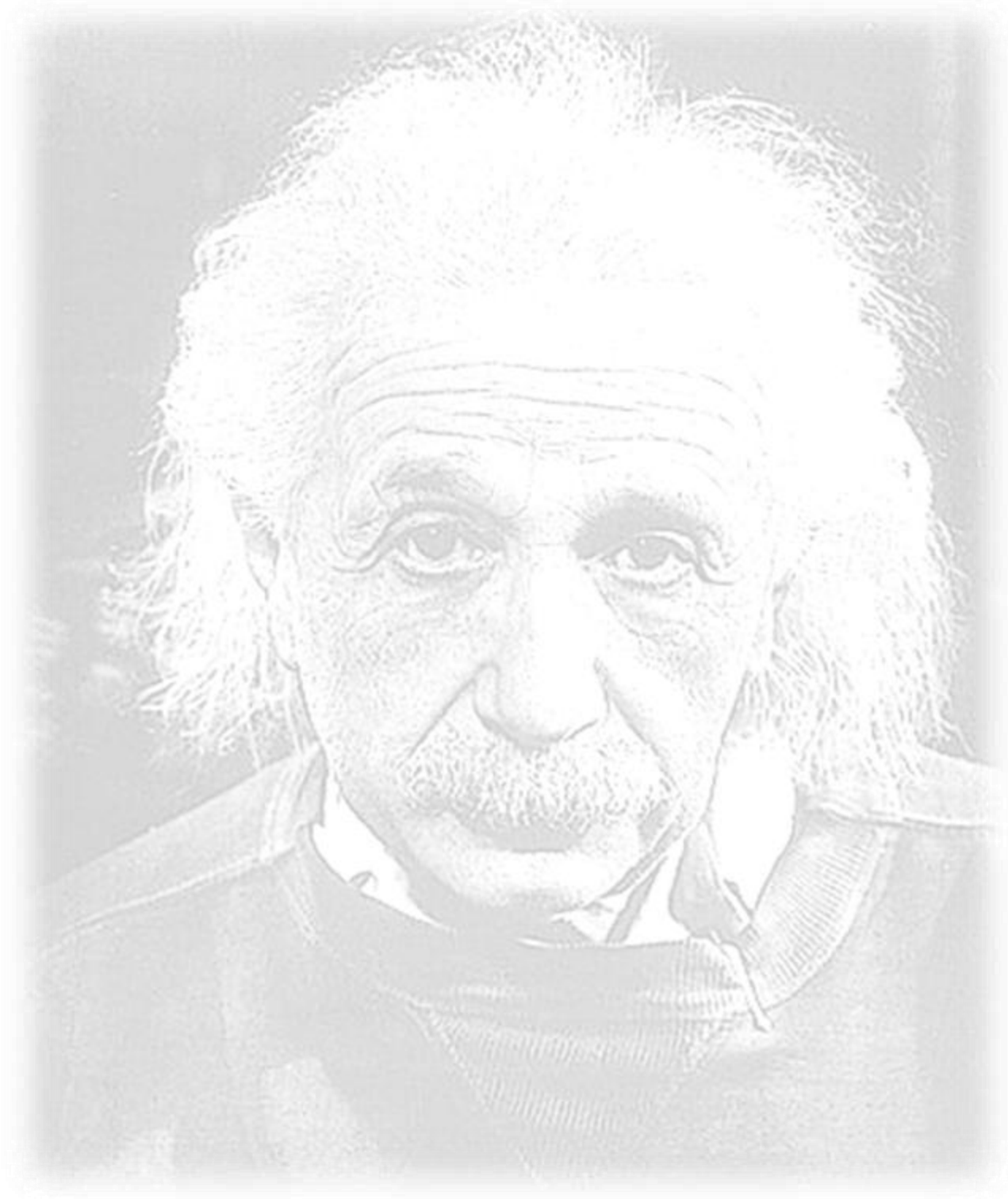
A mi mamá Regla que tanto me ha educado para ser un hombre de bien, a mi papá por ser mi amigo incondicional que siempre ha estado ahí para ayudarme.

A mis segundas madres Teresa, Anita y María por todo el cariño que he recibido desde que era un niño.

A mis primos Humberto y Mayito, A todas mis tías y tíos por el cariño que me han brindado siempre.

A mi familia en general y a la Revolución por darme la oportunidad de forjarme como ingeniero.

Frase Célebre



…No pretendamos que las cosas cambien si siempre hacemos lo mismo….La creatividad nace de la angustia como el día nace de la noche oscura….Sin crisis no hay desafíos, sin desafíos la vida es una rutina…

Albert Einstein

Resumen

La Universidad de las Ciencias Informáticas se hizo responsable de implementar un producto para satisfacer las necesidades que existen en el Banco Nacional de Cuba, debido a su creciente cambio en su actividad bancaria y financiera. La primera versión de este producto es Quarxo, el cual no cuenta con la posibilidad de gestionar una serie de Transferencias que han sido incluidas en el negocio de la Gerencia de Cuentas Corrientes y otros departamentos del Banco Nacional. Este sistema no permite realizar ninguna operación directamente sobre las Transferencias, como es el caso de cancelar, consultar ni actualizar, además, las nuevas operaciones que han sido incluidas en la institución bancaria no permite el envío de mensaje hacia los bancos con los que interactúan.

El objetivo del presente trabajo de diploma es el desarrollo de un subsistema capaz de resolver los problemas que en la actualidad se presentan en la entidad financiera antes mencionada, referente a la gestión de las Transferencias. Para el desarrollo de la solución se realizó un estudio sobre la metodología seleccionada, los diferentes patrones de diseño aplicados, el ambiente de desarrollo; así como una valoración de algunos de los sistemas financieros que gestionan las Transferencias. Se exponen los diferentes artefactos generados para la solución de la problemática planteada, los diagramas generados por cada capítulo, además de la validación del diseño y las pruebas que se le aplicaron a los requisitos funcionales. Como resultado se obtiene el desarrollo del subsistema Transferencia de Quarxo el cual permite gestionar las Transferencias que en el BNC se realizan, permitiendo el envío de mensaje correspondiente a cada transacción que lo requiera.

Palabras claves:

Análisis, Banco Nacional de Cuba, Diseño, Implementación, Transferencias.

Índice de Contenido

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

El actual desarrollo de las Tecnologías de la Información y las Comunicaciones en Cuba brinda la posibilidad de crear sistemas financieros capaces de tener una mejor administración, registro y control de los flujos monetarios. La incorporación de los avances científicos y técnicos propicia que las soluciones informáticas contribuyan al incremento de la capacidad de organización en los procesos de negocios. La competencia entre productos informáticos eleva la calidad y el grado de adaptabilidad en las entidades donde son utilizados. Estos productos además, permiten la comunicación con los sistemas financieros con los que interactúan, ya que estos dependen en gran medida del intercambio de información. Por lo general se usa el estándar SWIFT[1] que responde a las necesidades de los bancos con respecto a la liquidación y compensación bilateral de los pagos interbancarios, sobre todo los pagos transfronterizos de un valor de medio a alto.

Como parte de la informatización de la sociedad que se ha estado llevando en Cuba, los sectores económicos se han sumado al desarrollo y renovación de sus sistemas contables y financieros. El Banco Nacional de Cuba (BNC) no se encuentra ajeno a este importante proceso y el creciente cambio en su actividad bancaria se le ha creado la necesidad de automatizar sus procesos de negocio. La Universidad de las Ciencias Informática en conjunto con el Banco Nacional de Cuba se sumó a la tarea de desarrollar una solución informática que satisfaga las necesidades de automatización de esa entidad y pueda ejecutar con una mejor eficacia y seguridad sus obligaciones bancarias.

La primera versión de una solución que automatiza los procesos que se realizan a diario en el Banco Nacional de Cuba es Quarxo. Este software ha sido desarrollado por el equipo de trabajo del proyecto SAGEB[2].Dentro de sus múltiples funcionalidades se cuenta con un módulo que gestiona las Transferencias bancarias. Dentro de dicho módulo se registran los distintos tipos de transferencias que son utilizadas en algunas de las operaciones más importantes que se realizan en el BNC.

Las funcionalidades para gestionar las Trasferencias bancarias que proporciona QUARXO, aún no cuenta con la posibilidad de contabilizar otros tipos de transferencias que han sido incluidas en el negocio de la Gerencia de Cuentas Corrientes en el BNC. Existe la necesidad de automatizar las Transferencias BFI[3], Transferencias recibidas de clientes particulares y

[1]SWIFT: (en inglés: Society for Worldwide Interbank Financial Telecommunication) es una organización que posee una red internacional de comunicaciones financieras entre bancos y otras entidades financieras.
[2]Sistema Automatizado de Gestión Bancaria.
[3]Banco Financiero Internacional.

Transferencias de instituciones, así como las Transferencias enviadas de Tesorería. Esta parte del sistema Quarxo no permite: rechazar, actualizar, consultar ninguna operación sobre las Transferencias que se hacen en la entidad financiera. El sistema no cuenta con la posibilidad de enviar los mensajes correspondientes de las nuevas Transferencias que han sido incorporadas al negocio de la institución financiera.

A partir de la situación problemática anteriormente detallada, se define el siguiente **problema a resolver**: ¿Cómo mejorar las Transferencias en el Banco Nacional de Cuba?

Se tiene como **objetivo general**: Desarrollar un subsistema que permita gestionar las Transferencias en el BNC. Teniendo como **objeto de estudio:** El proceso de las, Trasferencias en entidades financieras bancarias centrando como **campo de acción:** La gestión de Trasferencias en el BNC.

Se plantea la siguiente **idea a defender:** Con la implementación del subsistema Transferencias en su versión 2.0 se prevee mejorar el proceso de las transferencias para el BNC. Partiendo de esto se identifican los siguientes **objetivos específicos**:

- Realizar una fundamentación teórica de la solución.
- Obtener los requerimientos funcionales para el desarrollo de la solución.
- Modelar los artefactos del negocio.
- Diseñar e implementar los módulos del subsistema de Transferencias de Quarxo.
- Validar la solución implementada para el subsistema Transferencias de Quarxo en su segunda fase.

Teniendo como **posible resultado** el desarrollo de un subsistema que permita la gestión de las transferencias en el Banco Nacional de Cuba. El presente trabajo de diploma fue estructurado en cuatro capítulos:

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace un estudio del estado del arte de la gestión de las Transferencias bancarias en el mundo. Se analizan algunos sistemas bancarios mundiales. Son especificadas las características de la metodología utilizada, los lenguajes de modelado, además de las herramientas y Frameworks que conforman el ambiente de desarrollo sobre el cual se realizó el sistema.

CAPÍTULO 2. NEGOCIO, REQUERIMIENTOS Y ANÁLISIS DEL SISTEMA.

En el capítulo se realiza una descripción acerca de las características del sistema a partir del modelo del negocio; se identifican involucrados y artefactos, se describen las reglas del

negocio y se definen cuáles son los procesos a automatizar. Se identifican las funcionalidades que tendrá el subsistema Transferencia de Quarxo, así como una definición de cada requerimiento. Se muestran los artefactos generados en el análisis de la propuesta solución, se analizan los casos de usos críticos, definiendo las clases del análisis para cada uno, así como la organización de los módulos mediante un diagrama de paquetes.

CAPÍTULO 3. DISEÑO E IMPLEMENTACIÓN.

En este capítulo se modelan los diagramas correspondientes al diseño de la solución, además se muestra la implementación de la solución propuesta del subsistema de Transferencia de Quarxo donde se abordan aspectos esenciales a tener en cuenta a la hora de implementar, como son los estándares de codificación, la iteración de los componentes en sus diagramas correspondientes, la descripción de sus clases y funcionalidades,

CAPÍTULO 4. Validación de la solución

Se realiza la validación del diseño, además de la validación de la solución mediante las pruebas de caja blanca que fueron realizadas al software y las pruebas de caja negra que se realizaron al sistema para comprobar su correcto funcionamiento.

Se muestran al finalizar las conclusiones del trabajo, la bibliografía, y los anexos.

Capítulo1: Fundamentación Teórica.

1.1. Introducción.

En este capítulo se presenta un estudio del estado del arte de la gestión de las Transferencias bancarias en el mundo, los principales conceptos y definiciones referentes a este tema. Se analizaran algunos sistemas bancarios que gestionan el manejo de las Transferencias bancarias.

Se especifican algunas características de la metodología a utilizar, lenguajes, herramientas y Frameworks que conforman el ambiente de desarrollo sobre el cual se va a implementar el subsistema de Transferencias de Quarxo.

Conceptos y definiciones.

1.2. Banco Nacional de Cuba (BNC).

El Banco Nacional de Cuba (BNC) fue creado mediante la Ley No. 13, del 23 de diciembre de 1948, como banco central del estado, con autonomía orgánica, personalidad jurídica independiente y patrimonio propio. El 23 de febrero de 1998 se promulga el Decreto Ley No. 181, que recoge la estructura, funciones y actividades asignadas como organización bancaria internacional.

Entre sus funciones se encuentra:

- Obtener y otorgar créditos en moneda nacional y en moneda libremente convertible.
- Emitir garantías bancarias de todo tipo previa evaluación de la gestión económico-financiera de la entidad solicitante.
- Librar, aceptar, descontar, avalar y negociar, en cualquier forma, letras de cambio, pagarés, cheques y otros documentos mercantiles negociables denominados en moneda nacional y divisas.
- Fijar las tasas de interés que deberán aplicar a las operaciones que efectúe el Banco, dentro de los límites que establezca el Banco Central de Cuba.
- Constituir entidades de seguro de crédito oficial a la exportación con arreglo a la legislación vigente en materia de seguros.

Mantener el registro, control, servicio y atención a la deuda externa que el Estado cubano y el **Banco Nacional de Cuba** tiene contraída con acreedores extranjeros hasta la fecha de entrada en vigor del Decreto Ley No. 172, de 1997, del Banco Central de Cuba. (1)

1.3. Transferencia Bancaria.

Una transferencia bancaria es un sistema mediante el cual se transfieren fondos entre distintas cuentas bancarias sin necesidad de transportar físicamente el dinero. Las transferencias bancarias pueden realizarse a través de cajeros automáticos, home banking, o las sucursales de la entidad bancaria. (2)

Cuando la transferencia tiene lugar entre cuentas de la misma entidad de crédito, la operación se suele denominar "**traspaso interno**". (3)

Existen varios criterios para la clasificación de las transferencias bancarias:

- Área geográfica: Las nacionales son aquellas que tanto el que envía el dinero como el que lo recibe están en el mismo país. Las internacionales o exteriores son aquellas que el receptor se encuentra en otro país. (3)
- Modo de ordenarlas: Se pueden realizar estas transferencias personalmente en la sucursal de la entidad, a través de cajeros, por teléfono y por internet. (3)

Las transferencias suelen poseer comisiones y los traspasos suelen ser gratuitos, debido a que se efectúan dentro de la misma entidad. El valor de las comisiones por transferencia depende de la entidad y su comisión la amortiza el titular de la cuenta que realiza la transferencia. Las transferencias y traspasos se efectúan cada vez más por medio de la banca electrónica, ya que facilita y agiliza el proceso. (4)

Para realizar una transferencia al extranjero se necesita el número IBAN[4]que es el número internacional de la cuenta, formado por una secuencia alfanumérica variable entre los distintos países y con un máximo de 34 caracteres. El IBAN permite reconocer al beneficiario de la transacción. El IBAN está determinado por la norma ISO 13616:1997, conforme a las siguientes reglas:

- Los primeros dos caracteres hacen referencia al código del país.
- Los siguientes dos caracteres son un dígito de control.
- Hasta treinta caracteres, según el país, que corresponden al número de cuenta.

Asimismo, para llevar a cabo una transferencia internacional se necesita el código SWIFT/BIC[5], que permite identificar a la entidad financiera y sucursal a donde se dirige la transacción. El código SWIFT/BIC se basa en la norma ISO 9362 y puede contener entre 8 a 11 caracteres:

- Cuatro caracteres que se refieren al código del banco.
- Dos caracteres que se refieren al código ISO del país.
- Dos caracteres que se refieren a la localidad.
- Tres caracteres que se refieren a la sucursal.

Es de útil ayuda considerar las siguientes premisas:

[4] International Bank Account Number, es el número que posee la cuenta internacional del banco.
[5] Bank Identification Code, código que identifica el banco.

- Si el código tiene ocho caracteres, no se incluyen los referentes a la sucursal y la transferencia llegará a la principal oficina del banco en la ciudad destinataria.
- Si el código tiene once caracteres, incluye todos los campos mencionados y la transferencia llegará exactamente a la oficina del destinatario.

Se indagó respecto a quién pagaría o correría con los gastos de las comisiones al realizar una transferencia y se obtuvieron tres posibilidades:

- **Paga el ordenante:** es lo que en el argot financiero internacional se conoce como OUR (nuestro en inglés). El que envía la transferencia corre con todos los gastos y comisiones bancarias. (3)
- **Paga el beneficiario:** es la opción BEN (apócope de beneficiary, beneficiario en inglés). Todos los gastos y comisiones son a cargo del que recibe la transferencia, al que, en consecuencia, le llegará menos dinero del que le fue enviado. (3)
- **Pagan los dos:** se denomina modalidad SHA (apócope de share, es decir, compartir). El ordenante paga a su entidad y el beneficiario a la suya. (3)

1.4. Sistemas Financieros.

1.4.1. SAP para Banca.

SAP para Banca ofrece un completo conjunto de capacidades para la planeación estratégica, contabilidad, costos y el control de la empresa. Combinando estas capacidades con componentes específicos de la industria, como lo son administración de la rentabilidad, riesgo, la relación con los clientes y un sistema totalmente integrado para el manejo de cuentas de clientes. SAP para Banca habilita a los Bancos a mantener una única visión de sus clientes. (5)

SAP para Banca soporta sus procesos críticos con sistemas bancarios específicos de cada industria, entre los que se cuentan:

- SAP Accounting for Financial Instruments.
- SAP Basel II.
- SAP Deposits Management
- SAP Financial Database.
- SAP Leasing.
- SAP Loans Management.

Administración de Transacción es una de las funcionalidades de SAP que ofrece un conjunto de funciones para el manejo de las transacciones financieras requeridas, su objetivo es:

- Manejar transacciones financieras y posiciones, así como su correspondiente transferencia a la Contabilidad Financiera. (6)
- Proveer facilidades para la generación de reportes y definición de estructuras de evaluación para el análisis de las transacciones financieras, (6)

En la siguiente figura se describen los campos que aparecen en la ficha Transferencia bancaria de la ventana Medios de pago. (6)

Figura 1: Ficha de Trasferencia Bancaria.

Medios de pago: Ficha Transferencia bancaria

Campo	Descripción/Actividad
Cuenta de mayor	Introduzca la cuenta de mayor del plan de cuentas desde o hasta donde se deba contabilizar la transferencia bancaria.
Fecha de transferencia	Indique la fecha de la transferencia bancaria.
Referencia	Introduzca una referencia para la transferencia bancaria, por ejemplo, la finalidad de la transferencia.
Total	Indique el importe de la transferencia bancaria.

1.4.2. SABIC

El SABIC (Sistema automatizado para la Banca Internacional de Comercio) es un sistema diseñado y desarrollado por la Dirección de Sistemas Automatizados del Banco Central de Cuba para satisfacer las necesidades de procesamiento de datos de Bancos e instituciones no bancarias, utilizando los medios técnicos de computación disponibles en el mercado.

El problema fundamental de SABIC con respecto a la gestión de las transferencias es que no posee integración con el sistema de mensajería SWIFT para la ejecución de transferencias bancarias que necesiten de la emisión de mensaje.

1.4.3. Abanfin

Este sistema permite realizar las transacciones financieras con diferentes modalidades, entre las que se encuentran: Ordinarias, Magnéticas, Cuenta a Cuenta y Orden de Movimiento de Fondos (OMF). (7)

La emisión de transferencias se realiza a través de medios electrónicos utilizando dos sistemas:

- **Mediante la conexión a sistema de banca electrónica**: Se realiza habitualmente por internet, donde el usuario puede ordenar la transferencia en cualquier momento del día, utilizando como firma una clave numérica o alfanumérica personal. (7)
- **Mediante la transmisión de un fichero normalizado a la entidad financiera**: Cuando se realizan un número elevado de transferencias es muy habitual utilizar los

denominados cuadernos[6] de la Asociación Española de la Banca. Dichos cuadernos no son más que una serie de normas para la codificación de un fichero electrónico donde se detallan todos los datos del emisor de las transferencias así como todos y cada uno de los datos necesarios de los beneficiarios. Esta normalización permite que los programas de gestión más modernos permitan confeccionar de forma automática dicho fichero para posteriormente transmitirlo y autorizarlo, habitualmente vía internet, a la entidad desde la que se desea realizar la transferencia. (7)

1.4.4. Bantotal.

Es una solución que comprende el procesamiento integral de todas las operativas de la Entidad Financiera (Core Bancario –Core Banking Solutions). Por tanto, además de brindar el soporte funcional a cada uno de los productos y servicios que la Entidad Financiera ofrece, incluye el Sistema Contable (cumplimentando las normas de registro y exposición que determinen los organismos reguladores en cada país) como la emisión de los reportes requeridos por estos. (8)

A su vez, ofrece soluciones especializadas para las áreas de Microfinanzas, Comercio Exterior, Tesorería (incluyendo operaciones relacionadas con los Mercados de Dinero, Divisas y Valores), Canales de Distribución (Online Banking), Información Gerencial (basada en tecnología OLAP), e Información a Organismos Reguladores y de Supervisión en cada país donde se instale. (8)

- **Sistema de Transferencias**

Permite el procesamiento de los Servicios indicados, apoyándose en los Sistemas de Precios a efectos de cálculo de las comisiones de los servicios y en las funciones provistas en forma genérica por la aplicación (generación de impresos, mensajes SWIFT).

La función asociada al Sistema de Transferencias es el Ingreso de Transferencia al Exterior (pago por caja/débito en cuenta).Esta función es la que permite la gestión de las transferencias.

1.4.5. Valoración sobre los Sistemas Bancarios.

Los sistemas antes mencionados tienen licencias privativas, lo que dificulta tanto la adquisición como el uso y mantenimiento de los mismos. Ofrecen servicios de transferencias bancarias que es elemento clave para la actividad económica y financiera a nivel mundial. Los servicios asociados a las transferencias bancarias no se corresponden con el negocio que se lleva a cabo en el BNC, ya que no cumplen con la mayoría de las operaciones

[6]Serie de normas usadas para la codificación de un fichero electrónico en el cual se detallan los datos tanto del emisor como del beneficiario de las transferencias bancarias.

bancarias llevadas a cabo en esta institución, pues no permiten definir los tipos de Transferencias que se llevan a cabo en la entidad donde son implementados estos sistemas financieros. En el caso de Abanfin a pesar que es privativo no permite el envío de mensaje SWIFT después de cada Transacción que se realice.

Con la necesidad de automatizar el Banco Nacional de Cuba con las nuevas tecnologías se desarrolla el producto Quarxo; y en la búsqueda de un sistema flexible y robusto se le añade el subsistema Transferencia, que permite gestionar todo tipo de transferencias bancarias llevadas a cabo en el BNC brindando la posibilidad además de añadir nuevos tipos de transferencias que surjan de la toma de decisiones en aras de modernizar sus estrategias de negocio.

1.5. Ingeniería de Requerimientos.

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando las necesidades, confirmando su viabilidad, negociando una solución razonable, validando la especificación y gestionando los requisitos para que se transformen en un sistema operacional. (9)

La parte más difícil de construir un sistema es precisamente definir las funcionalidades que debe de ofrecer. Muchos han sido los productos de software que se han creado y las experiencias que éstos han dejado demuestran que los requerimientos o requisitos de software constituyen una pieza fundamental, que permiten obtener en gran medida las verdaderas funcionalidades que tendrá un sistema automatizado. Dichas funcionalidades marcan el punto de partida para las actividades en la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo. (10)

1.5.1. Requerimientos de Software.

Los requerimientos son una descripción de las necesidades o deseos de un producto. La meta primaria de la fase de requerimientos es identificar y documentar lo que en realidad se necesita en una forma que claramente se lo comunique al cliente y a los miembros del equipo de desarrollo. (11)

1.5.2. Tipos de Requerimientos de Software.

Los requisitos funcionales son aquellos que definen los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares. (12)

Los requisitos no funcionales son las restricciones a los servicios o funciones del sistema, tales como restricciones de tiempo, sobre el proceso de desarrollo, estándares. (12)

1.5.3. Técnicas para la Captura de Requerimientos de Software.

- **Entrevistas.**

Es una técnica muy utilizada y aceptada para la obtención de la información de los clientes. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. La estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados. (13)

- **Modelado de Negocio.**

En el modelo de negocio se describen los procesos del negocio y es fundamental para la comprensión general de los mismos. (13)

Existen otras técnicas para la elicitación de requerimientos como son:

- Brainstorming/Tormenta de ideas.
- Cuestionarios.
- Casos de Uso.
- Comparación de terminología.

Las técnicas que se emplearán en la elicitación de los requerimientos funcionales para el desarrollo del subsistema de **Transferencia de Quarxo** serán la entrevista y el modelado del negocio.

1.5.4. Técnicas para la Validación de Requerimientos de Software.

- **Reviews/ Revisiones.**

Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida (13). Los conflictos, contradicciones, omisiones y errores, deben señalarse durante la revisión y registrarse formalmente.

- **Prototipos.**

Algunas propuestas se basan en obtener de la definición de requisitos, prototipos que sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario (13). Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

- **Otras técnicas para la validación de requerimientos son:**
  - Auditorías.
  - Matrices de trazabilidad.

La técnica que se empleará para la validación de los requerimientos es el **prototipo.**

1.6. Tecnologías y herramientas utilizadas en el modelado.

- **Metodología RUP.**

Una metodología no es más que el estudio de los métodos más apropiados que se emplean para desarrollar software de manera eficiente; o como precisan otros autores (14), define Quién debe hacer Qué, Cuándo, y Cómo debe hacerlo.

RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos. Utiliza el Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software, siendo UML una parte esencial del Proceso Unificado. (15)

- **Lenguaje de Modelado UML.**

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Puede ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero es útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientado a objetos. (16)

Es un lenguaje gráfico para visualizar, especificar y documentar un sistema de software. Ofrece un estándar para describir un panorama del sistema (modelo), incluyendo aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos, componentes de software reutilizables y aspectos conceptuales como los procesos de negocios y funciones del sistema. (15)

- **Herramienta de modelado Visual Paradigm para UML.**

Visual Paradigm es una herramienta que ofrece un entorno de creación de diagramas usando las notaciones UML 2.0 y BPMN[7], con un diseño centrado en casos de uso y enfocado además al negocio, lo cual genera un software de alta calidad. Esta herramienta fue creada para el ciclo completo de desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes. Permite escribir toda la especificación de un caso de uso sin necesidad de utilizar una herramienta externa como editor de texto, utilizando plantillas que se encuentran definidas, o que pueden ser creadas por los usuarios. (17)

Visual Paradigm una herramienta CASE[8] que presenta ventajas para el modelado de cualquiera de los diagramas para UML, además de que soporta las últimas versiones del Lenguaje Unificado de Modelado y la Notación de Modelado de Procesos de Negocios, por estas razones se decidió por parte de la dirección nuestro proyecto utilizarlo para el desarrollo de los diagramas correspondientes.

- **BPMN (Business Process Modeling Notation).**

Es una notación gráfica que describe la lógica de los pasos de un proceso de Negocio. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. (18)

BPD es un Diagrama diseñado para representar gráficamente la secuencia de todas las actividades que ocurren durante un proceso, incluye además toda la información que se considera necesaria para el análisis. (18)

BPD es un Diagrama diseñado para ser usado por los analistas de procesos, quienes diseñan, controlan y gestionan los procesos. Dentro de un Diagrama de Procesos de negocios BPD se utilizan un conjunto de elementos gráficos, que se encuentran agrupados en categorías. (18)

¿Por qué es importante Modelar con BPMN?

- BPMN es un estándar internacional de modelado de procesos aceptado por la comunidad.
- BPMN es independiente de cualquier metodología de modelado de procesos. BPMN crea un puente estandarizado para disminuir la brecha entre los procesos de negocio y la implementación de estos.

[7]Notación de Modelado de Procesos de Negocios por sus siglas en español
[8]Computer Aided Software Engineering / Ingeniería de Software Asistida por Ordenador.

1.7. Patrones de Diseño.

- **Patrones de asignación de responsabilidades. (GRASP[9])**

Los patrones GRASP, describen los principios fundamentales de la asignación de responsabilidades a objetos, expresadas en forma de patrones. Craig Larman en su libro "UML y Patrones" define: "Los patrones GRASP constituyen un apoyo para la enseñanza que ayuda a uno a entender el diseño de objetos esencial, y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable." (19)

- **Creador.**

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. (20)

- **Experto.**

Este patrón permite asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. La responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantendrá encapsulada. (20)

- **Bajo acoplamiento.**

Este patrón consiste en tener las clases lo más independientes entre sí posible y con la menor cantidad de relaciones; lo que posibilita en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, permitiendo la reutilización y disminuyendo la dependencia entre las clases, asignándoles una responsabilidad para mantener un bajo acoplamiento. (20)

- **Alta Cohesión.**

La principal característica de este patrón es asignar responsabilidades de modo que la cohesión siga siendo alta. La información que almacena una clase debe de ser coherente y debe estar en la medida de lo posible relacionada con la clase. (20)

- **Controlador.**

Es el intermediario entre una interfaz y la implementación de la lógica de las operaciones del sistema. Se responsabiliza por controlar el flujo de eventos del sistema. (11).

[9] Patrones generales de software para asignar responsabilidades por sus siglas en español.

1.7.1. Patrones Estructurales.

- **Fachada.**

Es el único punto de entrada para los servicios de un subsistema; la implementación y otros componentes del subsistema son privados y no pueden verlos los componentes externos. Proporciona protección frente a los cambios en las implementaciones de un subsistema (21)

- **Patrón de Acceso a Datos (DAO).**

El patrón de acceso a datos (DAO por sus siglas en inglés **D**ata **A**ccess **O**bject) resuelve el problema de contar con diversas fuentes de datos, además de que encapsula y oculta la forma de acceder a la base de datos, eliminando así complejidad del uso de JDBC a la capa de presentación o de negocio. Se trata de que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o de cuál es la fuente de almacenamiento. (19)

1.7.2. Patrones de Presentación.

- **Vistas Compuestas.**

Un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include. (22)

- **Modelo Vista Controlador (MVC).**

El **modelo** administra el comportamiento y la información del modelo del negocio, de la misma forma que responde sobre acciones que se quieran efectuar sobre éste, la **vista**, es la encargada de administrar la presentación de la información, mientras que el **controlador** es el componente intermedio entre el modelo y la vista para lograr un desacoplamiento total entre estos dos. El controlador es como el “mensajero” entre la parte visual y la información a ser manipulada. (23)

1.8. Ambiente de Desarrollo.

- **Plataforma JEE.**

La plataforma J2EE es una especificación para el desarrollo de aplicaciones empresariales basadas en web que corren en ambientes multicapa. J2EE proporciona un modelo de programación que consiste en un conjunto de Application Programming Interface (APIs) y dirige la manera de construir aplicaciones. Por otra parte, J2EE especifica cómo debe ser la infraestructura de la aplicación sobre la cual puedan correr las aplicaciones. Esta infraestructura de la aplicación es proporcionada por los contenedores de las implementaciones de J2EE. (24)

- **Lenguaje Java.**

Java es un lenguaje sencillo orientado a objetos; es independiente de plataforma, por lo que un programa hecho en Java se ejecutará igual en un PC con Windows que en una estación de trabajo basada en Unix. Su capacidad multihilo y su robusta integración que tiene con el protocolo TCP/IP, lo hace un lenguaje ideal para Internet. (25)

El JDK (Java Development Kit) es un producto que permite crear aplicaciones en Java. Este paquete incluye un conjunto de herramientas para compilar, depurar, generar documentación e interpretar código escrito en Java. (25)

Para el desarrollo del producto Quarxo se utiliza la versión: Java (TM) 6 Update 20 versión: 6.0.200.

- **Contenedor Web (Apache Tomcat).**

Tomcat es un contenedor web basado en el lenguaje Java que actúa como motor de Servlets y JSPs. Se ha convertido en la implementación de referencia para las especificaciones de Servlets y JSPs. Fue seleccionado como la implementación de referencia de contenedores de componentes web Sun (JSPs/Servlets) y está desarrollado en un entorno abierto Open Source. (26)

- **Gestor de Bases de Datos (SQL).**

SQL Server es una solución de datos global, e integrada que habilita a los usuarios en toda su organización mediante una plataforma más segura, confiable y productiva para datos empresariales y aplicaciones de Inteligencia Empresarial (BI). SQL Server 2005 provee herramientas sólidas y conocidas a los profesionales , así como también a trabajadores de la información, reduciendo la complejidad de la creación, despliegue, administración y uso de aplicaciones analíticas y de datos empresariales en plataformas que van desde los dispositivos móviles hasta los sistemas de datos empresariales. (27)

- **ER/Estudio**

ER/Studio es una solución intuitiva y visual de modelado de datos para el diseño y mantenimiento de bases de datos transaccionales, de ayuda a la toma de decisiones y para la Web. Permite a los profesionales de bases de datos controlar, documentar y desplegar rápidamente cambios en el diseño en las principales plataformas de bases de datos.

En la realización del diseño de la base de datos se utiliza la versión Embarcadero ERStudio 8.0.0.5865.

- **Eclipse 3.3 IDE[10].**

Eclipse es un IDE para Java muy potente. Fue creado por la IBM bajo la filosofía de software libre. Se está convirtiendo en el estándar de puntera de los entornos de desarrollo para Java. Es Eclipse un marco de trabajo que está compuestos por componentes que se pueden o no incluir en dependencia de las necesidades del desarrollador, a estos complementos se les llama (plugins). De hecho, existen complementos que permiten usar Eclipse para programar en otros lenguajes aparte del Java como son PHP, Perl, Python, C/C++.

1.9. Tecnologías y Frameworks.

- **Frameworks.**

Los Frameworks son librerías de clases que dan solución a un problema determinado o brindan utilidades sobre áreas definidas. Resulta normal que un framework no se ajuste totalmente a las necesidades de un proyecto en específico, por lo que es razonable que se necesite variar el funcionamiento que propone por lo que posee una estructura de Software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. (28)

- **Spring Framework.**

Spring Web Flow es una extensión de Spring MVC que permite la aplicación de los "flujos" de una aplicación web. Un flujo encapsula una secuencia de pasos que guían al usuario a través de la ejecución de una tarea de negocios. Abarca múltiples peticiones HTTP, tiene ofertas del Estado, con los datos transaccionales, es reutilizable y puede ser dinámico y de larga duración en la naturaleza. (29)

Se utiliza la versión 2.0.6 en el desarrollo de Quarxo.

- **Spring WebFlow.**

Spring Web Flow proporciona un lenguaje de definición de flujo declarativo para los flujos de creación en un nivel más alto de abstracción. Se le permite ser integrado en una amplia gama de aplicaciones sin ningún tipo de cambios (en el modelo de programación de flujo) incluyendo Spring MVC, JSF. (29)

Se utiliza para el desarrollo del producto Quarxo la versión 2.0.6.

- **Hibernate.**

Es un framework para persistir objetos en una base de datos relacional, disponible para Java con el nombre de Hibernate y para tecnología .NET con el nombre de NHibernate.

[10] Integrated Development Environment (Entorno de Desarrollo Integrado por sus siglas en español.)

Tiene grandes ventajas en cuanto a productividad, mantenibilidad, rendimiento e independencia del proveedor. Entre sus funciones está permitir transacciones, asociaciones, polimorfismo, herencia, carga mediante referencia, la cual consiste en cargar sólo la referencia de donde se encuentran los datos, es llamada (carga lazy), persistencia transitiva, estrategias de fetching además de permitir distintas formas de realizar las consultas como son Hibernate Query Language (HQL), Java Persistence Query Language (JPQL), consultas por criterios y SQL nativo. (30)

Se utiliza en el desarrollo de Quarxo la versión 3.5.

- **DojoToolkit.**

Dojo es un framework de JavaScript que nos ofrece muchos instrumentos, como son controles del interfaz de usuario, efectos, utilidades comunes; es una API para gestionar el acceso asíncrono al servidor. (31)

Para el desarrollo del producto Quarxo se utiliza la versión 1.3.

1.10. Conclusiones parciales.

En este capítulo se presenta un estudio del estado del arte de la gestión de las Transferencias bancarias en el mundo, los principales conceptos y definiciones referentes a este tema. Se analizaron algunos sistemas bancarios que son para el manejo de las Transferencias bancarias.

Se caracterizaron metodología, lenguajes, herramientas y Frameworks que conforman el ambiente de desarrollo sobre el cual se implementó el sistema Quarxo y se desarrolla la segunda versión para la gestión de las Transferencias bancarias en el BNC.

Capítulo 2: Características y Análisis del Sistema.

2.1. Introducción

En el capítulo se realiza una descripción de las características del sistema a partir del modelo del negocio; se identifican involucrados y artefactos, se describen las reglas del negocio y se definen cuáles son los procesos a automatizar. Se identifican las funcionalidades que tendrá el subsistema Transferencia de Quarxo, así como una definición de cada requerimiento. Se muestran los artefactos generados en el análisis de la propuesta solución, se analizan los casos de usos críticos, definiendo las clases del análisis para cada uno, así como la organización de los módulos mediante un diagrama de paquetes.

2.2. Modelado de Negocio.

Mediante el modelado de los procesos de negocio se identificaron las necesidades de cada área involucrada en el proceso de transferencia. Se realizó un análisis que permitió valorar el estado de desarrollo de los subprocesos en función de las actividades que realizan. Como parte de las técnicas aplicadas durante el desarrollo de la elicitación de requisitos, se definieron involucrados y artefactos, asociados a cada uno de los procesos del negocio.

A continuación se modela uno de los procesos identificados en la captura de requisitos, los demás procesos se podrán encontrar en los anexos.

Figura 2: Proceso: Transferencia Enviadas de Tesorería.

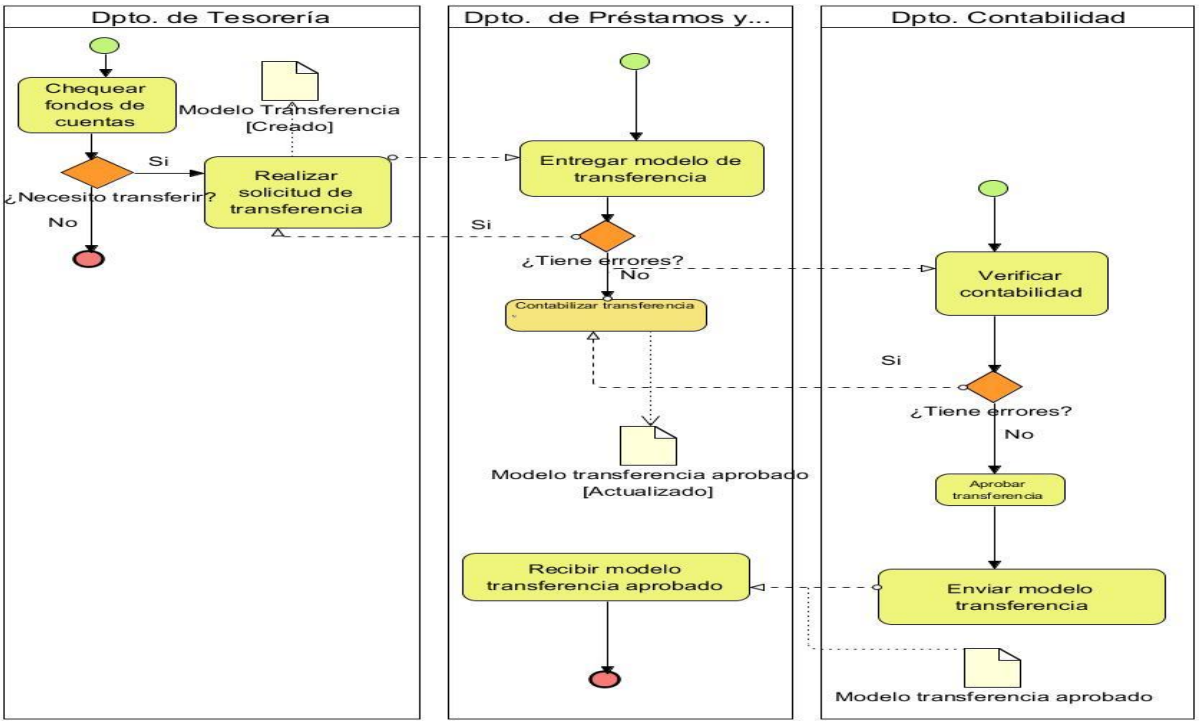
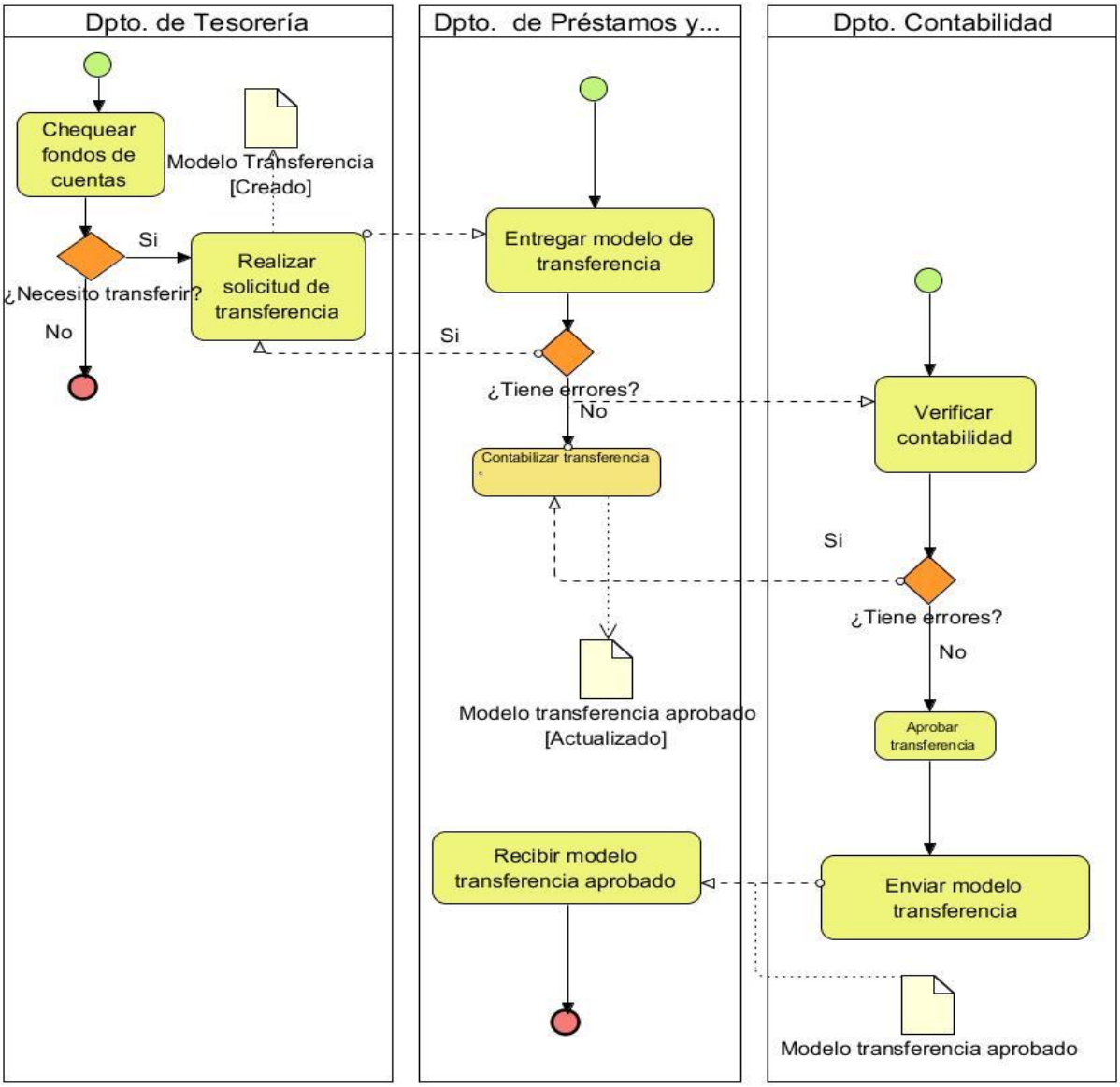


Figura 3: Subproceso: Registrar transferencia.



2.2.1. Involucrados y artefactos.

Los involucrados son aquellos que participan en los procesos del negocio, ya sean trabajadores de la entidad financiera o cualquier persona natural o jurídica que interactúe con las acciones o procesos que se desarrollan dentro de la misma. Los involucrados son todas aquellas instituciones, áreas o personas que interactúan con el negocio a la hora de una transferencia. Para hacer más entendible el papel que juegan en dicha interacción es necesario conocer algunos conceptos:

Tabla 1: Involucrados en la Gestión de Transferencia de Quarxo.

Involucrado	Descripción

Banco Emisor	Entidad financiera que emite la transferencia hacia otro banco, con su correspondiente mensaje SWIFT. Ver Tabla 2.
Banco Receptor	Entidad financiera que recibe la transferencia desde el banco emisor con el mensaje SWIFT correspondiente. Ver tabla 2.
Gestor o Supervisor	Es el encargado del control y revisión de los mensajes y modelos emitidos.
Área Operativa	Departamentos del Banco Nacional de Cuba que entre los procesos que realizan, incluyen la contabilización de sus operaciones. Dígase Deuda Externa, Carta de Crédito, Título Valores, Transferencia, Cuentas Clientes, otras.
Departamento de Contabilidad	Dpto. del Banco Nacional de Cuba encargado de la recepción de todos los movimientos que las áreas operativas del banco realizan y que una vez al realizar el cierre contable del día, dichos movimientos figuran en el reporte de estado de "Nuestras Cuentas", el cual es generado automáticamente una vez que se realice el cierre del día.
Departamento de Cuentas Corrientes y Órdenes de Pago	Dpto. encargado del ingreso de cuentas scrows[11] y cuentas corrientes, además de atender las transferencias recibidas de clientes particulares y de Instituciones, además del traspaso de cuentas.
Departamento de Tesorería	Dpto. del banco encargado entre sus actividades se encuentra el envío hacia al Dpto. de Préstamo y Depósito del modelo de Transferencias, que es el encargado de especificar el valor del importe y hacia que cuenta realizar la transferencia.

Los artefactos que son utilizados a la hora de realizar una transferencia, ya sea por el Dpto. de Cuentas Corrientes y Órdenes de Pago, Préstamo y Depósito, o cualquier otro departamento involucrado se describen a continuación. Se muestran los artefactos que de alguna manera son creados, eliminados o modificados durante el desarrollo de las actividades correspondientes a las transferencias, entre estos se encuentran los diferentes mensajes SWIFT que se emiten luego de una transacción contable.

Tabla 2: Artefactos que están presentes en las transferencias de Quarxo.

Artefactos	Descripción
MT-103	Mensaje que se emite de una transferencia de crédito única para un cliente particular desde un banco en el exterior, este mensaje es de tipo informativo y lo recepciona la Gerencia Cuentas Corrientes
MT-299	Mensaje que se emite por el Dpto. de Préstamo y Depósito hacia las

[11]Cuentas que se establecen en virtud de un acuerdo de financiamiento con una institución financiera y representan una garantía para el prestamista, ya que en las mismas se depositan fondos o fluyen ingresos que aseguran la amortización de las deudas.

	entidades bancarias luego de haber realizado la transferencia, este mensaje es de tipo informativo.
MT-202	Mensaje que se emite por el Dpto. de Préstamo y Depósito hacia las entidades bancarias luego de haber realizado la transferencia, este mensaje es conocido como "Orden de Pago".
MT-999	Mensaje que se emite después de haber rechazado una transferencia de tipo recibida a favor de clientes particulares.
Modelo de "Transferencia Enviada" al Dpto. de Préstamo y Depósito	Modelo que se emite al Dpto. de Préstamos y Depósito el cual describe la institución que van a enviar la transferencia, así como el importe de la misma.
Modelo de "Transferencia Recibida" emitida por el Dpto. de Depósito	Modelo que se emite al Revisor después de haber realizado una transferencia recibida por parte del BFI.

2.2.2. Descripción de los procesos de negocios

Para el desarrollo del subsistema de Transferencia fue necesario identificar los procesos del negocio, ya que estos constituyen el punto de partida para la creación de las funcionalidades del sistema. A continuación se presenta un breve resumen de algunos procesos de negocio referentes a Transferencias Enviadas, que se realiza en el Dpto. de Préstamos y Depósitos y las Transferencias Recibidas a favor de clientes particulares con el propósito de facilitar en detalles cómo es el método de realización de los mismos, el resto de los procesos y subprocesos se detallan en los anexos.

Tabla 3: Descripción del proceso Transferencia Enviadas de Tesorería.

Nombre: Transferencia Enviadas de Tesorería
Entrada(s) :
✓ Modelo de Transferencia
Resumen:
El Dpto. de Tesorería envía hacia el Dpto. de Préstamos y Depósito un modelo (Modelo de Transferencias) de carácter urgente o no. El operador de este Dpto. es el encargado revisar este modelo para que no falte ningún dato. Una vez realizada la revisión del modelo comienza a insertar el importe, la cuenta de donde se va a extraer ese importe, la cuenta hacia donde van a transferir el dinero. Una vez concluido la entrada de estos datos el operador contabiliza, imprime la contabilización realizada y la envía para que el Supervisor y el J' de Dpto. lo revise y lo firmen.
Salida(s) :
➢ Mensaje MT-202. ➢ Mensaje MT-299. ➢ Contabilización de la Transferencia enviada impresa.

➢ Recepción del Mensaje Swift sobre la transferencia enviada.
Reglas del Negocio:
Se podrá realizar la Transferencia si el modelo que envían al Dpto. de Préstamos y Depósito se encuentra con todos los datos que necesita el operador y con la firma del Dpto. de Tesorería. Se podrá enviar la transferencia si están presentes en la contabilización la firma del J' del Dpto. y la de Supervisor.

Tabla 4: Descripción del proceso Transferencia Recibida a clientes particulares.

Nombre: Transferencia Recibida a clientes particulares
Entrada(s) :
✓ Mensaje de la transferencia MT-103.
Resumen:
El mensaje llega al Dpto. de Cuentas Corrientes y Órdenes de Pago, el operador registra los datos correspondientes a la transferencia, la cancela, y envía un mensaje rechazando el pago de las mismas.
Salidas(s):
✓ Mensaje MT-999.
Reglas del Negocio:
Se podrá realizar el rechazo de la transferencia si se encuentra con el mensaje MT-103 donde se encuentran los datos de la misma.

Tabla 5: Descripción de la Transferencia enviadas BFI cubriendo sobregiros.

Nombre: Transferencia Recibidas de BFI
Entrada(s):
✓ Modelo de Transferencia.
Resumen:
Del Dpto. de Conciliaciones se envía un modelo para que el Dpto. de Préstamos y Depósito realice la transferencia correspondiente hacia el BFI, para así realizar un traspaso de fondos cubriendo sobregiros. En dependencia de la factibilidad de la operación se realiza un débito hacia la cuenta que tiene BNC o un crédito. El operador entra los datos correspondientes y contabiliza, luego imprime la contabilización para que el Supervisor lo revise.
Salida(s) :
✓ Contabilización de la transferencia impresa.
Reglas del Negocio:
Se podrá realizar la transferencia de este tipo si el Modelo de Transferencia está con las firmas y los datos correspondiente a la operación asignada.

2.2.3. Mejoras a los procesos de negocio.

Al concluir el análisis realizado acerca de los procesos del negocio se determinaron un grupo de mejoras para darle solución a los problemas existentes del Dpto. Préstamos y Depósito y el Dpto. de Cuentas Corrientes y Órdenes de Pago del Banco Nacional de Cuba a la hora de realizar una transferencia.

Las mejoras que el sistema proporciona son:

- Permitirá realizar las transferencias enviadas de Tesorería.
- Permitirá emitir los mensajes MT-103, MT-299, MT-202.
- Permitirá registrar las transferencias recibidas de instituciones extranjeras a clientes particulares.
- Permitirá cancelar las transferencias recibidas de instituciones extrajeras.
- Permitirá emitir el mensaje de rechazar las transferencias recibidas de instituciones extranjeras a clientes particulares por medio del MT-999.
- Permitirá registrar las transferencias que se envían al BFI cubriendo sobregiro.
- Permitirá registrar las transferencias enviadas hacia otras instituciones bancarias.
- Permitirá imprimir las transferencias de cualquier tipo una vez finalizada.
- Permitirá imprimir los mensajes emitidos, después de haber realizado la transferencia.

2.3. Análisis del sistema.

2.3.1. Introducción

En el capítulo se realiza una descripción de las características del sistema a partir del modelo del negocio; se identifican involucrados y artefactos, se describen las reglas del negocio y se definen cuáles son los procesos a automatizar. Se identifican las funcionalidades que tendrá el subsistema Transferencia de Quarxo, así como una definición de cada requerimiento. Se muestran los artefactos generados en el análisis de la propuesta solución, se analizan los casos de usos críticos, definiendo las clases del análisis para cada uno, así como la organización de los módulos mediante un diagrama de paquetes.

2.3.2. Definición de los requerimientos funcionales.

Los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea. Mediante la elicitación de requisitos se definieron los requisitos que el sistema debe cumplir. Los requisitos funcionales más importantes para la propuesta son:

- RF1 Registrar Tipos de Transferencia.
- RF2 Actualizar Tipos de Transferencia.

- RF3 Eliminar Tipos de Transferencia.
- RF4 Consultar Tipos de Transferencia.
- RF5 Envío de Mensaje SWIFT.
- RF6 Imprimir los mensajes SWIFT.
- RF7 Agregar comisiones.
- RF8 Modificar comisiones a cobrar según el Tipo de Transferencia.
- RF9 Buscar Transferencia.
- RF 10 Registrar Transferencia
- RF11 Eliminar Transferencia.
- RF12 Cancelar Transferencia.
- RF13 Rechazar Transferencia.
- RF14 Consultar Transferencia.

2.3.3. Definición de actores y casos de uso.

El diagrama de Casos de Uso es la relación entre los actores y los casos de uso. Este diagrama es una serie de eventos que hace A continuación se describe los actores de los casos de uso en con que cuenta el subsistema de Transferencia:

Tabla 6: Actores de los Casos de Uso.

Actor	Descripción
Usuario	Usuario del sistema que está encargado de registrar las transferencias enviadas de Tesorería, transferencias recibidas del BFI, el registro de las transferencias enviadas a los clientes particulares, encargado además de imprimir la contabilidad y los mensajes SIWFT, también del rechazo de las transferencias enviadas a clientes particulares.
Operador	Usuario del sistema encargado de realizar la contabilidad de la transferencia, modificar y agregar comisiones.
Supervisor	Usuario del sistema encargado de revisar la contabilización realizada por la transferencia.
Jefe de Área	Encargado de crear los tipos de transferencia que el operador va a realizar.

Los casos de uso son las funciones que proporciona un sistema para añadir valor a sus usuarios. Estos se han adoptado casi universalmente para la captura de requisitos de sistemas de software, sin embargo son más que simplemente una herramienta para la captura de requisitos; sino que dirigen todo el proceso de software. (14)

Las descripciones definidas y los prototipos de interfaz de los casos de uso del subsistema de Transferencia se muestran a continuación:

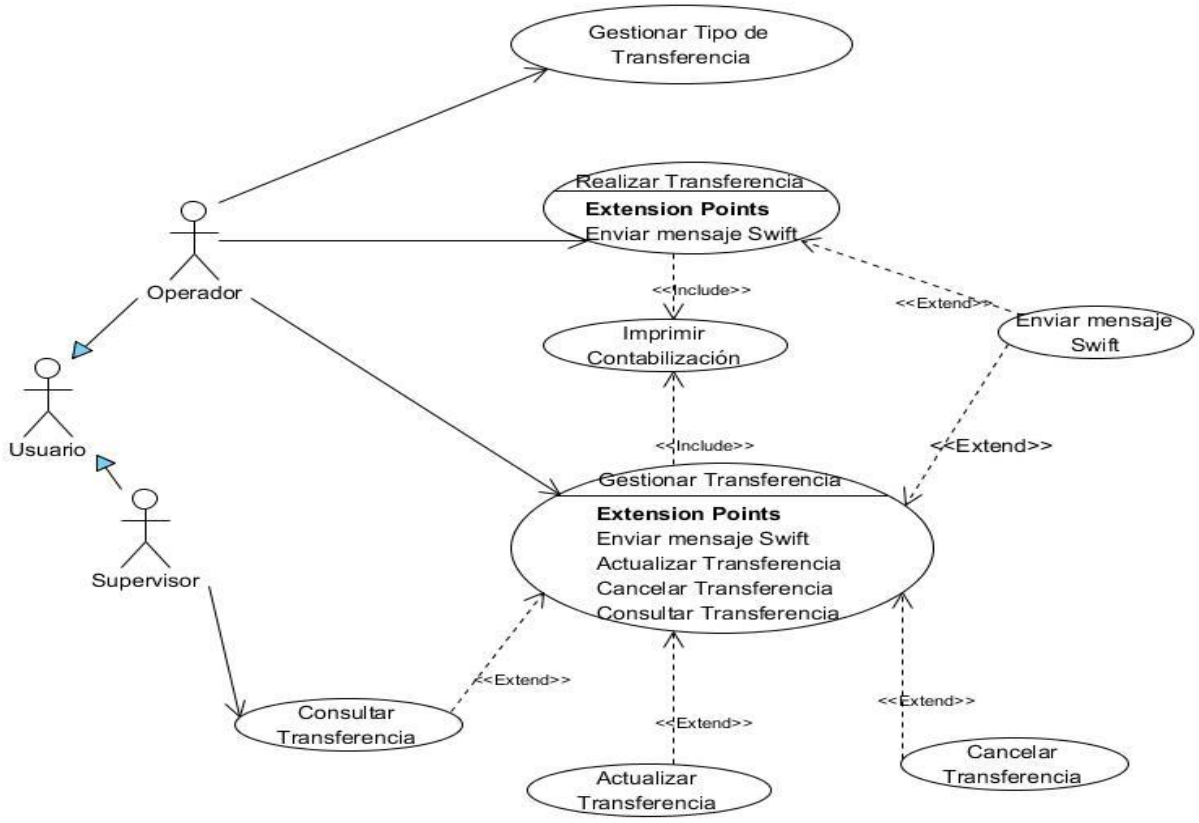
- Registrar transferencia Enviada de Tesorería.

- Registrar transferencia Recibidas de BFI.
- Registrar transferencia Recibida a favor de clientes particulares.
- Rechazar la transferencia Recibida a favor de clientes particulares.
- Cancelar transferencia.
- Imprimir contabilidad.
- Brindar la opción de enviar Mensaje SWIFT.
- Imprimir los mensajes SWIFT.
- Agregar comisiones.
- Modificar comisiones.
- Listar Tipos de Transferencia.
- Cancelar Tranferencia.
- Actualizar Transferencia.
- Actualizar Tipos de Transferencia.

2.3.4. Modelo de Casos de Uso del sistema.

A continuación se muestra el modelo de casos de uso del subsistema Transferencia:

Figura 4: Modelo de casos de uso del subsistema Transferencia.



2.3.5. Descripción de los Casos de Uso del sistema.

Las descripciones de casos de uso son reseñas textuales del caso de uso. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso. (32)

Tabla 7: Descripción del CU del sistema Crear Tipo de Transferencia

Caso de Uso:	Crear Tipo de Transferencia
Actores:	Administrador.
Resumen:	Permite crearlos tipos de transferencia en el sistema como son las Transferencia recibidas de Tesorería, Enviadas, de Traspaso, entre otras.
Precondiciones:	El administrador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1- Seleccionar la opción de Gestionar Tipo de transferencia en el menú principal del subsistema.	2- El sistema muestra los tipos de transferencias que existen en la actualidad. Existen las opciones: crear nuevo tipo de transferencia, consultar tipo de trasferencia, eliminar tipo de transferencia, modificar tipo de transferencia.
3- El usuario selecciona la opción deseada.	3.1- Si selecciona crear nuevo tipo de transferencia, ir a **Sección** Crear nuevo tipo de transferencia, 3.2- Si selecciona consultar tipo de trasferencia, ir a **Sección** Consultar tipo de trasferencia. 3.3- Si selecciona eliminar tipo de transferencia, ir a **Sección** Eliminar tipo de transferencia. 3.4- Si selecciona modificar tipo de transferencia, ir a **Sección** Actualizar tipo de transferencia.
	4- El caso de uso se termina.
Sección: Crear nuevo tipo de transferencia.	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona crear nuevo tipo de transferencia	2- El sistema muestra el formulario para introducir los datos correspondientes al tipo de transferencia, además de la posibilidad de seleccionar las comisiones y el tipo de cambio si este lo lleva.
3- El usuario introduce los datos en el	4- Se valida los datos introducidos. Si no está correcto.

formulario, además de la(s) comisiones que esta pueda realizar.	**Alternativa 1,** Datos Incorrectos. Se registra el tipo de transferencia.
	5- Finaliza la acción.
Sección: Consultar tipo de trasferencia	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona consultar tipo de transferencia	2- El sistema muestra el tipo de transferencia con los datos asociados a la misma.
	3- Finaliza la acción.
Sección: Eliminar tipo de transferencia	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona eliminar tipo de transferencia	2- El sistema elimina el tipo de transferencia seleccionada y muestra un mensaje: “Operación realizada satisfactoriamente”.
	3- Finaliza la acción.
Sección: Actualizar tipo de transferencia	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona actualizar tipo de transferencia.	2- El sistema muestra el formulario para introducir los nuevos datos para actualizar en el tipo de transferencia.
3- El usuario introduce los datos que va a actualizar en el formulario y selecciona la opción “Aceptar”.	4- Se valida los datos introducidos. Si no está correcto. **Alternativa 1,** Datos Incorrecto. Se actualiza los datos en el formulario referente al tipo de transferencia y se selecciona “Aceptar”. Los datos quedan registrados en la Base de Datos.
	5- Finaliza la acción.

Tabla 8: Descripción del CU del sistema Gestionar Transferencia.

Caso de Uso:	Gestionar Transferencia
Actores:	Operador.
Resumen:	Permite registrar en el sistema todas las transferencias de cualquier tipo, que hayan sido creadas en el gestionar tipo de transferencia.
Precondiciones:	El operador debe estar autenticado en el sistema.
Flujo normal de eventos	
Acción del Actor	**Respuesta del Sistema**
1- Seleccionar la opción de Gestionar transferencia en el menú principal del	2- El sistema muestra las transferencias que existen en la actualidad.

subsistema.	Existen las opciones: crear nueva transferencia, consultar trasferencia, actualizar transferencia,.
3- El usuario selecciona la opción deseada.	3.1- Si selecciona crear nuevo tipo de transferencia, ir a **Sección** Crear nueva de transferencia, 3.2- Si selecciona consultar tipo de trasferencia, ir a **Sección** Consultar tipo de trasferencia. 3.4- Si selecciona actualizar tipo de transferencia, ir a **Sección** Actualizar tipo de transferencia.
	4- El caso de uso se termina.
Sección: Crear nueva transferencia.	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona crear nueva transferencia	2- El sistema muestra una interfaz con los campos necesarios para registrar la Transferencia.
3- El usuario introduce los datos en el formulario, además de la(s) comisiones, el tipo de cambio de moneda si esta operación lo lleva y el usuario selecciona la opción “Aceptar”.	4- Se valida los datos introducidos. Si no está correcto. **Alternativa 1,** Datos Incorrectos. Se registra la transferencia.
	5- El sistema contabiliza la Transferencia. Finaliza el caso de uso.
Sección: Consultar tipo de trasferencia	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona consultar transferencia	2- El sistema muestra la transferencia con los datos asociados a la misma.
	3- Finaliza la acción.
Sección: Actualizar transferencia	
Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
1- Selecciona actualizar transferencia.	2- El sistema muestra el formulario para introducir los nuevos datos para actualizar en la transferencia.
3- El usuario introduce los datos que va a actualizar en el formulario y selecciona la opción aceptar.	4- Se valida los datos introducidos. Si no está correcto. **Alternativa 1,** Datos Incorrecto. Se actualiza los datos en el formulario referente al tipo de transferencia y se selecciona “Aceptar”. Los datos quedan registrados en la Base de Datos.
	5- El sistema contabiliza la Transferencia. Finaliza el caso de uso.
Sección:Alternativa1, Datos Incorrectos	

Flujo normal de eventos	
Acción de Actor	**Respuesta del sistema**
	1. El sistema señala los datos incorrectos y muestra el mensaje “Entrada de datos no válidos”.
2. El usuario corrige los datos y selecciona la opción “Aceptar”.	3. El sistema pasa a la acción 4 del flujo normal de eventos.

2.3.6. Requisitos no funcionales.

Existen RNF que especifican el comportamiento del producto, la velocidad de ejecución, memoria requerida, requisitos organizacionales, requisitos externos, verificables, de usabilidad, fiabilidad, rendimiento, soporte, seguridad e integridad de la información.

Para que el usuario final u operador interactúe con las funcionalidades implementadas se necesitan los siguientes requerimientos NF

- Mozilla Firefox 3.6.
- Máquina virtual de Java 6.0.
- Procesador: Pentium IV o superior 2.0 GHZ o superior
- RAM: 256 MB (recomendado 512 Mb)
- Tarjeta de Red: 1.
- Adobe Reader 9.0

Funcionalidad

El sistema mostrará los errores en forma de mensajes. Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

Usabilidad

Los formularios serán estandarizados, por tanto:

- Los campos de texto tendrán un tamaño estándar, de acuerdo con el espacio que se tenga en el área de la página y en la medida que se llene esa área primaria agregar la barra de desplazamiento vertical.
- No se utilizarán textos extensos para las etiquetas de la interfaz de usuario

El menú de navegación estará disponible en todas las páginas.

En caso de que los resultados de las consultas tengan más de 10 coincidencias, estos se mostrarán de forma paginada en una tabla.

- Se mostrará en la parte inferior de la tabla el total de elementos encontrados, enlaces de navegación: ir hacia delante, hacia atrás o ir al inicio de los resultados mostrados.

Fiabilidad

El sistema estará disponible durante toda la jornada laboral del BNC.

Seguridad

El sistema permitirá la visualización de la información según el usuario autenticado.
El sistema implementará el uso de campos obligatorios y validaciones para garantizar la integridad de la información que se introduce por el usuario.

2.3.7. Validación de los requerimientos funcionales.

A continuación se presentan los prototipos Gestionar tipo de Transferencia y Gestionar Transferencia en el subsistema de Transferencia, que se utilizaron para validar los requisitos funcionales. Estos prototipos fueron mostrados al cliente el cual tuvo una buena aceptación, pues cubría las necesidades que en un inicio se planteaba.

Figura 5: Prototipo Gestionar tipo de Transferencia.

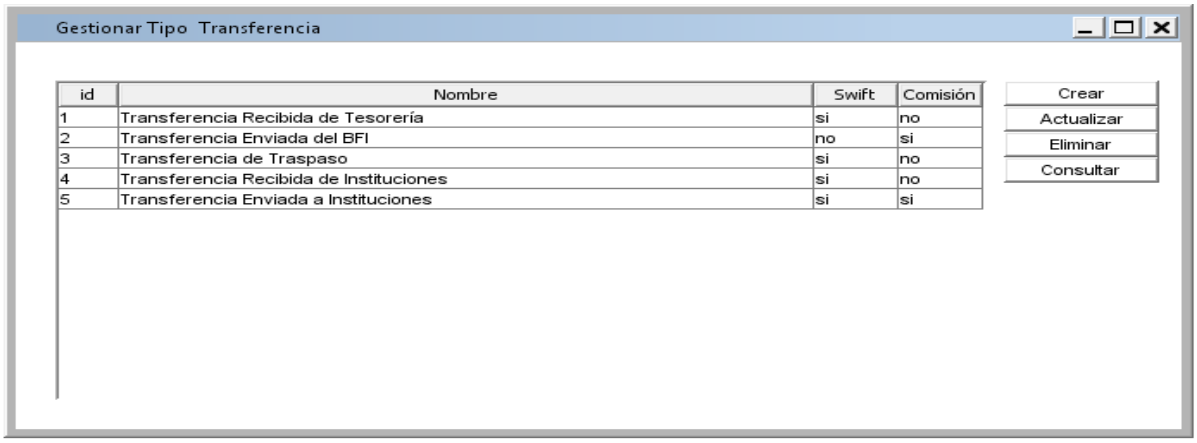


Figura 6: Prototipo Crear Tipo de Transferencia.

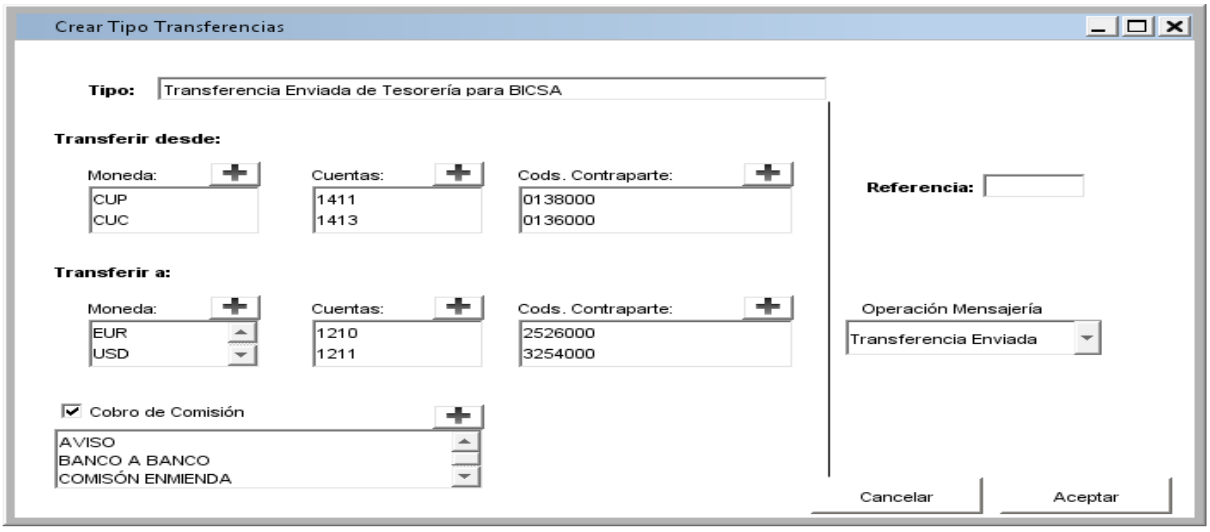
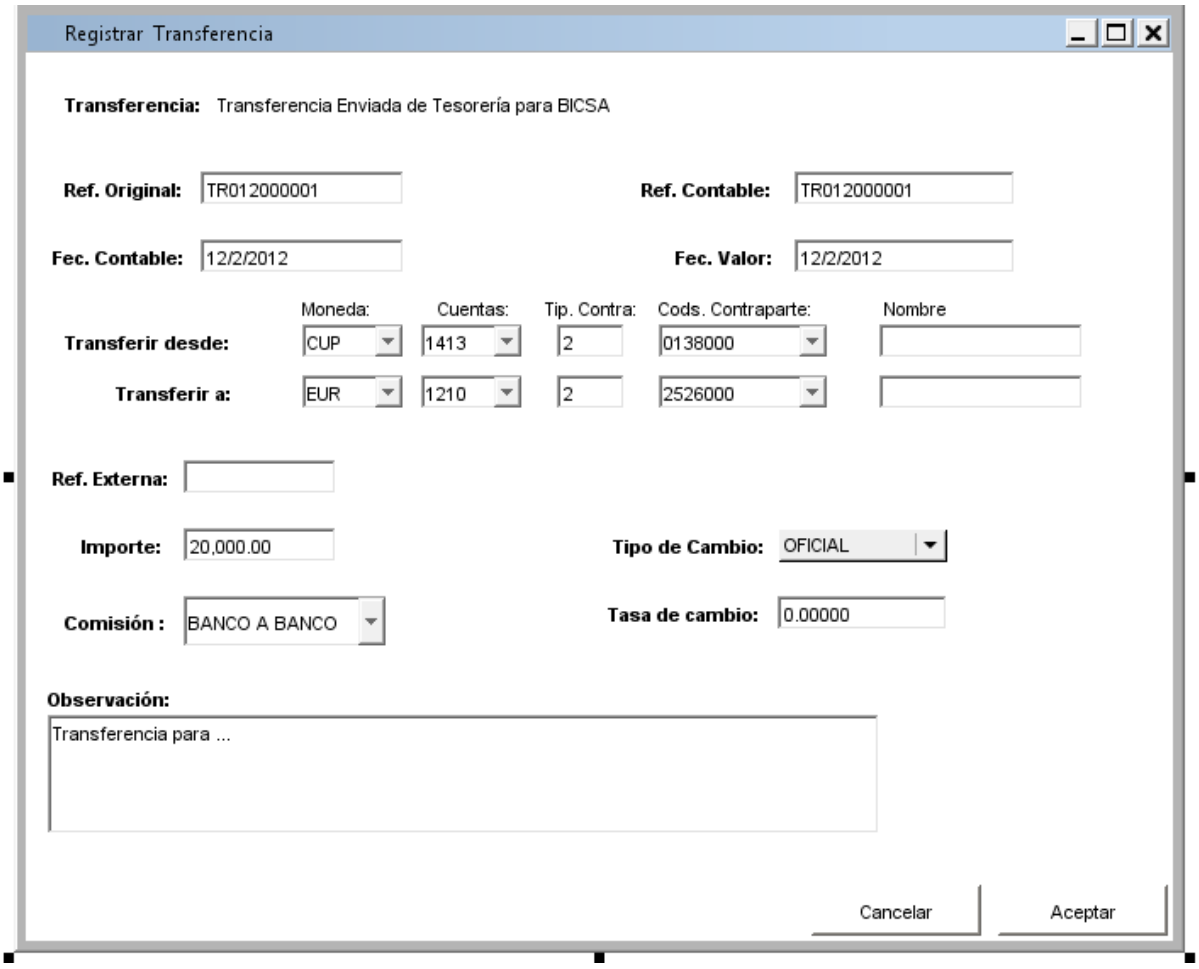


Figura 7: Prototipos no funcionales del Caso de Uso Registrar Transferencia.



2.3.8. Definición del modelo de clases del análisis.

El modelo de clases del análisis es un modelo que describe la realización de casos de uso, además de proporcionar una estructura centrada en el mantenimiento, la flexibilidad ante los cambios y la reutilización.

RUP como metodología establece una serie de estereotipos para este tipo de modelo, entre los que se encuentran a continuación:

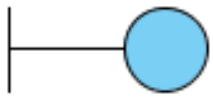


Fig. Clase Interfaz.

Fig. Clase Controlador.

Fig. Clase Entidad.

La clase interfaz representado en la figura anterior modela la interacción entre el sistema y sus actores, las clases controladoras son aquellas que coordinan la realización de un caso de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso y las clases entidades modelan la información que posee larga vida y que a menudo persistente.

En las siguientes figuras se encuentran los diferentes diagramas de clases del análisis de las operaciones de negocio que presentan los módulos de Gestionar Tipo Transferencia y Gestionar Transferencia.

Figura 8: Modelo de clase de análisis del módulo de Gestionar Tipo Transferencia subsistema Transferencia.

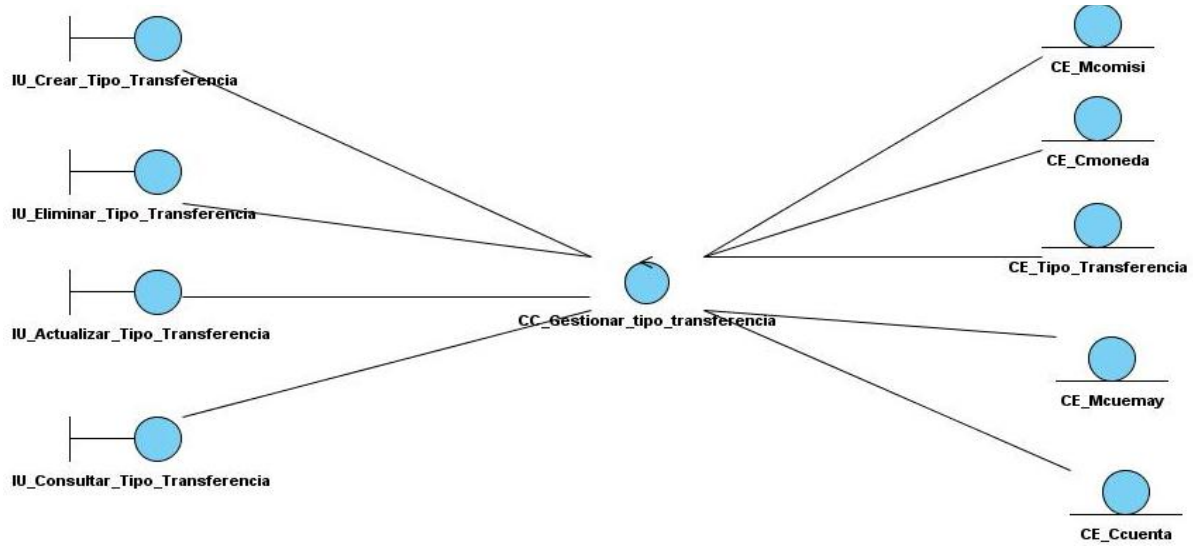
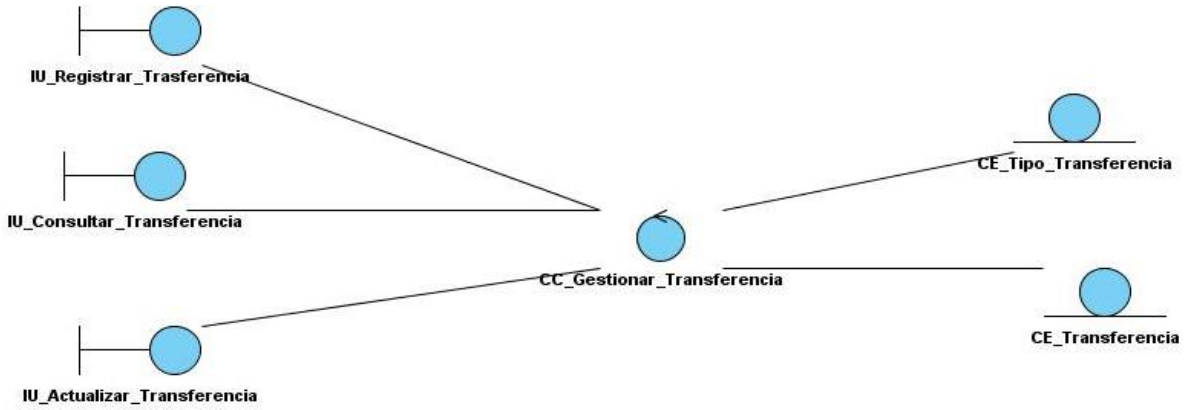


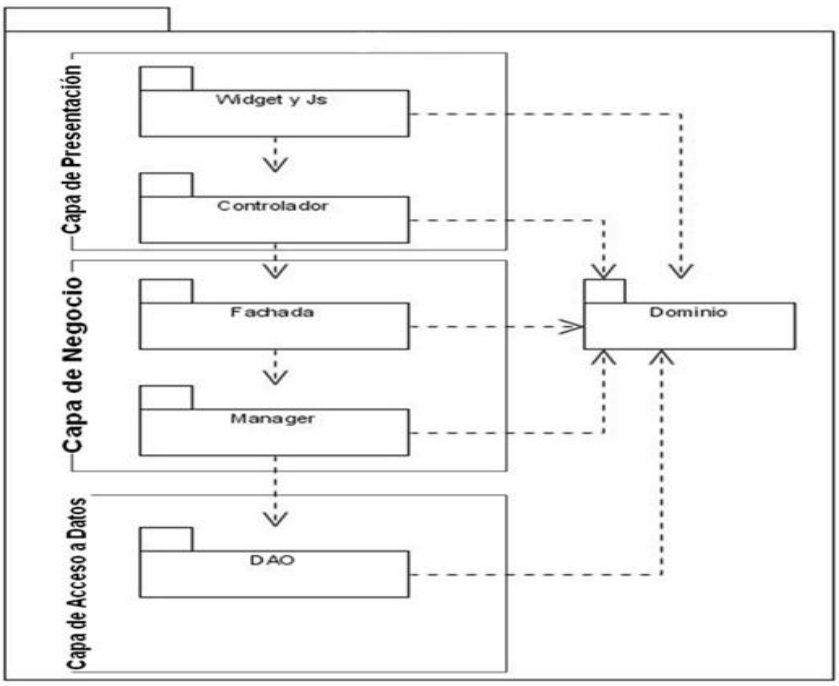
Figura 9: Modelo de clase de análisis del módulo de Gestionar Transferencia del subsistema Transferencia.



2.4. Fundamentación de la arquitectura.

El sistema Quarxo está implementado en una arquitectura por capas, de las cuales: *Capa de Presentación, Capa de Negocio, Capa de Acceso a Datos* son las principales; y además posee la *Capa de Dominio* con objetos del dominio que están presentes en el resto de la capas.

Figura 10: Estructura de las capas del Sistema Quarxo.



Las capas lógicas definidas para el sistema Quarxo se presentan a continuación:.

- **Capa de Presentación.**

En esta capa se desarrollará la lógica de presentación. En el servidor se utilizará Spring MVC para recibir, controlar y enviar una respuesta a las peticiones realizadas desde el cliente. En el cliente se utilizará la librería Dojo para generar las interfaces que interactuarán

con el usuario. La capa de presentación estará relacionada con la Capa de Negocios y Capa de Dominio.

- **Capa de Negocio.**

Esta capa estará divida en dos subcapas. Estas subcapas son Fachada y Manager. La Fachada será el punto de intercambio entre la capa de presentación y la capa de negocio. Esta capa no tendrá lógica de negocio sino que agrupará funcionalidades según su naturaleza para que pueda ser invocada desde la capa de presentación. La subcapa Fachada delegará a la subcapa Manager la realización de la lógica del negocio. Por otro lado la subcapa Manager tendrá la jerarquía de clases suficiente para implementar el negocio de la aplicación. Esta subcapa utilizará la capa de acceso a datos para obtener datos que están persistidos y la capa de dominio para generar los objetos de dominios. Desde la capa de negocio se envolverá todas las funcionalidades de la aplicación en diferentes niveles de transacción para evitar inconsistencia en los datos persistentes.

- **Capa de Acceso a Datos.**

En esta capa se realizarán todas las operaciones relacionadas con el gestor de base de datos y cualquier recurso que contenga información persistida. Para desarrollar esta capa se utilizará el patrón DAO (Data Access Object en inglés), en español Objetos de acceso a datos. Se utilizará la filosofía de un objeto DAO genérico utilizado. El diseño general de esta capa será una interface que responderá a un grupo de operaciones de una entidad de dominio persistente y la correspondiente implementación de esa interface.

- **Capa del Dominio.**

En esta última capa es donde se declaran todas las clases que representan entidades del negocio. Estas clases de dominio estarán presentes en todas las capas anteriormente descritas.

2.5. Conclusiones parciales.

En este capítulo se mostró una breve descripción de los principales artefactos, trabajadores y procesos de negocio, identificándose además algunas mejoras para dichos procesos. Se modeló el sistema quedando definido actores, casos de uso así como las especificaciones de cada caso de uso. Posteriormente se identificaron y validaron los requerimientos funcionales para dicho subsistema por medio de los prototipos, además, se especificaron cuáles eran los estereotipos a utilizar en el modelo de clases del análisis y se modelaron sus principales clases. Al finalizar se realizó una fundamentación de la arquitectura para su posterior diseño.

Capítulo 3: Diseño e Implementación.

3.1. Introducción.

Este capítulo se presenta los diagramas correspondientes al diseño de la solución. Además se muestran algunos ejemplos del desarrollo de la implementación de la solución propuesta del subsistema de Transferencia de Quarxo en si versión 2.0.Se abordan aspectos esenciales a tener en cuenta a la hora de implementar, como son los estándares de codificación, la iteración de los componentes en sus diagramas correspondientes, la descripción de sus clases y funcionalidades Paquetes del diseño.

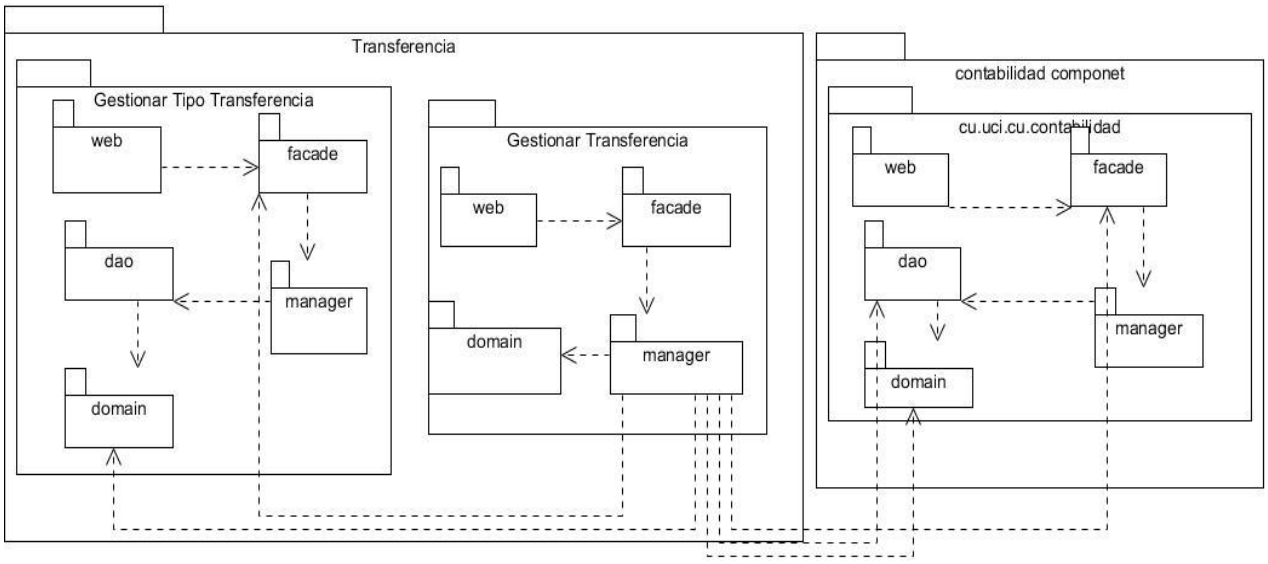
3.2. Paquetes del diseño.

Un paquete de diseño es una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes. Se utiliza para estructurar el modelo de diseño dividiéndolo en pequeñas partes. Para el desarrollo de este artefacto a continuación se detallan el diagrama de paquetes, las clases del diseño y la realización de casos de uso.

3.2.1. Diagrama de paquetes.

Con el diseño de estos diagramas se obtiene una visión del sistema de información orientado a objetos, organizándolo en subsistemas y/o módulos, agrupando los elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos. A continuación se muestra el diagrama de paquetes para los módulos de gestionar tipo de transferencia y los de gestionar transferencia.

Figura 11: Diagrama de paquetes.



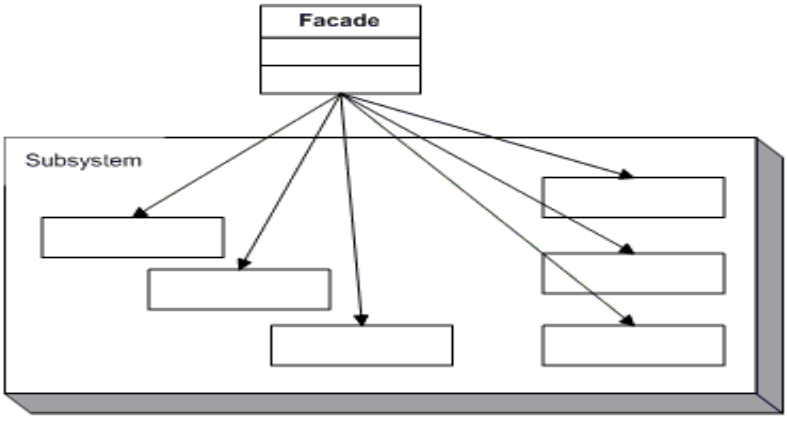
3.3. Clases del diseño.

Las clases del diseño son las que representan las clases utilizadas en la implementación del sistema, en estas clases se describen las responsabilidades de los métodos implicados en el desarrollo de una funcionalidad determinada. Los patrones de diseño que se describen a continuación son utilizados debido a que son orientados por la Arquitectura del proyecto Quarxo, además de las ventajas que ofrecen en su definición.

Para el diseño e implementación del módulo Gestionar Tipo Transferencia y gestionar Transferencia se utilizaron los patrones GRASP[12], creacionales y estructurales. A continuación se brinda una breve descripción de estos. El patrón MVC propone dividir la aplicación en cuatro capas el Modelo, la Vista y el Controlador, el modelo es la representación del dominio o datos del sistema, la vista se encarga de presentar la interfaz al usuario, en sistemas web, esto es típicamente HTML, aunque pueden existir otro tipo de formatos y el controlador es el encargado de escuchar los cambios en la vista y enviarlos al modelo el cual le regresa los datos a la vista como un ciclo. Cuando se aplica este patrón los accesos a la base de datos se hacen en el modelo, la vista y el controlador no deben de saber si se usa o no una base de datos. El controlador es el que decide que vista se debe de imprimir y que información es la que se envía.

Fachada: Este patrón permite simplificar el acceso a un conjunto de clases o interfaces. Proporcionar una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La fachada satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas. Un ejemplo del uso de este patrón se ve evidenciado en la creación de la interfaz GestionarTipoTransferenciaFacade.

Figura 12: Diagrama correspondiente al Patrón Fachada.



[12] **G**eneral Responsibility **A**ssignment **S**oftware **P**atterns.(Patrones de Software para la Asignación General de Responsabilidades)

Data Access Object (DAO): El Objeto de Acceso a Datos es un patrón que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una base de datos o un archivo. Permite abstraer y encapsular los accesos, gestionar las conexiones a la fuente de datos y obtener los datos almacenados. Se evidencia la utilización de este patrón en la creación de todas las clases DAO, ejemplo de esto es la clase TipoTransferenciaDAO.

Experto: Es un patrón GRASP que se puede encontrar en diferentes clases en el sistema, como son los dominios, los cuales son expertos en tener los atributos que se les definen, también se les encuentra en las clases DAO, como TipoTransferenciaDAO, la cual es la responsable de listar todos los tipos de transferencia que existen.

Singleton: Es un patrón creacional que permite siempre que se solicite un objeto de una clase determinada, obtener la misma instancia. Este patrón lo utiliza particularmente el framework Spring a la hora de devolver instancias de un bean[13] manejado en una misma sesión.

3.3.1. Diagrama de clases del diseño.

Los diagramas de clases son una representación gráfica que muestra una colección de elementos declarativos del modelo. Son de gran utilidad porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación.

En el diagrama de clases del diseño **Crear tipo de transferencia** interactúan varios grupos de clases, a continuación se describen algunas de ellas:

- **TipoTransferenciaMultiActionController.**

Esta clase es la encargada de atender las peticiones que están relacionadas con el caso de uso Crear tipo de transferencia, entres sus métodos se encuentran **listarMoneda**(), y **listarComisión**(). El método **listarMoneda()** es el encargado de devolver a la vista las monedas que se encuentran en la base de datos para que el usuario pueda seleccionar aquellas que irán en el tipo de transferencia. El método **listarComisión**() es el encargado de devolver a la vista las comisiones que se encuentran en la base de datos para que el usuario pueda seleccionar aquellas que irán en el tipo de transferencia.

- **TipoTransferenciaFacadeImpl.**

[13] **bean:** Se le denomina en el contexto de Spring a un objeto o instancia de una clase que es creado y manejado por el contenedor de Spring.

Esta clase está creada para que se pueda tener acceso a la implementación del negocio.

- **TipoTransferenciaManagerImpl.**

Esta clase es la encargada de manejar el negocio del módulo Gestionar Tipo Transferencia. Entre sus métodos se encuentra **obtenerDatosTipoTransferencia(),** el cual obtiene los datos correspondientes al tipo de transferencia. Otro de los métodos es **persistirTipoTransferencia(),** por el cual se puede persistir los datos introducidos por el usuario en la bases de datos.

Figura 13: Diagrama de clases del diseño: Crear tipo de transferencia. Capa de Presentación

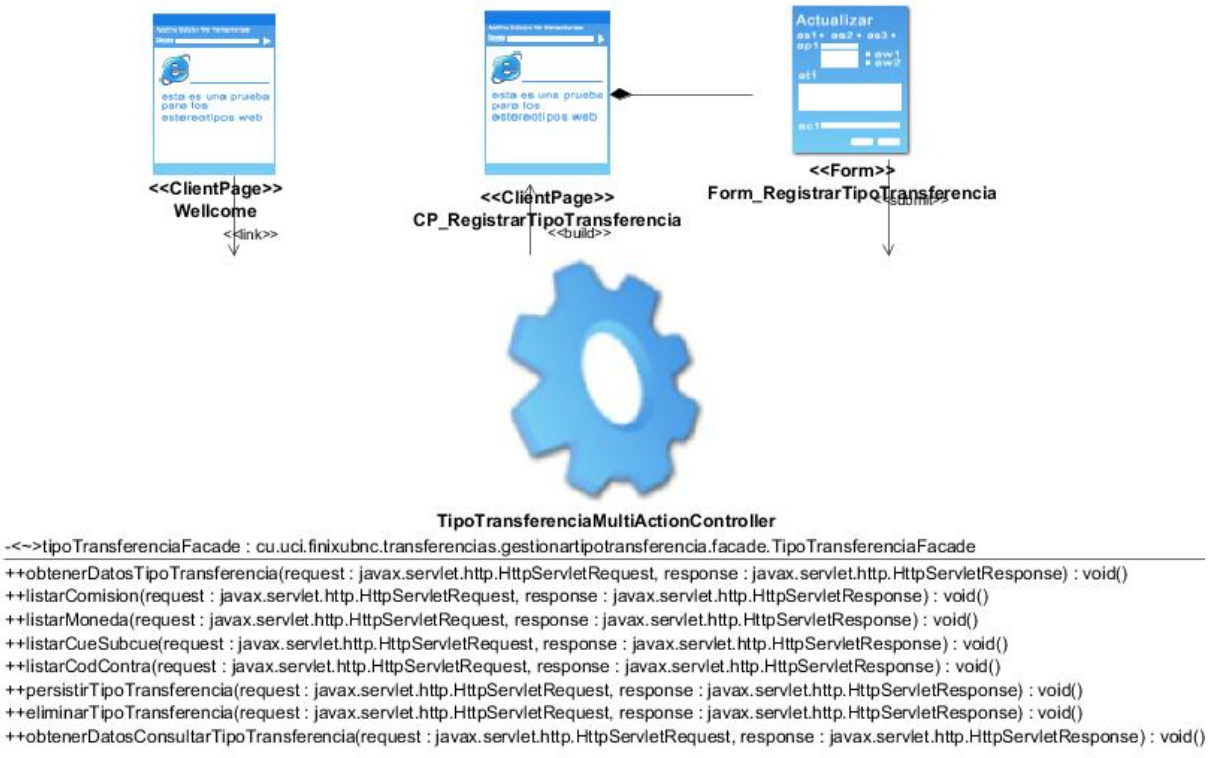


Figura 14: Diagrama de clases del diseño: Crear tipo de transferencia. Capa de Negocio.

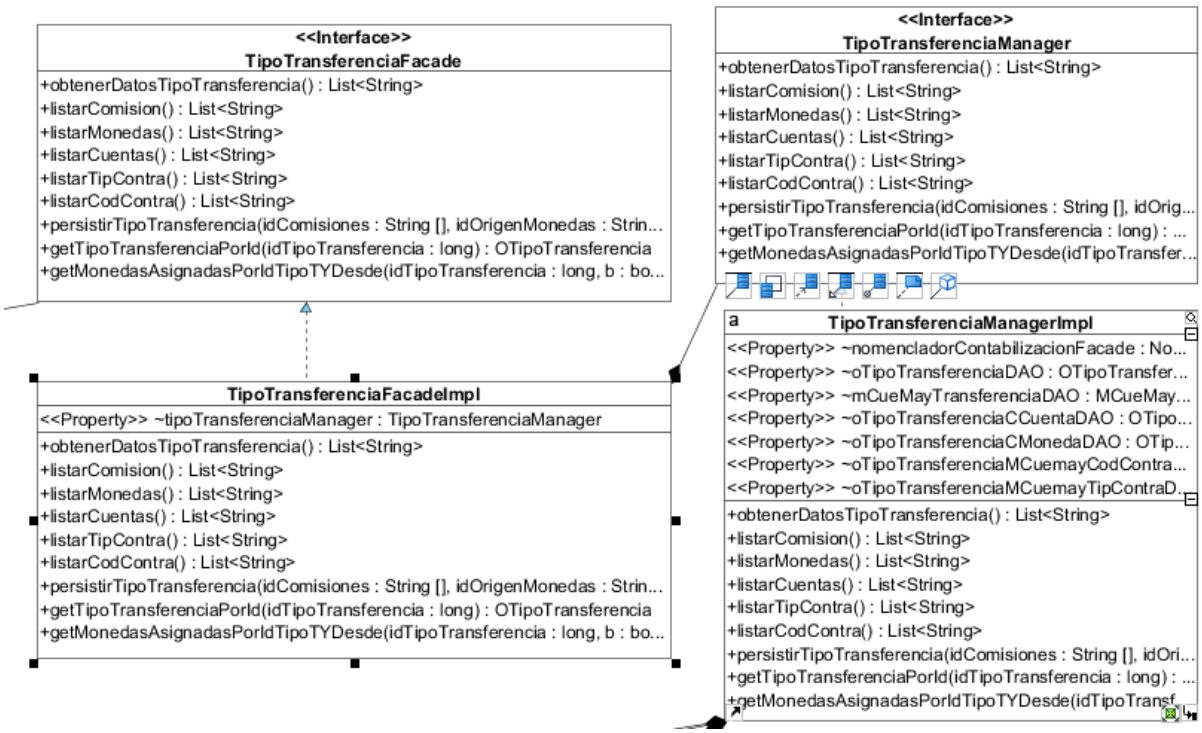
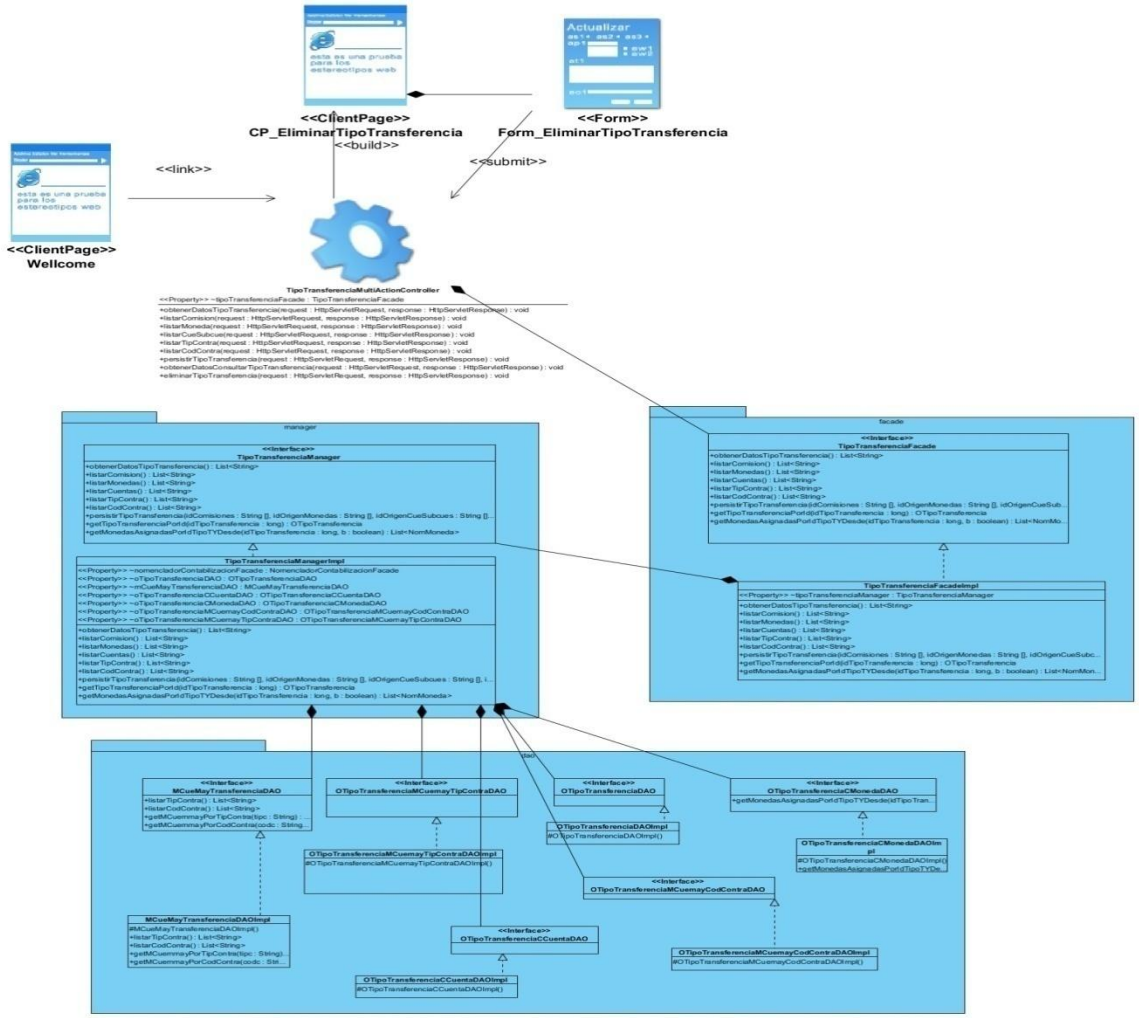


Figura 15: Diagrama de clases del diseño: Crear tipo de transferencia. Acceso a Datos.

Los diagramas anteriores de clases del diseño son los que modelan el CU Crear Tipo de Transferencia, que se ejecuta cuando se quiere crear un nuevo tipo de transferencia. La que se encarga de atender las peticiones que se realizan en la capa de presentación es TipoTransferenciaMultiActionController, además accede a través de la fachada TipoTransferenciaFacadea las funcionalidades que están implementadas en la clase TipoTransferenciaFacadeImpl, la que posee los métodos que responden al negocio del CU, y el acceso a datos se realiza a través de las clases:

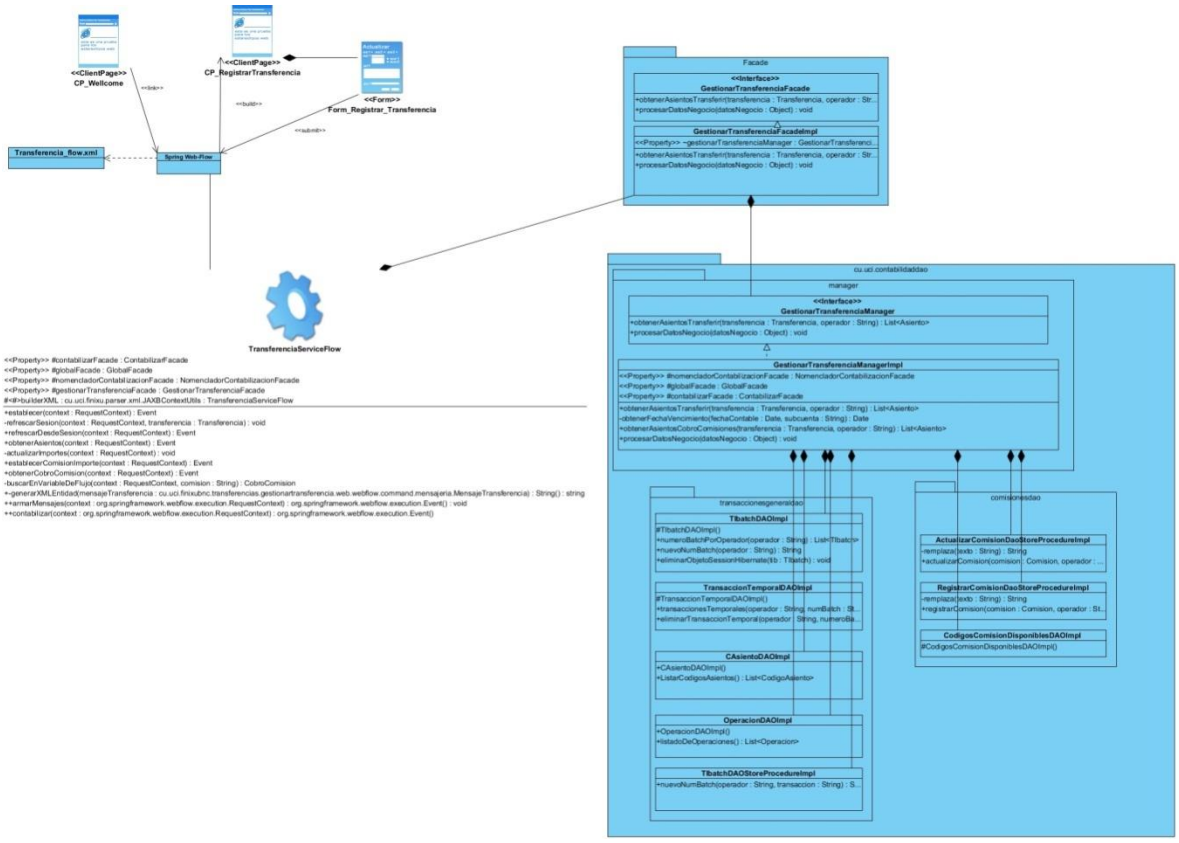
- **MCueMayTransferenciaDAO,**
- **OTipoTransferenciaMCuemayTipContraDAO.**
- **OTipoTransferenciaMCuemayTipContraDAO.**
- **OTipoTransferenciaDAO.**
- **OTipoTransferenciaMCuemayCodContraDAO.**
- **OTipoTransferenciaCMonedaDAO**

Figura 16: Diagrama del diseño: Eliminar tipo transferencia.



En el diagrama de clases del diseño al CU Registrar Transferencia del módulo Gestionar Transferencia se utilizó el framework Spring Webflow para la definición de los flujos. La clase TransferenciaServiceFlow es la encargada de atender a todos los eventos que se generen en la vista. La clase GestionarTransferenciaManagerImpl se encarga de la implementación de todo el negocio que se realiza en el caso de uso, esta clase usa las clases que se encuentran en el paquete cu.uci.contabilidaddao del subsistema de Contabilidad, en este paquete se encuentran declaradas las interfaces para ahorrar espacio en la imagen generada.

Figura 17: Diagrama del diseño del CU: Registrar Transferencia.



3.3.2. Diagrama de Interacción.

Los diagramas de interacción muestran los objetos, así cómo los mensajes que pasan entre ellos dentro del caso de uso. Los diagramas de interacción capturan cómo es el comportamiento de los casos de uso y se expresan de dos formas: diagramas de secuencias y diagramas de colaboración.

A continuación se muestran dos de los diagramas de secuencia de la propuesta solución que mayor impacto posee.

Figura 18: Diagrama de secuencia: Crear tipo transferencia.

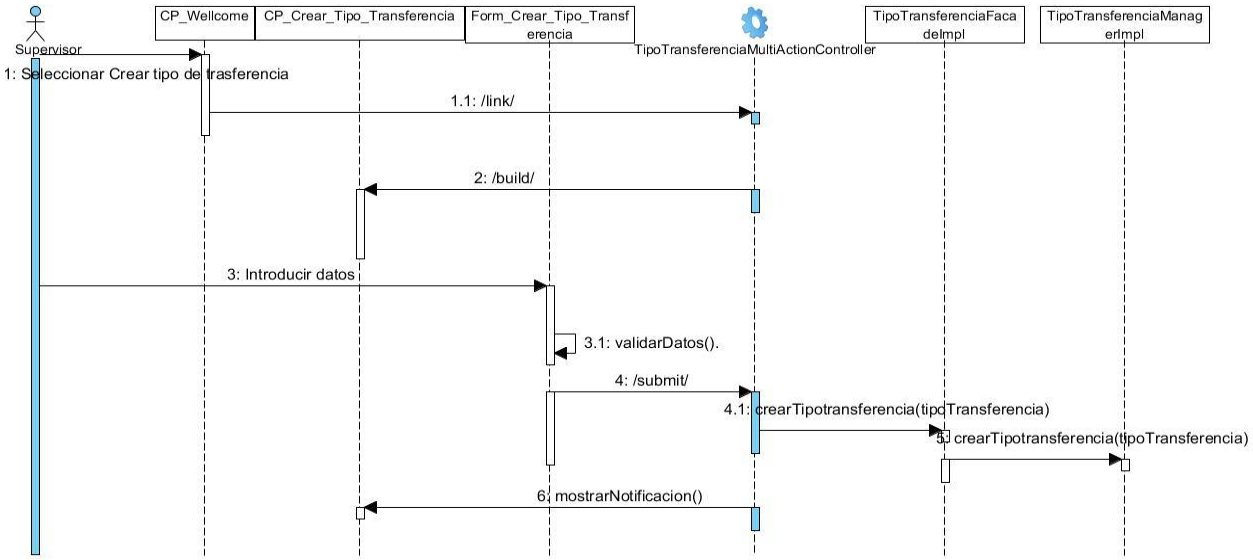
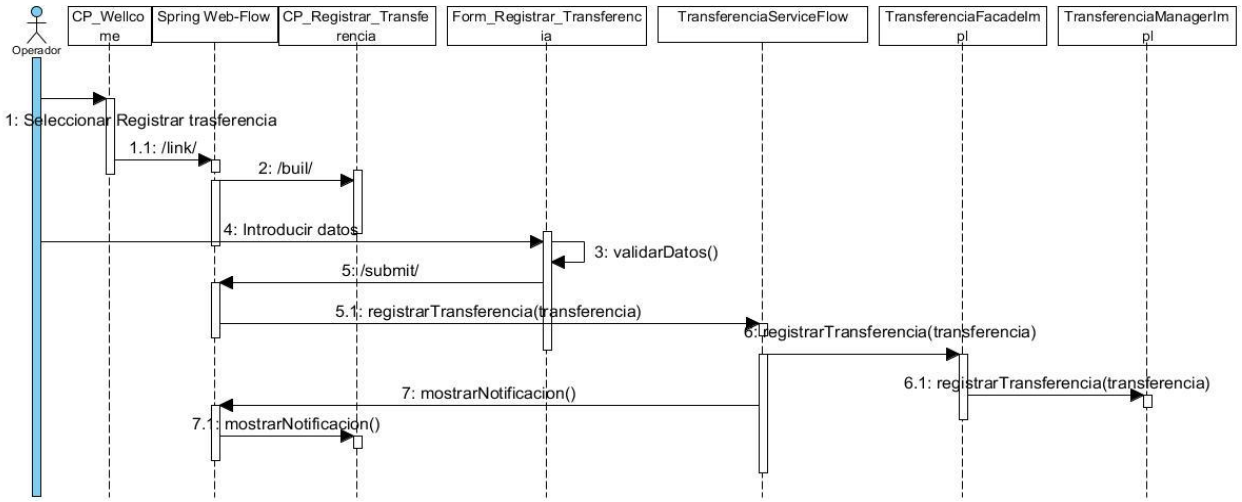


Figura 19: Diagrama de secuencia: Registrar transferencia.



3.4. Modelo de datos.

Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. Permiten describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí.

A partir de la base del núcleo del SABIC se desarrolló el sistema QUARXO. En su mayoría las tablas persistentes utilizadas para la gestión de las Transferencias fueron creadas y

acopladas al núcleo para lograr la integración, gestión de la información y la contabilización de las mismas.

Se creó un modelo de datos que responde al dominio de la propuesta de solución luego de realizar un estudio e identificar un conjunto de clases persistentes.

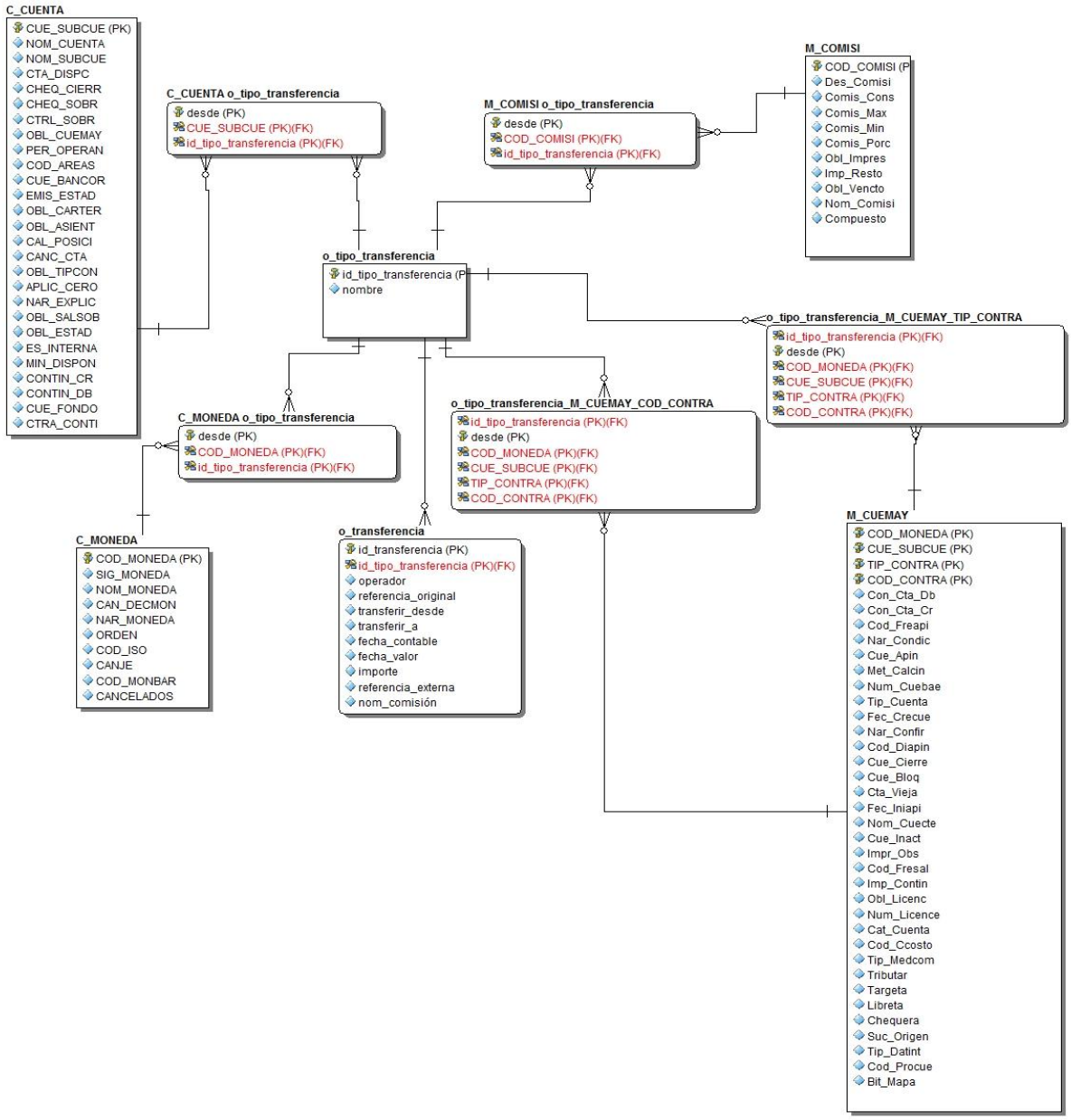
La tabla que se encarga de almacenar las transferencias que se crean es o_transferencia.

Las encargadas de almacenar los datos del tipo de transferencia son:

- o_tipotransferencia_mcuemay_tipcontra
- o_tipotransferencia_mcuemay_codcontra
- o_tipo_transferencia
- o_tipotransferencia_ccuenta
- o_tipo_transferencia_cmoneda
- o_tipotransferencia_comision

A continuación se muestra el modelo de datos.

Figura 20: Modelo de datos correspondiente al módulo Transferencia.

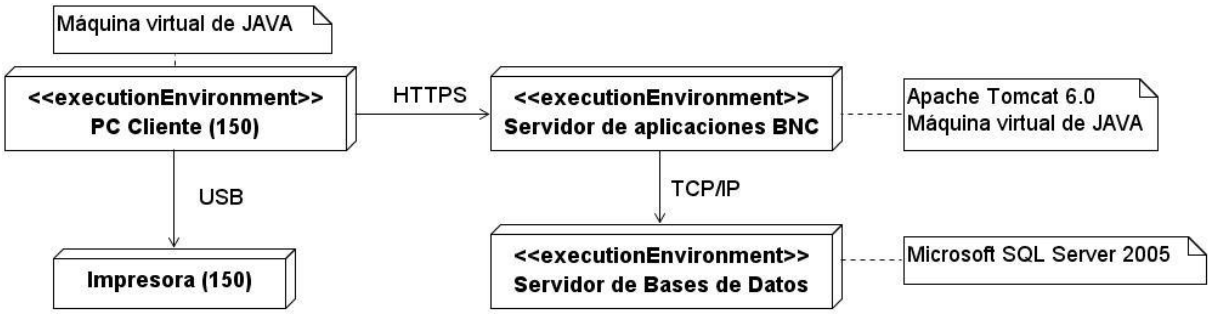


3.5. Implementación.

En la implementación se describe el modelo de implementación utilizado, el diagrama de componente, el estándar de codificación, así como las convenciones de nombres y la descripción de las principales clases utilizadas.

3.5.1. Diagrama de Despliegue.

A continuación el diagrama de despliegue correspondiente al subsistema Transferencia de Quarxo.



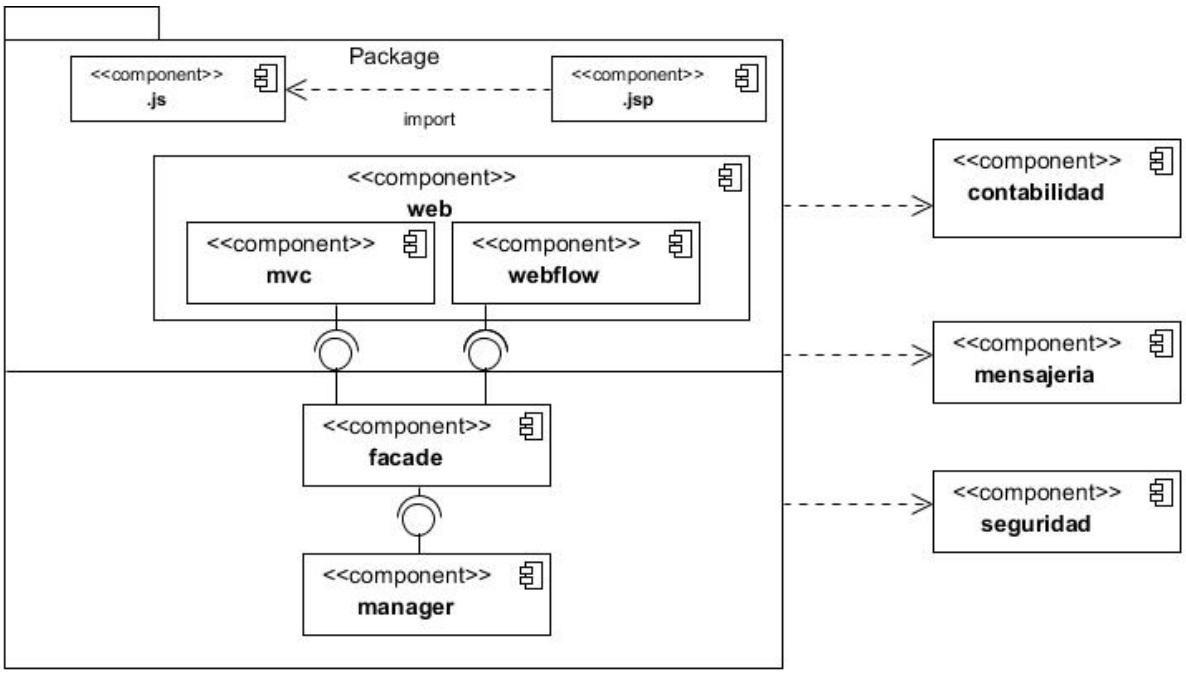
3.5.2. Modelo de implementación.

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (33)

3.5.3. Diagrama de componentes.

En el diagrama de componentes se muestran las relaciones entre las partes físicas y reemplazables del sistema. Estos diagramas contienen relaciones de dependencia que se utilizan para indicar que un componente se refiere a los servicios ofrecidos por otro componente. En el siguiente diagrama se muestran los componentes de los módulos siguiendo la arquitectura Modelo Vista Controlador.

Figura 21: Diagrama de componentes.



3.6. Estándares de codificación.

Los estándares de codificación son pautas de programación que están enfocadas a la estructura y apariencia física, esto facilita tanto la lectura, comprensión y mantenimiento del código. Un estándar de programación define la nomenclatura de las variables, objetos, métodos, funciones, además tiene que ver con el orden y legibilidad del código escrito.

3.6.1. Convenciones de nombres.

Como principio todos los nombres serán en español exceptuando aquellos que correspondan al nombre de un patrón conocido, ejemplo: Builder, Facade, entre otros.

- **Atributos.**

Los nombres de los atributos de las clases y variables internas comienzan con la primera letra en minúscula, en caso de que sea un nombre compuesto las siguientes palabras comenzarán con la primera letra en mayúscula.

Ejemplo: contabilizarFacade.

- **Métodos.**

Los métodos deben ser en infinitivo, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.

Ejemplo: obtenerAsientos().

- **Clases.**

Los nombres de las clases deben ser sustantivos, en el caso de ser compuestos tendrán la primera letra de cada palabra que lo forme en mayúscula. Los nombres de las clases deben ser simples y descriptivos. Además, se deben usar palabras completas, evitar acrónimos y abreviaturas (exceptuando los casos en los que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).

3.6.2. Nomenclatura según el tipo de clases.

Las clases que se encuentran dentro del paquete controller, se nombran adicionándoles el nombre del controlador de Spring del cual heredan al final del nombre de la clase.

Ejemplo: TipoTransferenciaMultiActionController.

Las clases que se encuentran dentro del paquete serviceFlow se les agrega la palabra ServiceFlow después del nombre de la clase.

Ejemplo: TransferenciaServiceFlow.

Las clases que se encuentran dentro del paquete facade se les agrega después del nombre la palabra Facade. Para el subpaqueteimpl se les nombra igual que la interfaz que implementan y se le agrega la palabra Impl.

Ejemplo: TransferenciaFacade, TransferenciaFacadeImpl.

Las clases que se encuentran dentro del paquete manager se les agrega después del nombre la palabra Manager. Para el sub paquete impl se les nombra igual que la interfaz que implementan y se le agrega la palabra Impl.

Ejemplo: TransferenciaManager, TransferenciaManagerImpl.

3.6.3. Descripción de las principales clases utilizadas.

A continuación se describen las clases más importantes desde el punto de vista funcional para el módulo gestionar tipo transferencia y gestionar transferencia.

Tabla 9: Descripción de la clase TipoTransferenciaMultiActionController.

Nombre: TipoTransferenciaMultiActionController	
Tipo de clase: Controladora	
Atributo	Tipo
tipoTransferenciaFacade	tipoTransferenciaFacade
Para cada responsabilidad:	
Nombre:	Descripción:
obtenerDatosTipoTransferencia(HttpServletRequest, HttpServletResponse)	Permite obtener los datos de los tipos de transferencia.
listarComision(http.HttpServletRequest, HttpServletResponse)	Permite listar todas las comisiones que posee la entidad bancaria.
listarMoneda(http.HttpServletRequest, HttpServletResponse)	Permite listar todas las monedas que posee la entidad bancaria.
listarCueSubcue(http.HttpServletRequest, HttpServletResponse)	Permite listar todos los CueSubcueque tiene la entidad

	bancaria.
listarTipContra(HttpServletRequest, HttpServletResponse)	Permite listar todos los Tipcontra que posee la entidad bancaria.
listarCodContra(HttpServletRequest, HttpServletResponse)	Permite listar todos los CodContra que posee la entidad bancaria.
eliminarTipoTransferencia(HttpServletRequest, HttpServletResponse)	Permite eliminar el tipo de transferencia seleccionado por el usuario.
persistirTipoTransferencia(HttpServletRequest, HttpServletResponse)	Permite persistir en la base de datos el tipo de transferencia creado por el usuario.

Tabla 10: Descripción de la clase TipoTransferenciaManagerImpl.

Nombre: TipoTransferenciaManagerImpl	
Tipo de clase: Manager	
Atributo	**Tipo**
nomencladorContabilizacionFacade	NomencladorContabilizacionFacade
oTipoTransferenciaDAO	OTipoTransferenciaDAO
mCueMayTransferenciaDAO	MCueMayTransferenciaDAO
oTipoTransferenciaCCuentaDAO	OTipoTransferenciaCCuentaDAO
oTipoTransferenciaCMonedaDAO	OTipoTransferenciaCMonedaDAO
oTipoTransferenciaMCuemayCodContraDAO	OTipoTransferenciaMCuemayCodContraDAO
oTipoTransferenciaMCuemayTipContraDAO	OTipoTransferenciaMCuemayTipContraDAO
Para cada responsabilidad:	
Nombre:	**Descripción**:
obtenerDatosTipoTransferencia()	Permite obtener todos los datos de los tipos de transferencia.
listarComision()	Permite listar todas las comisiones que posee la entidad financiera.
listarMonedas()	Permite listar todas las monedas de la entidad financiera.

listarCuentas()	Permite listar todas las cuentas de la entidad financiera.
listarTipContra()	Permite listar todas los TipContra de la entidad financiera.
listarCodContra()	Permite listar todas los CodContra de la entidad financiera.
persistirTipoTransferencia()	Permite persistir el tipo de transferencia creado por el usuario.
getTipoTransferenciaPorId()	Permite obtener el tipo de transferencia por el id.

3.7. Conclusiones parciales.

En el capítulo se detallaron los diagramas correspondientes al diseño de la solución y los diagramas de la implementación de la solución propuesta del subsistema de Transferencia de Quarxo. Se abordaron temas referentes a los estándares de codificación, ya que son pautas de programación que están enfocadas a la estructura y apariencia física, esto facilita tanto la lectura, comprensión y mantenimiento del código.

Capítulo 4: Validación de la Solución.

4.1. Introducción.

En este capítulo se realiza la validación del diseño aplicando las métricas correspondientes. Mediante la aplicación de las pruebas unitarias, se hace una breve descripción del as pruebas de caja blanca y caja negra que se realizaron al software.

4.2. Validación del diseño.

Para la presente validación del diseño se utilizó varias de las métricas orientadas a objetos las cuales son capaces de ayudar a la hora de evaluar el diseño antes de que un sistema se construya.

4.2.1. Tamaño Operacional de Clases (TOC).

Esta métrica fue propuesta por Lorenz y Kidd, la cual especifica que el tamaño general de una clase puede medirse determinando las siguientes medidas:

- Total de Operaciones (operaciones tanto heredadas como privadas de la instancia).
- El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por una clase.

Valores grandes de TOC representa gran responsabilidad de la clase. Esto implica la reducción de la reutilización de la clase y complica la implementación y las pruebas. De forma general, operaciones y atributos deben ser ponderados al determinar el tamaño de la clase. Para valores pequeños de TOC para una clase existe mayor posibilidad de que la clase pueda ser reutilizada.

Tabla 11: Parámetros de calidad para valores grandes de TOC.

Parámetros de Calidad	Valores Grandes de TOC
Reutilización	Reduce la reutilización de las clases.
Implementación	Complica la implementación.
Complejidad de las pruebas	Hace complejas las pruebas del sistema.
Responsabilidad	La clase debe tener gran responsabilidad.

Se le aplicaron las siguientes medidas o umbrales a estas métricas, teniendo en cuenta que son el número de las operaciones y/ atributos:

Tabla 12: Umbrales para TOC.

TOC	Umbral

Pequeño	<=20
Mediano	>20 y <=30
Grande	>30

A continuación se muestran las medidas a las principales clases de la solución:

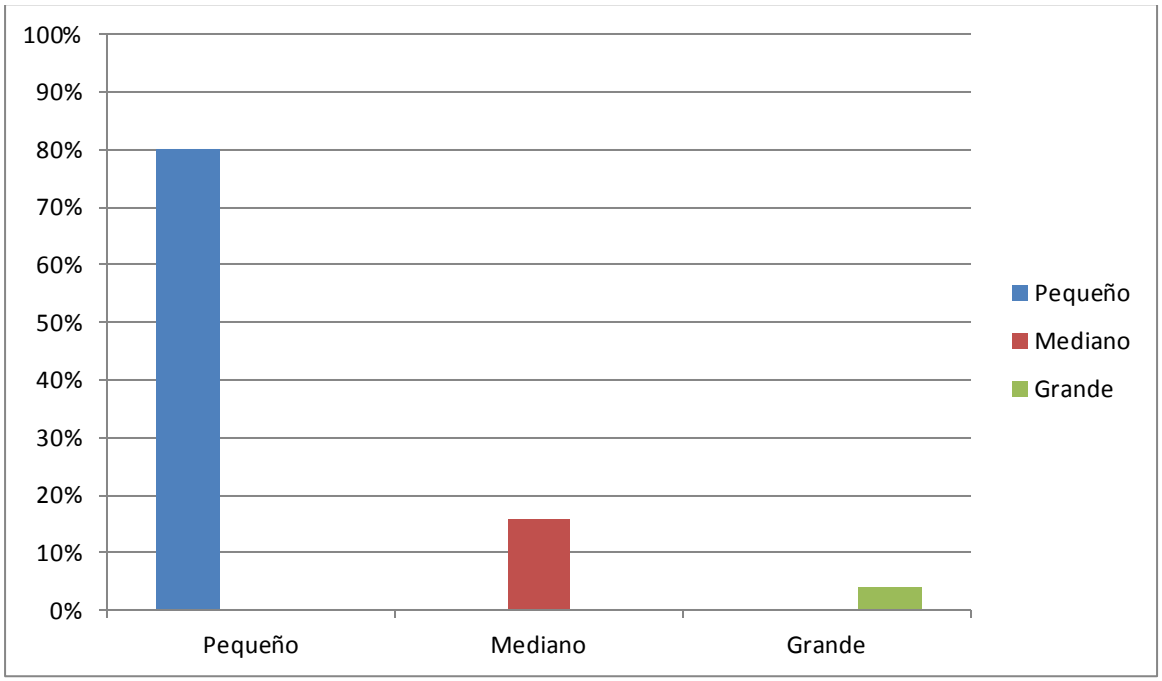
Tabla 13: Clases a las cuáles se le aplicó la métrica TOC

No	Clase	Atributo	Operaciones	Tamaño
1	OTransferenciaDAO	0	2	Pequeño
2	OTipoTransferenciaMCuemayCodContraDAO	0	2	Pequeño
3	OTipoTransferenciaDAO	9	9	Pequeño
4	OTipoTransferenciaCCuentaDAO	0	2	Pequeño
5	OTipoTransferenciaCMonedaDAO	0	2	Pequeño
6	OperacionDao	1	2	Pequeño
7	TipoTransferenciaFacadeImpl	0	15	Pequeño
8	TipoTransferenciaManagerImpl	0	15	Pequeño
9	GestionarTransferenciaFacadeImpl	0	6	Pequeño
10	GestionarTransferenciaManagerImpl	0	6	Pequeño
11	ContabilizarMultiAction	4	16	Pequeño
12	TipoTransferenciaMultiActionController	0	16	Pequeño
13	CobrarComisionesCuentaUnicaCommand	11	9	Mediano
14	TransferenciasMultiActionController	5	28	Grande
15	OperacionDao	0	1	Pequeño
16	TransferenciaMultiAction	4	14	Pequeño
17	ComisiónATransaccion	4	7	Pequeño
18	Asiento	35	56	Grande
19	AsientoId	15	6	Mediano
20	GestionarComisionATransaccionFacade	2	8	Pequeño
21	GestionarComisionATransaccionManager	0	8	Pequeño
22	AbstractMensajeriaIntegrationTemplateManager	2	12	Pequeño
23	TransferenciaComand	13	16	Mediano
24	TipoTransferenciaComand	8	10	Pequeño
25	Operacion	1	2	Pequeño

Resultados obtenidos de la métrica al sistema

Se le aplico la métrica TOC a 25 clases con un total de 114 atributos y 270 operaciones, con un promedio de atributos de 4.56 y un promedio de operaciones de un 10.8. Se obtuvieron 20 clases de tamaño pequeña, 3 de tamaño mediano y 2 de tamaño grande.

Figura 22: Por ciento de clases por tamaño



Después de mostrados los datos en las tablas anteriores se llega la conclusión que al poseer 21 clases de TOC pequeño, 3 mediano y 2 grandes, existe un aumento en la reutilización de las clases, además, tiene una baja responsabilidad no complica ni las pruebas ni la implementación.

4.3. Pruebas

4.3.1. Pruebas de Caja Blanca.

La prueba de caja blanca, denominada a veces **prueba de caja de cristal** es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

La prueba de caja blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. (34)

En el desarrollo de las pruebas de caja blanca se va a concentrar principalmente en la validación a través del framework de software libre JUnit, que cada uno de los módulos o segmentos de códigos que funcione correctamente. JUnit ofrece las funcionalidades necesarias para implementar cualquier proyecto que esté desarrollado en Java, además de que cuenta con una interfaz simple que informa si cada una de las pruebas realizadas o el conjunto de las mismas falla, son ignoradas o simplemente pasa la prueba.

Luego de definir los casos de prueba para las clases implementadas, se definieron e implementaron los ficheros de configuración para establecer la comunicación entre las diferentes capas de la aplicación que existen en la aplicación. Se identificaron los métodos a probar dentro de cada caso de prueba, así como el resultado esperado en cada uno. Al finalizar se procedió a realizar los casos de prueba diseñados.

A continuación se muestra un grupo de imágenes con las descripciones para mostrar la ejecución de los casos de prueba.

Figura 23: Declaración de los atributos en la clase de prueba

```
15 public class Test_JUnit extends TestCase {
16
17     TipoTransferenciaMultiActionController tipoTransferenciaMultiActionController;
18
19     SessionFactory sessionFactory;
20
21     private MockControl controlHttpServlet;
22     private HttpServletRequest mockhttpServletRequest;
23
24     private MockControl controlHttpResponse;
25     private HttpServletResponse mockhttpServletResponse;
26
27     private MockControl controlHttpSession;
28     private HttpSession mockHttpSession;
```

Se muestra la creación de una clase en la que se van a definir los casos de prueba implementados para probar las funcionalidades que se encuentran en la clase TipoTransferenciaMultiActionController. Primeramente se realiza una declaración de un atributo de dicha clase, luego dos mocks[14] para simular los objetos HttpServletRequest y HttpServletResponse que se le deben pasar a los métodos a probar, estos objetos son

[14] **Mocks:** *son objetos simulados que imitan el comportamiento de los objetos reales en forma controlada. Normalmente se crea un objeto mock para probar el comportamiento de algún otro objeto.*

controlados por objetos de tipo MockControl, por último uno de tipo SessionFactory[15] necesario para que Hibernate se comunique con la base de datos.

Figura 24: Implementación del método setUp() en la clase de prueba.

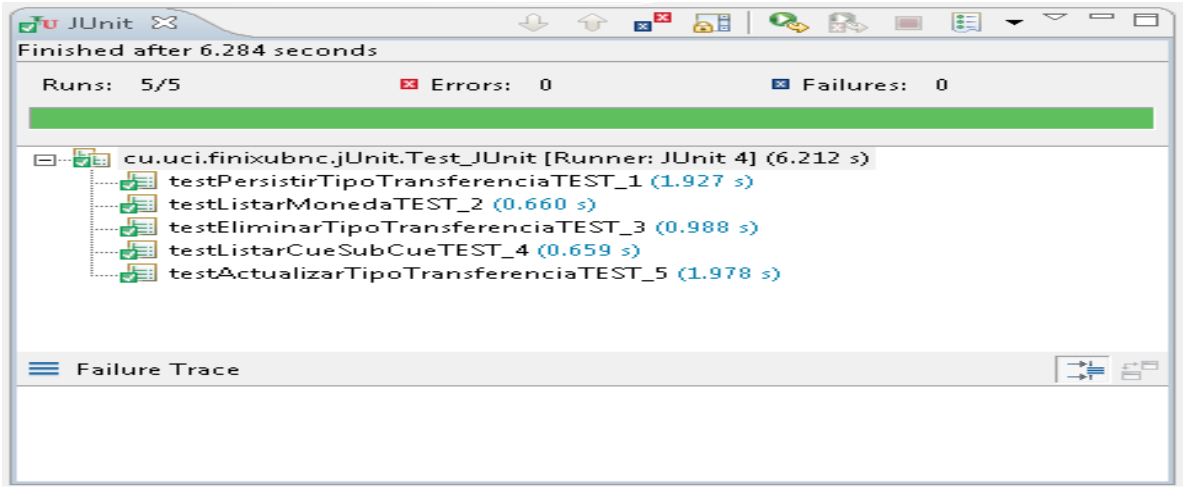
```
36    protected void setUp()throws Exception {
37
38        ApplicationContext context = new ClassPathXmlApplicationContext(
39        "classpath:cu/uci/finixubnc/jUnit/transferencias-context.xml");
40
41        tipoTransferenciaMultiActionController = (TipoTransferenciaMultiActionController)
42            context.getBean("tipoTransferenciaMultiActionController");
43
44        sessionFactory = (SessionFactory) context.getBean("finixuSessionFactory");
45        Session session = sessionFactory.openSession();
46        TransactionSynchronizationManager.bindResource(sessionFactory, new SessionHolder(session));
47
48        controlHttpServlet= MockControl.createControl(HttpServletRequest.class);
49        mockhttpServletRequest=(HttpServletRequest) controlHttpServlet.getMock();
50
51        controlHttpResponse= MockControl.createControl(HttpServletResponse.class);
52        mockhttpServletResponse=(HttpServletResponse) controlHttpResponse.getMock();
53
54        controlHttpSession= MockControl.createControl(HttpSession.class);
55        mockHttpSession = (HttpSession) controlHttpSession.getMock();
56
57        super.setUp();
58    }
```

El método setUp () es donde se inicializan todos los atributos declarados. Para esconder y hacer más entendible el caso de prueba, se decidió introducir en el fichero transferencias-context.xml todo el proceso de inicializar el objeto tipoTransferenciaMultiActionController debido a su complejidad. Usando la librería EasyMock son creados los objetos mockHttpServletRequest, controlHttpServletRequest, mockHttpServletResponse y controlHttpServletResponse.

Unas de las pruebas que se realizaron fue al método persistirTipoTransferencia (HTTPServletRequest, requestHttpServletResponse response), primeramente se preparan los objetos mock para pasarle los datos correspondientes al método que se está probando. En la última línea del método se hace la llamada al método assertTrue() y se le pasa el método que se está probando del controlador.

[15] Es aquella que se encarga de decir al sistema, donde se encuentran todos los ficheros de mapeo de Hibernate, también va a ser la encargada de asociar los DAO dentro de las Fachadas.

Figura 25: Resultado de la ejecución de las pruebas



En la figura anterior se muestran las pruebas realizadas al caso de Uso Gestionar Tipo Transferencia, realizadas sobre la clase tipoTransferenciaMultiActionController, donde todas arrojan resultados satisfactorios. Se pueden observar los demás métodos en el Anexo 1.

4.3.2. Pruebas de Caja Negra

Las pruebas de caja negra, también denominadas pruebas de comportamiento, se centran en los requisitos funcionales de software. O sea, la prueba de caja negra permite al ingeniero de software obtener un conjunto de condiciones de entrada que ejerciten completamente los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las pruebas de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores de los que los métodos de caja blanca. (34)

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de iniciación y de determinación.

Dentro de las técnicas de prueba de caja negra, la que se utilizó para aplicarle al subsistema de Transferencia, fue la técnica de Partición Equivalente. Esta técnica permite obtener un conjunto de pruebas que reducen el número de casos de prueba que se deben realizar para evaluar correctamente el software, esto se debe a que se centra en la evaluación de clases de equivalencia que representan un conjunto de estados válidos o no válidos para condiciones de entradas existentes en el software.

Para definir las clases de equivalencia se tuvieron en cuenta un conjunto de reglas, entre las que se encuentran:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica la cantidad de valores, se identifica una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata de forma diferente a cada uno de ellos, se identifica una clase válida para cada uno de ellos y una clase inválida.

Luego de definir aquellas clases válidas e inválidas definidas, se diseñaron los casos de prueba para el caso de uso Registrar Tipo Transferencia. El mismo puede ser consultado en el Anexo 2.

4.3.3. Resultados de las pruebas realizadas.

Luego de haberse realizado las pruebas internas a la propuesta de solución donde se realizaron 3 iteraciones; en estas iteraciones se identificaron 6 errores en la primera, 2 en la segunda y ninguna en la tercera, de estos errores solo dos fueron significativos, y fueron en la aplicación, luego de ejecutarse la acción persistir Tipo Transferencia, la cual no persistía las monedas seleccionadas y el otro error significativo fue cuando contabilizaba no permitía ver los asientos contables. Desde el punto de vista tanto interno como funcional, atendiendo al correcto comportamiento que se presentó ante diferentes las diferentes situaciones, los módulos fueron probados funcionalmente. Se aplicaron los métodos de prueba de caja blanca y caja negra para validar el correcto funcionamiento del software implementado. Los resultados que arrojaron ambas pruebas contribuyeron a validar la calidad de la solución implementada.

4.4. Conclusiones del capítulo

En este capítulo se concluye que por la métrica TOC aplicada para validar el diseño del subsistema Transferencia, que el mismo permite un aumento en la reutilización de las clases, además, tiene una baja responsabilidad, no complica ni las pruebas ni la implementación

Se realizaron las pruebas de caja blanca y caja negra que permitieron corroborar de forma satisfactoria la implementación de los módulos pertenecientes al subsistema Transferencia de Quarxo. Desde el punto de vista tanto interno como funcional, atendiendo al correcto

comportamiento que se presentó ante diferentes las diferentes situaciones, los módulos fueron probados funcionalmente, los cuales cumplen con los requisitos funcionales definidos por el cliente

Conclusiones

Como resultado de la investigación realizada en el presente trabajo, se logró una mejor comprensión acerca de los procesos de negocios asociados a las transferencias bancarias, así como una valoración sobre algunos sistemas bancarios que trataban el tema referente a transacciones financieras, lo que se evidencia la necesidad de implementar el Subsistema Transferencias para el sistema de gestión bancaria Quarxo, ya que las funcionalidades que brindaban los sistemas informáticos analizados no se adaptaban a las estrategias de negocio del Banco Nacional de Cuba

El proceso de desarrollo de software fue guiado por la metodología RUP, cumpliendo con los elementos que dicha metodología propone y logrando una efectiva ingeniería de software. Las tecnologías seleccionadas para el desarrollo de la solución están acordes a los requerimientos del cliente y a las políticas del proyecto SAGEB.

Se modeló el sistema quedando definido actores, casos de uso así como las especificaciones de cada caso de uso. Posteriormente se identificaron y validaron los requerimientos funcionales para dicho subsistema por medio de los prototipos, además, se especificaron cuáles eran los estereotipos a utilizar en el modelo de clases del análisis y se modelaron sus principales clases. Los artefactos generados en el análisis posibilitaron a un mayor entendimiento respecto al negocio. Al finalizar se realizó una fundamentación de la arquitectura para su posterior diseño

El diseño de los módulos del subsistema Transferencia dotó al desarrollador de una vista lógica del sistema y los elementos que lo componen, lo que posibilitó la implementación del mismo, lo cual contribuyó a la obtención de un código flexible a futuros cambios

Se abordaron aspectos esenciales a tener en cuenta a la hora de implementar, como son los estándares de codificación, la iteración de los componentes en sus diagramas correspondientes, la descripción de sus clases y funcionalidades para facilitar tanto la lectura, comprensión y mantenimiento del código.

Para finalizar se realizaron las validaciones y las pruebas correspondiente, las cuales arrojaron a resultados satisfactorios que cumplen con los requerimientos funcionales que acordó el cliente, dando como resultado un producto con la calidad requerida para integrarse al sistema Quarxo

Bibliografía

1. **Sibanc.** Banco Central de Cuba. [En línea] 2009. [Citado el: 5 de 12 de 2011.] http://www.bc.gov.cu.

2. **Argentina, Banco Central de la República de.** Portal del Cliente Bancario. [En línea] 2009-2011. [Citado el: 14 de 02 de 2012.] http://www.clientebancario.gov.ar/default.asp.

3. **España, Banco de.** Banco de España Eurosistema. [En línea] 2011. [Citado el: 5 de 12 de 2011.] http://www.bde.es.

4. MONETOS. [En línea] 2012. [Citado el: 25 de 2 de 2012.] http://www.monetos.es/inversiones/cuentas-corrientes/transferencias-traspasos/.

5. **Sap.** Sap Argentina. [En línea] 2012. [Citado el: 18 de 02 de 2012.] http://www.sap.com/argentina/industries/banking/index.epx.

6. —. Sap Help. [En línea] [Citado el: 20 de 02 de 2012.] http://help.sap.com/saphelp_sbo2005asp1/helpdata/es_la/45/e556791c9b8744be4efc03fa67e054/content.htm.

7. **ABANFIN.** ABANFIN ASESORES BANCARIOS Y FINANCIEROS. [En línea] 2012. http://www.abanfin.com.

8. **Asociados, De Larrobla &.** Bantotal. [En línea] 2003. [Citado el: 7 de 12 de 2012.] http://www.bantotal.com.

9. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.Quinta edición.* Mexico DF. : McGraw Hill, 2005. pág. 171.

10. **Saavedra Darias, Mavi y Sánchez Imbert, José Antonio.** *Definición Los Requerimientos Funcionales Del Módulo Contabilidad Internacional Del Proyecto Banco Nacional.* Habana : s.n., 2008.

11. **Larman, Craig.** *Uml y Patrones.Introducción al análisis y diseñio orientado a objetos.* México : Pentice Hall, 1999.

12. **Laguna, Miguel A.** Departamento de Informática. Universidad de Valladolid. [En línea] Mayo de 2012. [Citado el: 14 de Marzo de 2012.] http://www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf.

13. **ESCALONA, M J y Koch, N.** Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo. [En línea] 2002. [Citado el: 8 de Febrero de 2012.] http://www.pst.ifi.lmu.de/~kochn/ideas03-escalona-koch.pdf.

14. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación S.A, 2000. 84-7829-036-2.

15. **Jacobson, Ivar.** *El proceso Unificado de Desarrollo De SoftWare.* Madrid : Addison Welsley, 2000.

16. **IvAr Jacobson, J Rumbaugh.** *El Lenguaje Unificado de Modelado Manuel de Referencia.* s.l. : Addison Wesley, 2001.

17. **RODRÍGUEZ, YOAN ANTONIO LÖPEZ.** *DEFINICIÓN DE LOS REQUERIMIENTOS FUNCIONALES DEL MÓDULO TESORERÍA, PRÉSTAMOS Y DEPÓSITOS DEL PROYECTO BANCO NACIONAL.* CUIDAD HABANA : s.n., 2008.

18. **Bizagi.** www.bizagi.com. [En línea] 27 de Febrero de 2012. [Citado el: 20 de Marzo de 2012.] http://wiki.bizagi.com/es/index.php?title=BPMN.

19. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. [Citado el: 12 de 05 de 2011.] http://www.proactiva-calidad.com/java/patrones/.

20. **Visconti, Marcello y Astudillo, Hernán.** Fundamentos de Ingeniería de Software. *Patrones de Diseño.* [En línea] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

21. **LARMAN, CRAIG.** *UML y Patrones Introducción al análisis y diseño orientado a objetos. Primera edición.* Mexico : Prentice Hall, 1999.

22. **Lara, Rafael Bello.** *Diseño e Implementación del Subsistema Cartas de Créditos Del Proyecto SAGEB.* Cuidad Habana : s.n., 2010.

23. **Marquina, Ernesto y Parra, José David.** *Guía de Patrones, Prácticas y Arquitectura .NET.* 2007.

24. http://scienti.colciencias.gov.co. [En línea] [Citado el: 21 de Enero de 2012.] http://scienti.colciencias.gov.co:8084/publindex/docs/articulos/1692-374X/2/10.pdf.

25. **Ciberaula.** Ciberaula. [En línea] [Citado el: 16 de Enero de 2012.] http://java.ciberaula.com/articulo/que_es_java.

26. **Sun.** Apache Tomcat. [En línea] 2011. [Citado el: 27 de 3 de 2011.] http://tomcat.apache.org/.

27. **Microsoft.** Microsoft SQL SERVER. [En línea] 2011. [Citado el: 27 de 3 de 2011.] http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx.

28. **Ramírez, Yoan Asdrubal Quintana.** *Diseño e Implementación de los módulos Chequera y Transferencias del sistema Quarxo.* 2011.

29. **Source, Spring.** Spring Source Community. [En línea] 2012. [Citado el: 10 de Enero de 2012.] http://www.springsource.org/spring-web-flow.

30. **COMUNITY, JBOSS.** HIBERNATE. [En línea] 2001. [Citado el: 27 de 3 de 2011.] http://www.hibernate.org/.

31. **Cala Rivero, Clara , Rodríguez Alcalde, Ángel L. y Barroso, José Ángel.** Digital.CSIC. [En línea] 2010. [Citado el: 13 de Marzo de 2012.] http://digital.csic.es/bitstream/10261/5110/1/Comunicacion_TCO-280-2007TY.pdf.

32. KDE DOCUMENTATION. [En línea] http://docs.kde.org/.

33. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación S.A, 2000. 84-7829-036-2.

34. **Pressman, Roger S.** *Ingenería de Software. Un enfoque práctico. Vol.I.* 1998.

35. **Prieto, Felix.** Departamento de Informática. Universidad de Valladolid. [En línea] 2009. [Citado el: 13 de 05 de 2011.] http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

36. **Martínez, Jesús.** Gestión MGD. *Gestión MGD.* [En línea] [Citado el: 7 de octubre de 2011.] http://ciberconta.unizar.es/leccion/gestionmgd/ .

Anexo 1: Implementación de algunos métodos probados con JUnit.

```
98      public void listarMoneda(HttpServletRequest request,
99              HttpServletResponse response) {
100         JSONObject json = new JSONObject();
101
102         List<String> monedas = tipoTransferenciaFacade.listarMonedas();
103
104         JSONArray array = new JSONArray();
105         for (String var : monedas) {
106
107             String[] aux = var.split("~");
108
109             JSONObject dato = new JSONObject();
110
111             dato.put("innerHTML", aux[1]);
112             dato.put("value", aux[0]);
113
114             array.add(dato);
115         }
116
117         json.put("identifier", "value");
118         json.put("label", "innerHTML");
119         json.put("value", "value");
120         json.put("items", array);
121
122         try {
123             ResponseUtil.escribirDatosEnElResponse(response, json);
124         } catch (IOException e) {
125             e.printStackTrace();
126         }
127     }
```

Anexo 2: Caso de prueba de caja negra del caso de uso Gestionar Tipo de Transferencia.

1. Descripción General.

- Se realiza el registro, consulta, eliminación y actualización de un Tipo de Transferencia.

2. Condiciones de Ejecución.

- El usuario debe estar autenticado con los permisos necesarios para realizar la acción.
- Debe existir conexión con la base de Datos.

3. Secciones a probar en el caso de uso Gestionar Tipo de Transferencia.

Nombre de la sección	Escenarios de la sección	Flujo normal de eventos
S 1: Gestionar Tipo Transferencia.	EC 1.1 Crear nuevo Tipo de Transferencia	- El usuario selecciona la opción “Crear Tipo de Transferencia”. - El sistema muestra la interfaz para un nuevo tipo de transferencia - El usuario introduce los datos del tipo de transferencia que va a crear y selecciona la opción “Aceptar”.
	EC 1.2 Cancelar Crear nuevo Tipo de Transferencia.	- El usuario selecciona la opción “Crear Tipo de Transferencia”. - El sistema muestra la interfaz para Crear Tipo de Transferencia - El usuario introduce los datos del tipo de transferencia que va a crear y selecciona la opción “Cancelar”.

Nombre de la sección	Escenarios de la sección	Flujo normal de eventos
		- Finaliza la operación Crear nuevo Tipo de Transferencia.
	EC 1.3 Crear nuevo Tipo de Transferencia con datos incorrectos.	- El usuario selecciona la opción “Crear nuevo Tipo de Transferencia”. - El sistema muestra la interfaz para Crear nuevo Tipo de Transferencia - El usuario introduce los datos del tipo de transferencia que va a crear y selecciona la opción “Aceptar”. - El sistema muestra un mensaje correspondiente al tipo de dato que se posea el error, por ejemplo: “Debe de introducir una Referencia Corriente”. - El usuario corrige el error y selecciona la opción “Aceptar”.
	EC 1.4 Actualizar Tipo de Transferencia	- El usuario selecciona el Tipo de Transferencia y escoge la opción “Actualizar Tipo de Transferencia”. - El sistema muestra la interfaz para actualizar el tipo de transferencia seleccionado. - El usuario introduce los datos con que va a actualizar el tipo de transferencia que seleccionó y selecciona la opción “Aceptar”.
	EC 1.5 Cancelar Actualizar Tipo de Transferencia	- El usuario selecciona el Tipo de Transferencia y escoge la opción “Actualizar Tipo de Transferencia”. - El sistema muestra la interfaz para actualizar el tipo de transferencia seleccionado.

Nombre de la sección	Escenarios de la sección	Flujo normal de eventos
		- El usuario introduce los datos con que va a actualizar el tipo de transferencia que seleccionó y selecciona la opción “Aceptar”. - El usuario introduce los datos con que va a actualizar el tipo de transferencia que seleccionó y selecciona la opción “Cancelar”. - Finaliza la operación Actualizar Regla semántica.
	EC 1.6 Actualizar Tipo de Transferencia con datos incorrectos.	- El usuario selecciona el Tipo de Transferencia y escoge la opción “Actualizar Tipo de Transferencia”. - El sistema muestra la interfaz para actualizar el tipo de transferencia seleccionado. - El usuario introduce los datos con que va a actualizar el tipo de transferencia que seleccionó y selecciona la opción “Aceptar”. - El sistema muestra un mensaje correspondiente al tipo de dato que se posea el error, por ejemplo: “Debe de introducir una Referencia Corriente”. - El usuario corrige el error y selecciona la opción “Aceptar”.
	EC 1.7 Consultar Tipo Transferencia.	- El usuario selecciona un Tipo de Transferencia y escoge la opción “Consultar Tipo de Transferencia”. - El sistema muestra la interfaz con los datos que posee el tipo de transferencia seleccionado.

Nombre de la sección	Escenarios de la sección	Flujo normal de eventos
	EC 1.8 Eliminar Tipo de Transferencia.	- El usuario selecciona un Tipo de Transferencia y escoge la opción “Eliminar Tipo de Transferencia”.

4. Descripción de variables.

No.	Nombre de campo	Clasificación	Valor Nulo	Descripción
[1]	Nombre	Campo de texto	No	El usuario introduce el nombre correspondiente al tipo de transferencia.
[2]	Referencia Corriente	Campo de texto	No	El usuario introduce la referencia corriente que tendrá el tipo de transferencia.
[3]	Operación mensajería	Lista desplegable	Si	Se selecciona el tipo de mensaje asociado al tipo de transferencia.
[4]	Moneda Origen	Combo	No	Se carga de la bases de datos las monedas.
[5]	CueSubcue Origen	Combo	No	Se carga de la bases de datos los cuesubcue.
[6]	CodContra Origen	Combo	Si	Se carga de la bases de datos los cod contras.
[7]	Moneda Destino	Combo	No	Se carga de la bases de datos las monedas.

[8]	CueSubcue Destino	Combo	No	Se carga de la bases de datos los cuesubcue.
[9]	CodContra Destino	Combo	Si	Se carga de la bases de datos los codcontras.
[10]	Comisiones	Combo	Si	Se carga de la bases de datos las comisiones existentes.

4. Matriz de datos.

Escenario	Nombre	Referencia Corriente	Operación mensajería	Moneda Origen	CueSubcue Origen	CodContra Origen	Moneda Destino	Cue Subcue Destino	Cod Contra Destino	Comisiones	Respuesta del Sistema	Resultado de la Prueba
EC 1.1 Crear nuevo Tipo de Transferencia con éxito.	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	Banca Banco	Se crea con éxito el nuevo Tipo de transferencia.	Satisfactorio
EC 1.2 Cancelar Crear nuevo Tipo de Transferencia.	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	Banca Banco	El sistema cancela la operación.	Satisfactorio
EC 1.3 Crear nuevo Tipo de Transferencia con datos incorrectos.		RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	Banca Banco	El nombre no puede ser nulo.	Satisfactorio
	Transferencia Enviada	12	Transferencia Enviada.	USD	1411	3212	USD	1210	0138000	Banca Banco	La Referencia Corriente no puede ser	Satisfactorio

	para BICSA										número.	
	Transferencia Enviada para BICSA	RE	VA	USD	1411	3212	USD	1210	0138000	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada		1411	3212	USD	1210	0138000	Banco a Banco	La moneda origen no puede ser nula.	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD		3212	USD	1210	0138000	Banco a Banco	El cuesubcue origen no puede ser nula no puede	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	VA	USD	1210	0138000	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212		1210	0138000	Banco a Banco	La moneda destino no puede ser nula.	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD		0138000	Banco a Banco	El cuesubcue destino no puede ser nula no puede	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	VA	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación.	Satisfactorio
	Transferencia Enviada	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	VA	El sistema muestra un mensaje	Satisfactorio

	para BICSA										notificando el éxito de la operación.	
EC 1.4 Actualizar Tipo de Transferencia	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1421	3210	USD	1210	0138000	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación	Satisfactorio
EC 1.5Cancelar Actualizar Tipo de Transferencia	Transferencia Enviada para BICSA	NA	NA	NA	NA	NA	NA	NA	NA	NA	El sistema cancela la operación.	Satisfactorio
EC 1.6 Actualizar Tipo de Transferencia con datos incorrectos.		RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	Banco a Banco	El nombre no puede ser nulo.	Satisfactorio
	Transferencia Enviada para BICSA	12	Transferencia Enviada.	USD	1411	3212	USD	1210	0138000	Banco a Banco	La Referencia Corriente no puede ser número.	Satisfactorio
	Transferencia Enviada para BICSA	RE	NA	USD	1411	3212	USD	1210	0138000	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada		1411	3212	USD	1210	0138000	Banco a Banco	La moneda origen no puede ser nula.	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD		3212	USD	1210	0138000	Banco a Banco	El cuesubcue origen no puede ser nula no puede	Satisfactorio
	Transferencia Enviada para	RE	Transferencia Enviada	USD	1411	NA	USD	1210	0138000	Banco a Banco	El sistema muestra un mensaje notificando el éxito	Satisfactorio

	BICSA										de la operación	
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212		1210	0138000	Banco a Banco	La moneda destino no puede ser nula.	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD		0138000	Banco a Banco	El cuesubcue destino no puede ser nula no puede	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	NA	Banco a Banco	El sistema muestra un mensaje notificando el éxito de la operación.	Satisfactorio
	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	NA	El sistema muestra un mensaje notificando el éxito de la operación.	Satisfactorio
EC 1.7 Consultar Tipo Transferencia	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	NA	El sistema muestra los datos asociados con el tipo de transferencia seleccionado.	Satisfactorio
EC 1.8 Eliminar Tipo de Transferencia.	Transferencia Enviada para BICSA	RE	Transferencia Enviada	USD	1411	3212	USD	1210	0138000	NA	El sistema elimina el tipo de transferencia seleccionado y muestra un mensaje notificando el éxitode la operación.	