

1.1 INTRODUCCIÓN

(Cabrera, 2012) Hoy en día la presencia de sistemas tanto institucional como personal es cada vez más usual e incluso para la gran mayoría de empresas y entidades que suministran productos y servicios es una necesidad de primer nivel, ya sea para evitar ser desplazados por su competencia, o con el ánimo de ofrecer mejores y oportunos servicios a sus clientes.

El crecimiento antes mencionado sumado al control manual resulta una dificultad para manejar información clasificada, actualizada y ordenada de las labores que realizan el personal de la empresa Edifica. Es por esto que se necesita sistematizar o automatizar el proceso de Control y seguimiento de las importaciones que actualmente se lo realiza manualmente.

Los sistemas de información son herramientas de tecnología que concebidas eficientemente, contribuyen de manera efectiva en el desempeño de las actividades laborales que desarrollan diariamente los trabajadores a todo nivel, consecuentemente es muy importante contar con herramientas automatizadas en favor de los empleados que desempeñen sus múltiples funciones con un alto grado de responsabilidad y en menor tiempo posible de una manera efectiva y eficiente.

1.2 ANTECEDENTES

Edifica S.R.L nace el año 2005 como una empresa importadora de material para la construcción que con una larga tradición, hoy día consolidado como una empresa líder en el mercado nacional por importar material de alta calidad.

Hoy, esta empresa emprende varios proyectos en materia de la construcción diversificándose con varias empresas para cubrir, el mercado de la construcción para el mercado nacional.

Esta empresa tiene un amplio sistema de relaciones internas y externas para la comercialización de materiales para el rubro de la construcción, ferretería, cerrajería e industria.

Misión

Es misión de **Edifica S.R.L** proveer productos y servicios para la construcción con altos niveles de calidad y seguridad coadyuvando al desarrollo de la región, altamente comprometidos con el rubro de la construcción.

Visión

Llegar a ser la empresa referente y reconocida a nivel regional y nacional por la importación de productos y servicios para la construcción.

Objetivo General

Ser una empresa líder en el mercado, proveyendo productos y servicios de calidad para la construcción que permitan a los clientes optimizar sus requerimientos, facilitando la comunicación, colaboración y coordinación con el cliente.

Objetivo Específicos

- Lograr productos de calidad inmejorables para ofrecer en el mercado.
- Satisfacer las necesidades de nuestros clientes en lo práctico y en lo estético, creando e innovando constantemente para estar a la altura de las nuevas tendencias.
- Asociarnos a los mejores proveedores del mundo para darle a nuestros productos un valor agregado que se traduzca en beneficios para nuestros clientes y personal de la empresa.
- Brindar a nuestros empleados la oportunidad de crecer junto a la empresa otorgando cursos de capacitación.
- Aumentar la productividad de la producción de Tanques de Agua para abastecer al mercado.

Actividad Comercial: Venta Al Por Mayor De Materiales De Construcción, Artículos De Ferretería, Equipo Y Materiales De Fontanería Y Calefacción. Ver Anexo N°1

1.3 PLANTEAMIENTO DEL PROBLEMA

La empresa EDIFICA S.R.L maneja actualmente de forma manual en libros de Excel toda la información referente a las importaciones y demás datos referentes a las importaciones de productos.

Uno de los problemas que actualmente se presenta en la Empresa Edifica S.R.L es la lentitud de acceso a la información actualizada rápida precisa y ordenada de las diferentes etapas en las cuales se pueden encontrar los documentos referentes a una importación de productos.

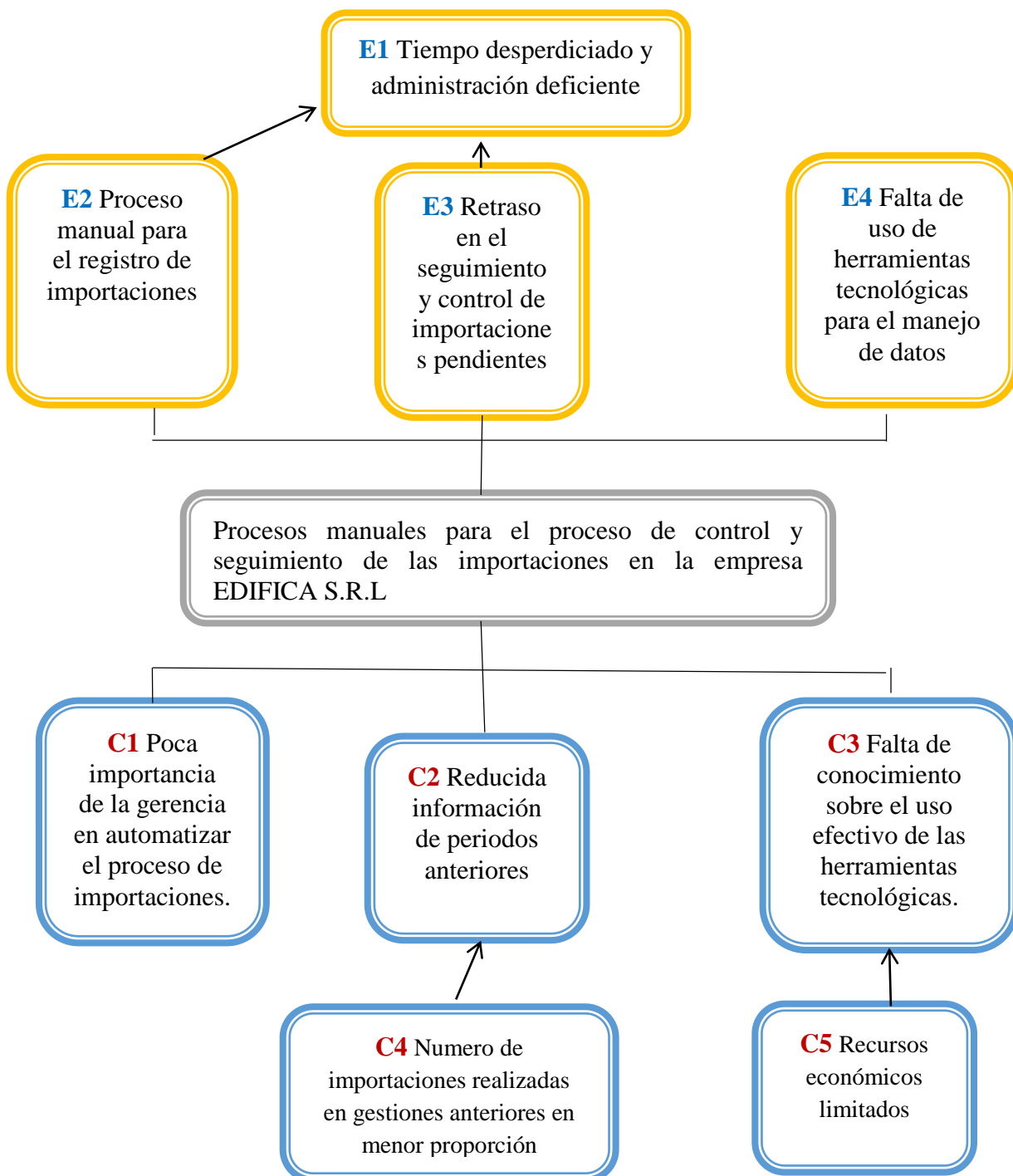
En consecuencia a lo mencionado el problema se centra en la limitación de la información rápida ordenada y consecuente al momento de ser requerida para poder realizar el control y seguimiento de las importaciones que permita al empleado y a la gerencia contar en tiempo real con la información de su proceso de importación y determinar la cantidad de procesos pendientes que se encuentran en las oficinas de la Aduana o en tramitadores con son las agencias despachantes de aduana.

1.4 Formulación del Problema

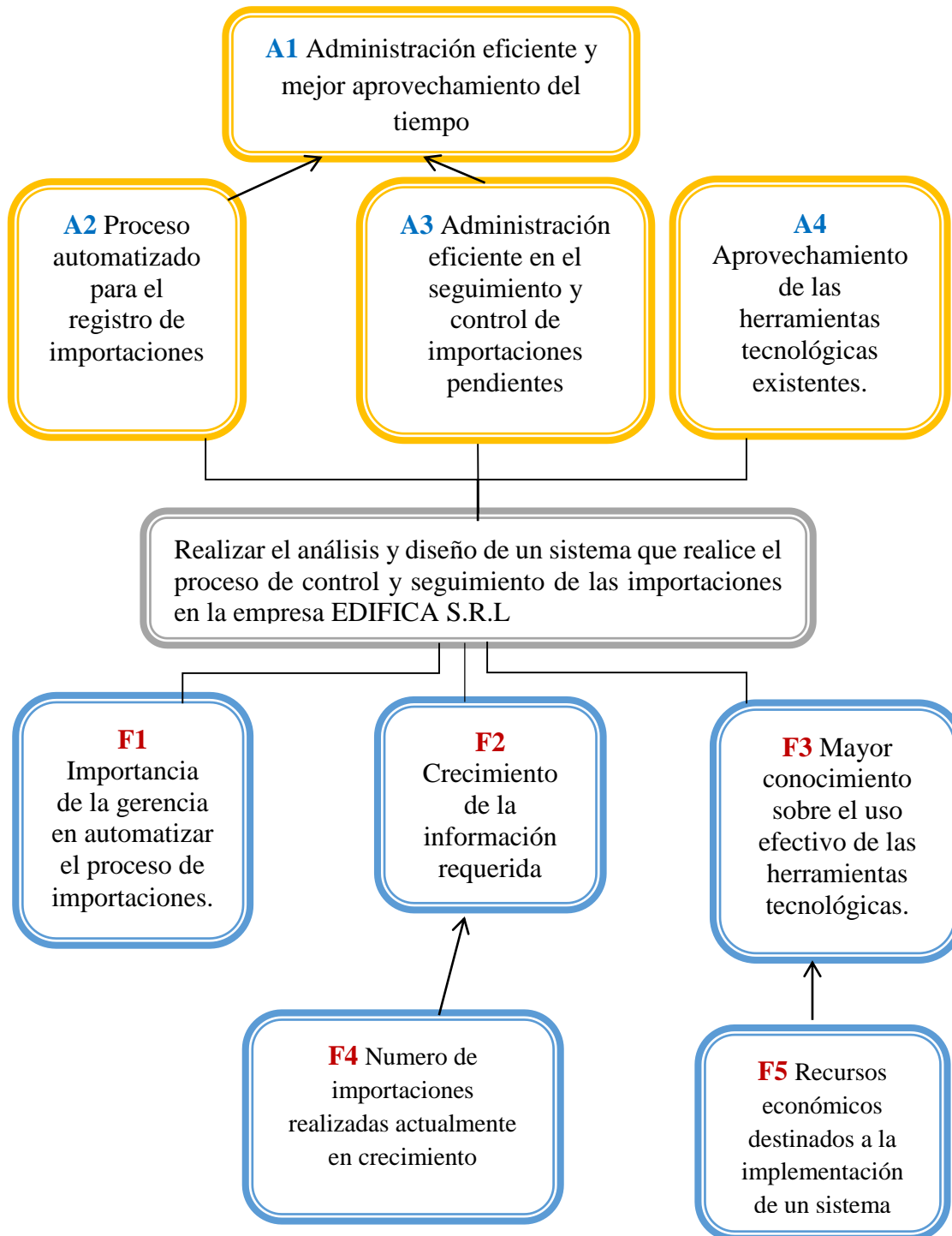
¿Cómo automatizar el proceso de gestión de la información de las importaciones en la empresa EDIFICA S.R.L?

1.5 SISTEMATIZACIÓN DEL PROBLEMA Y ABORDAJE DE LA SOLUCIÓN

ÁRBOL DE PROBLEMAS



ÁRBOL DE OBJETIVOS



1.6 OBJETIVOS

1.6.1 Objetivo General

Desarrollar el análisis y diseño de un sistema para la de gestión de la información, realizando un control y seguimiento de importaciones eficiente, que permita mejorar la disponibilidad de datos precisos y confiables a los empleados y gerencia de la empresa EDIFICA S.R.L.

1.6.2 Objetivos Específicos

- Analizar la información actual sobre el manejo de las importaciones mediante el cual se está trabajando dentro de la empresa EDIFICA SRL.
- Aplicar la metodología RUP realizando la utilización del lenguaje UML para la posterior documentación del sistema diseñado.
- Identificar los diferentes módulos que comprenden la gestión de la información.
- Diseño de una interfaz amigable para mayor adaptabilidad en la administración de la información en el sistema.
- Diseñar módulos para el proceso de registro, control y seguimiento de las importaciones.
- Mejorar el proceso de seguimiento de información mediante la elaboración de informes y reportes que ayuden en la toma de decisiones.

1.7 JUSTIFICACIÓN.

1.7.1 Justificación Científica

El presente trabajo aplica la ingeniería de Software para resolver las falencias encontradas en la gestión de la información en el Empresa Edifica S.R.L. Desarrollando el sistema bajo estas tecnologías se apoyara con la proporción de la información de manera automática, eficiente y oportuna para la toma de decisiones para la gerencia.

1.7.2 Justificación Social

La Empresa Edifica S.R.L en el departamento de Tarija provincia Cercado brinda sus servicios a un amplio rango de familias de diferentes zonas de la ciudad, juega un papel muy importante en cuando proporcionar un servicio a los clientes vinculado con proveer productos de calidad al mercado, por esa razón la automatización de gestión de la información de las importaciones, logrará que se impulse el manejo de la información con mayor eficiencia, significa introducir tecnología a esta área para así mejorar los proceso actuales, lo cual resulta de suma importancia para esta empresa, y más aún para la sociedad, debido a que esta almacena información valiosa y necesaria para una correcta atención.

1.7.3 Justificación Económica

La Empresa Edifica S.R.L presentará mayor ingreso resultado de un mejor aprovechamiento del tiempo para la prestación de servicios a clientes del Departamento de Tarija, y de esta manera el personal que realice el registro y control de su información lo realizara de manera eficiente y rápida, además que tendrá mayor tiempo para realizar sus tareas adicionales .

1.7.4 **Justificación Técnica**

Hoy en día la necesidad de información automatizada es un recurso imprescindible para toda empresa con el objeto de ofrecer la posibilidad de utilizar una información precisa y oportuna para poder controlar los datos registrados y así optimizar un buen funcionamiento y un alto grado en la toma de decisiones.

El desarrollo de este sistema es justificado tecnológicamente porque se cuenta con todos los medios tecnológicos necesarios para su implementación dentro de la Empresa.

Además que los operadores no necesitan tener conocimientos avanzados de informática eso hace que el manejo sea sencillo y gestionable.

1.8 ALCANCES Y LIMITACIONES.

1.8.1 Alcances

- Ingresar su nombre de usuario (login) y contraseña (password) para poder ingresar al sistema previa verificación de dichos datos.
- Cerrar sesión para mantener la seguridad en el sistema.
- Registro de Documentos de las facturas de emitidas por el Proveedor.
- Registro de Pagos realizados al Proveedor.
- Registro de pagos a la naviera para des consolidación naviera. (si corresponde).
- Registro de pagos a Administración de Servicios Portuarios Bolivia ASP-B
- Registro de Pagos de tributos aduaneros.
- Registro de los facturas de transporte terrestre nacional e internacional.
- Recepción de los documentos póliza de importación

1.8.2 Limitaciones

- Estará dedicada estrictamente a información sobre las importaciones de la empresa EDIFICA S.R.L
- No se realizará la puesta en marcha del sistema en el entorno previsto de uso, pero se llevará a cabo las pruebas para corroborar que el software funciona correctamente.
- No se realizará la contabilidad del sistema ya que se lo tomará como un módulo independiente, pudiendo ser propuesto como otro trabajo de grado por la complejidad del mismo.
- El sistema solo podrá ser manejado por personal autorizado y el cual tenga asignado un rol respectivo en la empresa.

2. MARCO TEÓRICO CONCEPTUAL Y/O REFERENCIAL

2.1. RÉGIMEN DE IMPORTACIONES (BOLIVIA, 2010)

Las importaciones bolivianas se rigen por lo establecido en la política arancelaria que define el nivel arancelario que se aplica a las importaciones de bienes. La norma está en concordancia a las disposiciones de la normativa internacional, como son el Código de Valoración del Acuerdo General sobre Arancel Aduanero y Comercio (GATT), a la legislación nacional y a la nueva nomenclatura denominada "Sistema Armonizado de Designación y Codificación de Mercancías" NANDINA, que es la nomenclatura oficial de los Países Miembros del Acuerdo de Cartagena, proceso subregional de integración (Comunidad Andina), así como la NALADISA que es aplicada en el marco de la ALADI.

El arancel aduanero de importación de Bolivia, a partir de noviembre el 2007, cuenta con una nueva estructura arancelaria, con alícuotas de cero (0%), cinco (5%), diez (10%), quince (15%), veinte (20%) y treinta cinco (35%).

Los productos originarios de los Países Miembros de la Comunidad Andina, de los países miembros de la ALADI, con los que Bolivia tiene Acuerdos de Complementación Económica, cuentan con preferencias arancelarias de hasta el 100%.

ARANCELES, IMPUESTOS Y SERVICIOS APLICADOS A LA IMPORTACIÓN EN BOLIVIA

ARANCELES/ IMPUESTOS	TASAS	BASE IMPONIBLE	OBSERVACIONES
Arancel Importación	de 0 %, 5%, 10%, 15%, 20% y 35%.	CIF Frontera	– Algunos bienes de capital (maquinarias y equipos) tienen una rebaja del 50% del arancel de importación.

Tasa de Almacén aduanero	0.5 %	CIF Frontera	– Valor según el servicio prestado y tiempo de permanencia. La tasa es un valor referencial.
Aporte Gremial	0.3 %	CIF Frontera	– Aporte gremial según la Cámara a la que se esta asociado.
Despacho Aduanero	0.1 % al 2,5%	CIF Frontera	– Comisión variable que se paga a la agencia despachante de aduana.
Impuesto al Valor Agregado (IVA).	14,94 %	CIF – Aduana	IVA importaciones, grava sobre el valor de mercadería más el arancel.
Impuesto al Consumo Específico (ICE).	---	CIF – Aduana	Impuesto variable, grava a licores, tabaco y bienes suntuarios.
Impuesto a los Hidrocarburos IEDH).	---	CIF – Aduana	Según el producto a ser importado.

Fuente: Elaborado en base a información Ministerio de Economía y Finanzas Publicas y Aduana Nacional de Bolivia.

El Régimen de Importaciones establece prohibiciones de importación de bienes que afectan a la salud y vida humana, animal o contra la preservación vegetal, a la moral, al medio ambiente, la seguridad del Estado y el sistema financiero de la Nación y otras autorizaciones previas, expresadas en la Ley General de Aduanas, tales como sustancias controladas, (químicos, precursores y otros), dichas importaciones requieren autorización de la entidad encargada del control de tráfico de sustancias peligrosas.

PRINCIPALES DOCUMENTOS PARA LA IMPORTACIÓN

- ✓ Factura comercial o documento equivalente.
- ✓ Documento de transporte, (guía aérea, conocimiento marítimos, carta de porte).
- ✓ Póliza de seguro o certificado de seguro (cuando corresponda).
- ✓ Parte de recepción.

- ✓ Certificado de inspección previa sin discrepancia o declaración jurada del valor en aduanas (esta última suscrita por el importador).
- ✓ Planilla de gastos portuarios (cuando corresponda).
- ✓ Factura de transporte (cuando corresponda).
- ✓ Lista de empaque.
- ✓ Certificado de origen si viene de países con los que Bolivia cuenta con convenios comerciales de preferencia arancelaria.
- ✓ Certificado bromatológico (si la mercadería contiene tóxicos), zoo sanitario (para animales), fitosanitario (para vegetales), zoosanitario (para peces), etc., (cuando corresponda).
- ✓ Certificados y/o autorizaciones previas (cuando corresponda).

2.2 Ingeniería de Software

Según Sommerville, I. (2005):

Dentro de la Ingeniería se cuenta con una disciplina que abarca todos los aspectos para la producción de software, pues la Ingeniería de Software comprende dichos aspectos, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza. Presenta dos fases:

- ❖ **Disciplina de la ingeniería.** Los ingenieros hacen que las cosas funcionen. Aplican Teorías, métodos y herramientas donde sean convenientes, pero las utilizan de forma selectiva y siempre tratando de descubrir soluciones a los problemas, aun cuando no existan teorías y métodos aplicables para resolverlos. Los ingenieros también saben que deben trabajar con restricciones financieras y organizacionales, por lo que buscan soluciones tomando en cuenta estas restricciones.
- ❖ **Todos los aspectos de producción de software.** La ingeniería del software no solo comprende los procesos técnicos del desarrollo de software, sino también con actividades

tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software.

En general, los ingenieros de software adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más eficiente de producir software de alta calidad. Sin embargo, aunque la ingeniería consiste en seleccionar el método más apropiado para un conjunto de circunstancias, un enfoque más informal y creativo de desarrollo podría ser efectivo en algunas circunstancias. El desarrollo informal es apropiado para el desarrollo de sistemas basado en Web, los cuales requieren una mezcla de técnicas de software y de diseño gráfico.

2.3 Sistema de Información

Sistema de Información. (s.f.). En Wikipedia. Recuperado el 20 de Mayo de 2015 de http://es.wikipedia.org/wiki/Sistema_de_Información

Un sistema de información es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad o un objetivo. Dichos elementos formarán parte de alguna de las siguientes categorías:

Personas; Datos; Actividades o técnicas de trabajo; Recursos materiales en general (generalmente recursos informáticos y de comunicación, aunque no necesariamente).

Todos estos elementos interactúan para procesar los datos (incluidos los procesos manuales y automáticos) y dan lugar a información más elaborada, que se distribuye de la manera más adecuada posible en una determinada organización, en función de sus objetivos

2.3.1 Ciclo de Vida de Los Sistemas de Información

- ❖ **Conocimiento de la Organización.** Analizar y conocer todos los sistemas que forman parte de la organización, así como los futuros usuarios del SI. En las empresas (fin de lucro presente), se analiza el proceso de negocio y los procesos transaccionales a los que dará soporte el SI.

- ❖ **Identificación de problemas y oportunidades.** El segundo paso es relevar las situaciones que tiene la organización y de las cuales se puede sacar una ventaja competitiva (Por ejemplo: una empresa con un personal capacitado en manejo informático reduce el costo de capacitación de los usuarios), así como las situaciones desventajosas o limitaciones que hay que sortear o que tomar en cuenta (Por ejemplo: el edificio de una empresa que cuenta con un espacio muy reducido y no permitirá instalar más de dos computadoras).
- ❖ **Determinar las necesidades.** Este proceso también se denomina elicitación de requerimientos. En el mismo, se procede identificar a través de algún método de recolección de información (el que más se ajuste a cada caso) la información relevante para el SI que se propondrá.
- ❖ **Diagnóstico.** En este paso se elabora un informe resaltando los aspectos positivos y negativos de la organización. Este informe formará parte de la propuesta del SI y, también, será tomado en cuenta a la hora del diseño.
- ❖ **Propuesta.** Contando ya con toda la información necesaria acerca de la organización, es posible elaborar una propuesta formal dirigida hacia la organización donde se detalle: el presupuesto, la relación costo-beneficio y la presentación del proyecto de desarrollo del SI.
- ❖ **Diseño del sistema.** Una vez aprobado el proyecto, se comienza con la elaboración del diseño lógico del SI; la misma incluye: el diseño del flujo de la información dentro del sistema, los procesos que se realizarán dentro del sistema, el diccionario de datos, los reportes de salida, etc. En este paso es importante seleccionar la plataforma donde se apoyará el SI y el lenguaje de programación a utilizar.
- ❖ **Codificación.** Con el algoritmo ya diseñado, se procede a su reescritura en un lenguaje de programación establecido (programación) en la etapa anterior, es decir, en códigos que la máquina pueda interpretar y ejecutar.
- ❖ **Implementación.** Este paso consta de todas las actividades requeridas para la instalación de los equipos informáticos, redes y la instalación de la aplicación (programa) generada en la etapa de Codificación.

- ❖ **Mantenimiento.** Proceso de retroalimentación, a través del cual se puede solicitar la corrección, el mejoramiento o la adaptación del SI ya creado a otro entorno de trabajo o plataforma. Este paso incluye el soporte técnico acordado anteriormente.

2.4 Lenguaje de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras.

“Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.” Lutz, M (2010). Learning Python.

Está formado por un **conjunto de símbolos y reglas sintácticas y semánticas** que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa informático se le llama programación.

También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos, a través de los siguientes pasos:

El desarrollo lógico del programa para resolver un problema en particular.

Escritura de la lógica del programa empleando un lenguaje de programación específico (codificación del programa).

Ensamblaje o compilación del programa hasta convertirlo en lenguaje de máquina.

Prueba y depuración del programa.

Desarrollo de la documentación.

2.4.1 Programación Orientada a Objetos

Lenguajes de Programación (2009). Recuperado el 21 de Mayo de: <http://www.lenguajes-de-programacion.com/programacion-orientada-a-objetos.shtml>

La programación orientada a objetos, intenta simular el mundo real a través del significado de objetos que contiene características y funciones. Los lenguajes orientados a objetos se clasifican como lenguajes de quinta generación.

Como su mismo nombre indica, la programación orientada a objetos se basa en la idea de un objeto, que es una combinación de variables locales y procedimientos llamados métodos que juntos conforman una entidad de programación.

2.4.2 C#

Microsoft Developer Network (2015). MSDN Library. Microsoft. Recuperado el 21 de Mayo de: <https://msdn.microsoft.com/es-es/library/z1zx9t92.aspx>

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores compilar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Puede utilizar C# para crear aplicaciones cliente de Windows, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y mucho, mucho más. Visual C# proporciona un editor de código avanzado, cómodos diseñadores de interfaz de usuario, depurador integrado y numerosas herramientas más para facilitar el desarrollo de aplicaciones basadas el lenguaje C# y .NET Framework.

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Todas las variables y métodos, incluido el método Main que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase. Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces. Los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra

clave override como medio para evitar redefiniciones accidentales. En C#, una struct es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- ❖ Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.
- ❖ Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- ❖ Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- ❖ Comentarios en línea de documentación XML.
- ❖ Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

2.5 Microsoft Visual Studio Express

Microsoft Visual Studio Express. (s.f.). En Wikipedia. Recuperado el 20 de Mayo de 2015 de http://es.wikipedia.org/wiki/Microsoft_Visual_Studio_Express

Microsoft Visual Studio Express Edition es un programa de desarrollo en entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Es de carácter gratuito y es proporcionado por la compañía Microsoft Corporation orientándose a principiantes, estudiantes y aficionados de la programación web y de aplicaciones, ofreciéndose dicha aplicación a partir de la versión 2005 de Microsoft Visual Studio.

2.6 Microsoft SQL Server Express

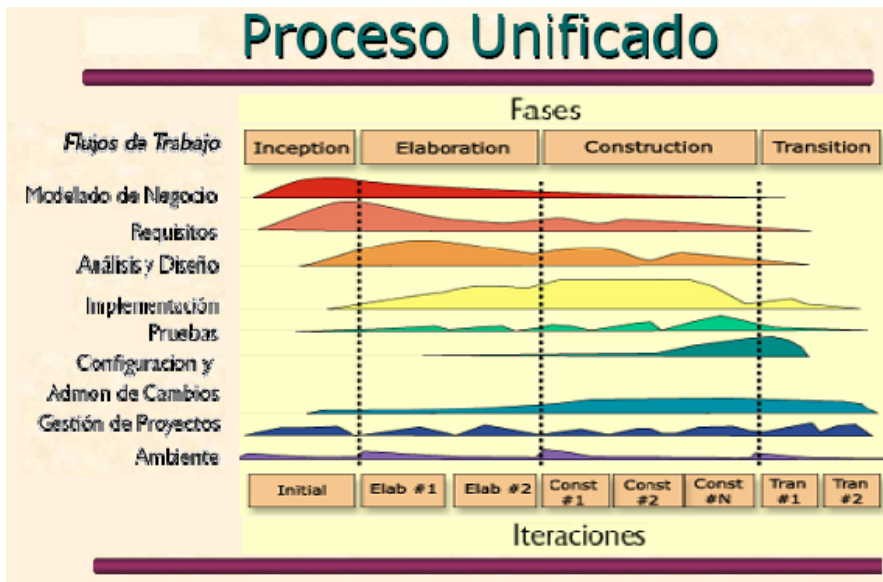
Microsoft Download Center (2015). Microsoft. Recuperado el 21 de Mayo de: <https://www.microsoft.com/es-es/download/details.aspx?id=29062>

Microsoft® SQL Server® 2012 Express es un sistema de administración de datos gratuito, eficaz y confiable que ofrece un almacén de datos completo y confiable para sitios web ligeros y aplicaciones de escritorio.

2.7 Metodología RUP

(Guerrero) La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process (Proceso Unificado Racional), es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada por el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP se caracteriza por ser interactivo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).



2.2.1

2.2.2

Fases y Ciclos

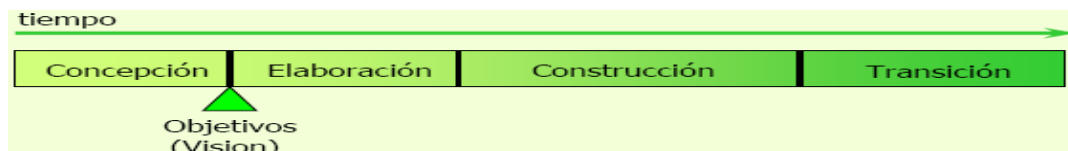
El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante.

FASE 1 Inicio

En el que se establece la oportunidad y alcance del proyecto. Se identifican todas las actividades externas con las que se trata (actores) y se define la interacción a un alto nivel de abstracción (identificar todos los casos de uso).

Producto Un documento de visión general: Requerimientos generales del proyecto, características principales, restricciones, modelo inicial de casos de uso, glosario.

Hito Las partes interesadas deben acordar el alcance y estimación de tiempo y costo, comprensión de los requerimientos plasmados en casos de uso.

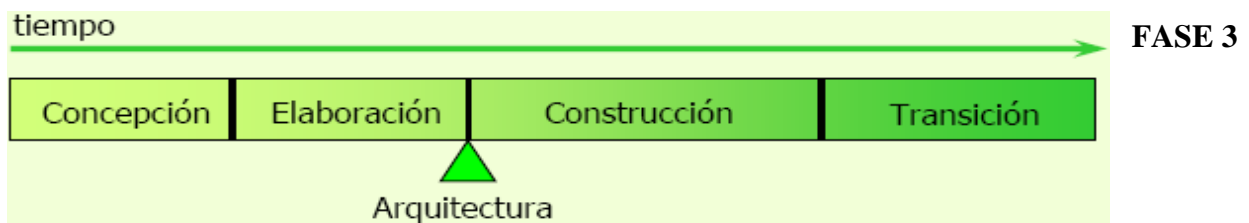


FASE 2 Elaboración

Se analiza el dominio del problema, se establece una arquitectura base sólida, se desarrolla un plan de proyecto y se eliminan los elementos de mayor riesgo para el desarrollo exitoso del proyecto.

Producto La elaboración es la parte más crítica del proceso, al final toda la ingeniería “dura” está hecha, se puede decidir si vale la pena seguir adelante, a partir de aquí la arquitectura, los requerimientos y los planes de desarrollo son estables. Ya son menos riesgosos y se puede planificar el resto del proyecto con menor incertidumbre. Se construye una arquitectura ejecutable que contemple los casos de uso críticos y los riesgos identificados, modelos de casos de uso con descripciones detalladas, un prototipo ejecutable de la arquitectura, plan de desarrollo para el resto del proyecto, un manual de usuario preliminar, descripción de la arquitectura del software (Diagrama de clases, secuencias y actividades)

Hito Condiciones de éxito de la elaboración: se establece la visión del producto, arquitectura, las pruebas de ejecución demuestran que los riesgos han sido abordados y resueltos, es el plan del proyecto algo realista, están de acuerdo con el plan todas las personas involucradas.

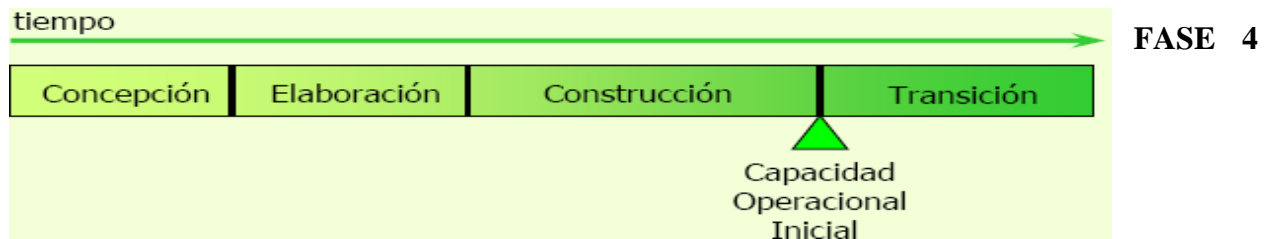


Construcción

En esta fase todas las componentes restantes se desarrollan e incorporan al producto, todo es probado en profundidad, el énfasis está en la producción eficiente y no ya en la creación intelectual, puede hacerse construcción en paralelo, pero esto exige una planificación detallada y una arquitectura muy estable.

Producto El producto de software integrado y corriendo en la plataforma adecuada y con manuales de usuario.

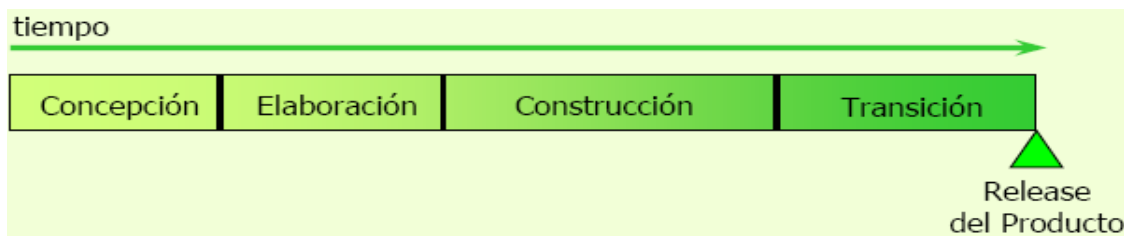
Hito Se obtiene un producto Beta que debe decidirse si puede ponerse en ejecución sin mayores riesgos. Condiciones de éxito: el producto está maduro y estable para instalarlo en el ambiente del cliente, están los interesados listos para recibirlo.



Transición

El objetivo es traspasar el software desarrollado, una vez instalado surgirán nuevos elementos que impliquen nuevos ciclos de desarrollo, incluye pruebas

Beta para validar el producto con las expectativas del cliente, ejecución paralela con sistemas antiguos, conversión de datos, entrenamiento de usuarios, distribución del producto.



Actividades

Y el flujo de trabajo (workflow) entre ellas en base a los llamados diagramas de actividades. El proceso define una serie de roles que se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado (artefactos) que se espera de ellos.

RUP define nueve actividades a realizar en cada fase del proyecto:

1. Modelado del negocio.
2. Análisis de requisitos.
3. Análisis y diseño.
4. Implementación.
5. Test.

6. Distribución.
7. Gestión y configuración de cambios.
8. Gestión del proyecto.
9. Gestión del entorno.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

2.8 Modelo Orientado a Objetos

La técnica que será utilizada es UML (Lenguaje de Modelado Unificado).

2.8.1 Lenguaje de Modelado Unificado (UML)

El Lenguaje de Modelamiento Unificado (UML – Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar y documentar los elementos creados mediante un lenguaje común describiendo modelos.

UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de la base datos y componentes de software reutilizables.

UML puede ser utilizado por cualquier metodología de análisis y diseño orientada por objetos para expresar los diseños.

2.8.1.1 Diagrama de Casos de Usos

Un Diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuario u otras aplicaciones).

Es una herramienta esencial para la captura de requerimientos y para la planificación y control de un proyecto interactivo.

Los Casos de Uso se representa en el diagrama por una elipse que denota un requerimiento solucionando por el sistema. Cada caso de uso es una operación completa desarrollada por los actores y por el sistema en un diálogo. El conjunto de casos de uso representa la totalidad de operaciones desarrolladas por el sistema.

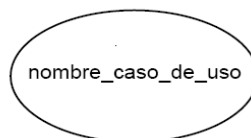
Elementos de los Casos de Uso

- **Actor:** Una definición previa, es que un Actor es un rol que un usuario juega con respecto al sistema. Es importante destacar el uso de la palabra rol, pues con esto se especifica que un Actor no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

Un solo actor puede actuar en muchos casos de uso; recíprocamente, un caso de uso puede tener varios actores. Los actores no necesitan ser humanos pueden ser sistemas externos que necesitan alguna información del sistema actual Ver Anexo No.1.



- **Caso de Uso:** Es una operación / tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso Ver Anexo No.1.



- **Relaciones:**

- **Asociación** Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.
 - **Dependencia o Instanciación** Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada Ver Anexo No.1.
 - **Generalización** Este tipo de relación es uno de los más utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de **Uso** (<<uses>>) o de **Herencia** (<<extends>>). Este tipo de relación está orientado exclusivamente para casos de uso (y no para actores).

Extends: Se recomienda utilizar cuando un caso de uso es similar a otro (características).

Uses: Se recomienda utilizar cuando se tiene un conjunto de características que son similares en más de un caso de uso y no se desea mantener copiada la descripción de la característica.

De lo anterior cabe mencionar que tiene el mismo paradigma en diseño y modelamiento de clases, en donde está la duda clásica de **usar** o **heredar**.

2.8.1.2 Diagrama de Clases

Un diagrama de clases o estructura estática muestra el conjunto de clases y objeto importante que forman parte de un sistema, junto con las relaciones existentes entre clases y objetos. Muestra de una manera estática la estructura de información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con los demás en el modelo.

Elementos del Diagrama de Clases

En UML la clase se encuentra representada por un rectángulo con tres divisiones internas, son los elementos fundamentales del diagrama. Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Elementos

- **Clase** Es la unidad básica que encapsula toda la información de un objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.). En UML, una clase es representada por un rectángulo que posee tres divisiones:
- **Atributos y Métodos:**
 - **Atributos:** Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno.
 - **Métodos:** Los métodos u operaciones de una clase son la forma en cómo ésta interactúa con su entorno, éstos pueden tener características.
- **Relaciones entre Clases:** En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación.

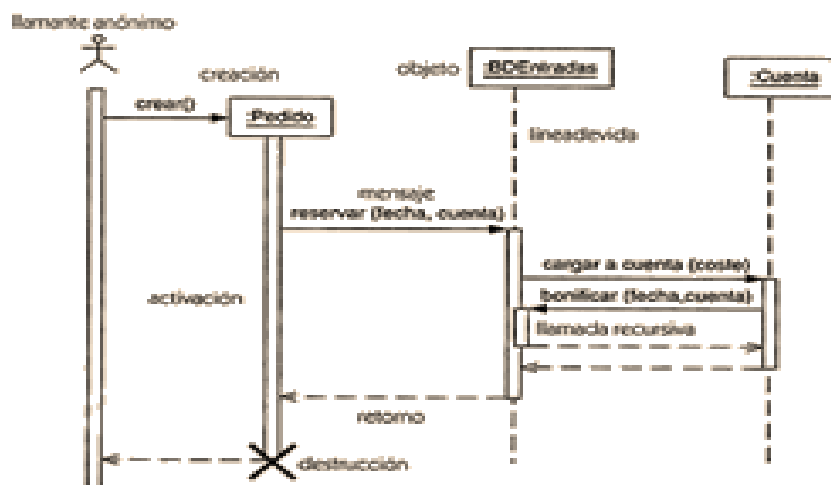
2.8.1.3 Diagrama de Interacción

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Habitualmente, un diagrama de interacción capta el comportamiento de un solo caso de uso. Hay dos tipos de diagrama de interacción:

2.8.1.4 Diagrama de secuencia

Muestran las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al de los objetos representados en los diagramas de colaboración, es decir son instancias concretas de una clase que participa en la interacción.

En un diagrama de secuencia un objeto se muestra como caja en la parte superior de la línea superior punteada, esta línea se llama línea de vida del objeto y representa la vida del objeto durante la interacción. En este tipo de diagramas también intervienen los mensajes, que son la forma en que se comunican los objetos: el objeto origen solicita (llama a) una operación del objeto destino, permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el tiempo de transición, que se especifica en el diagrama.



2.8.1.5 Diagrama de colaboración

(desic.upv) Muestra la interacción entre varios objetos y los enlaces que existen entre ellos.

Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones

muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente.

Formando parte de los diagramas de colaboración nos encontramos con objetos, enlaces y mensajes. Un objeto es una instancia de una clase que participa como una interacción, existen objetos simples y complejos. Un objeto es activo si posee un thread o hilo de control y es capaz de iniciar la actividad de control, mientras que un objeto es pasivo si mantiene datos pero no inicia la actividad.

Un enlace es una instancia de una asociación que conecta dos objetos de un diagrama de colaboración. El enlace puede ser reflexivo si conecta a un elemento consigo mismo. La existencia de un enlace entre dos objetos indica que puede existir un intercambio de mensajes entre los objetos conectados.








Los diagramas de interacción indican el flujo de mensajes entre elementos del modelo, el flujo de mensajes representa el envío de un mensaje desde un objeto a otro si entre ellos existe un enlace. Los mensajes que se envían entre objetos pueden ser de distintos tipos, también según como se producen en el tiempo; existen mensajes simples, sincrónicos, balking, timeout y asíncronos. Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces.

2.8.1.6 Diagrama de Actividades

(sparxsystems.com.ar) En UML un diagrama de actividades se usa para mostrar la secuencia de actividades, muestran el flujo de trabajo desde el punto de inicio hasta el punto final detallando muchas de las rutas de decisiones que existen en el progreso de eventos contenidos en la actividad.

Los diagramas de actividades son útiles para el modelado de negocios donde se usan para detallar el proceso involucrado en las actividades del negocio, son útiles para describir procesos complicados, pueden ser usados para describir un Caso de Uso.

Un diagrama de actividad permite seleccionar el orden en que se harán las cosas y pueden manejar procesos paralelos.

SÍMBOLO	SIGNIFICADO
	Punto de inicio del proceso
	Actividad
	Condicional
	Flujo de secuencia
	Bifurcación o entrada
	Punto final del proceso
	Swimlanes ("Calles")

3.1 Análisis de Requerimientos

Siguiendo la metodología RUP, la determinación de requerimientos es desarrollar un modelo del sistema que se va a construir, y la utilización de los casos de uso es una forma adecuada de crear ese modelo.

3.2 Vista de Casos de Uso

Los casos de uso son un mecanismo ampliamente utilizado para descubrir y registrar los requisitos, influyen mucho respecto de un proyecto. La escritura de caso de uso es una técnica excelente para entender y describir los requisitos.

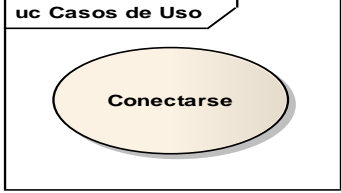




3.2.1 Identificación de Actores




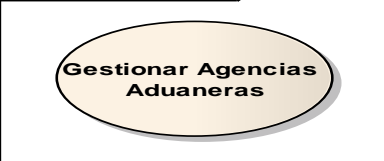


Para iniciar el análisis se comienza con la identificación de los actores que interactúan con el sistema, un actor es alguien o algo que tiene que interactuar con el sistema que se está desarrollando.

- **Administrador del sistema.**_ El actor encargado de administrar el sistema es el que se ocupa de las tareas administrativas del sistema es decir el registro de usuarios, modificación de datos del usuario, asignar roles, modificación de roles, asignar permisos, modificar permisos, además como también la modificación de login y password.
- **Encargado de Importación.**_ El actor usuario interactúa de manera directa con el sistema, es el encargado de ingresar los documentos y datos sobre la importación como la orden de compra, los proveedores, la factura internacional, los pagos realizados, la agencia aduanera

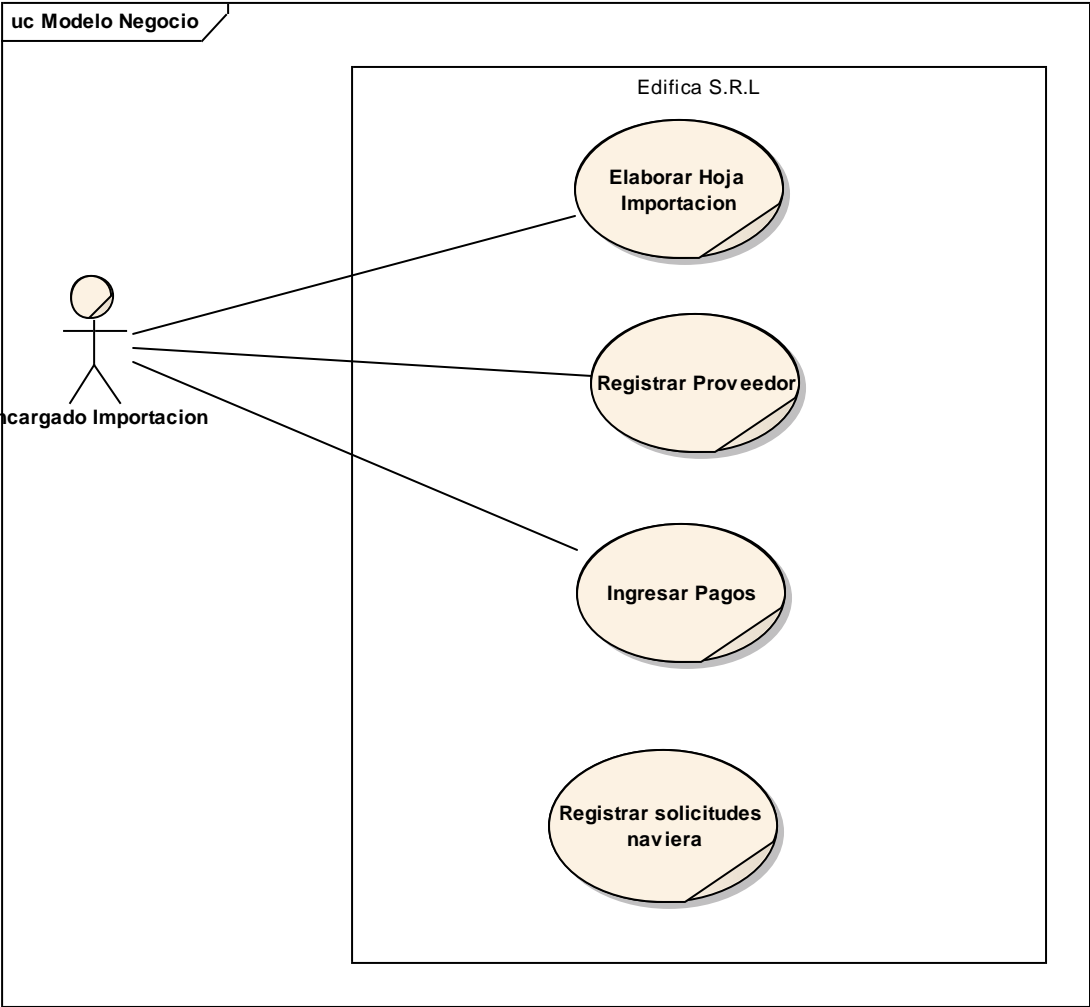
que realizara el trámite de importación de la mercadería así como la empresa de transporte marítimo y transporte terrestre de acuerdo al puerto de ingreso.

3.2.2 Descripción de Casos de Uso

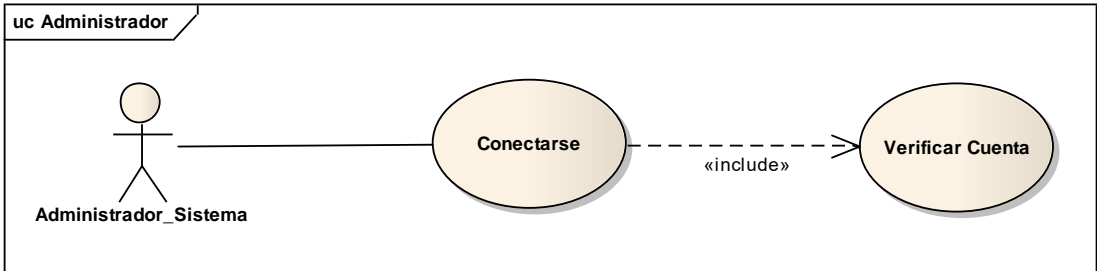
Caso de Uso	Descripción
	<p>Con el nombre y contraseña del usuario, se permite el ingreso del mismo al sistema.</p>
	<p>Permite registrar, modificar y asignar los datos de los usuarios.</p>
	<p>Permite adicionar los datos de la empresa en el sistema.</p>
	<p>Permite ingresar, modificar una importación es aquella que me permite hacer el seguimiento de los documentos y pagos.</p>
	<p>Permite generar reporte de pagos y detalle de documentos de la importación.</p>

<p>uc Casos de Uso</p> 	<p>Permite ingresar, modificar datos del proveedor para relacionarlo con una orden de compra</p>
<p>uc Casos de Uso</p> 	<p>Permite ingresar, modificar datos de la orden de compra</p>
<p>uc Casos de Uso</p> 	<p>Permite ingresar los pagos a los proveedores, tributos, empresas de transporte marítimo, empresa de transporte terrestre.</p>
<p>uc Casos de Uso</p> 	<p>Permite ingresar, modificar los datos de las agencias aduaneras</p>
<p>uc Casos de Uso</p> 	<p>Permite ingresar, modificar los datos de las empresas de transporte marítimo</p>
<p>uc Casos de Uso</p> 	<p>Permite ingresar, modificar los datos de las empresas de transporte terrestre</p>

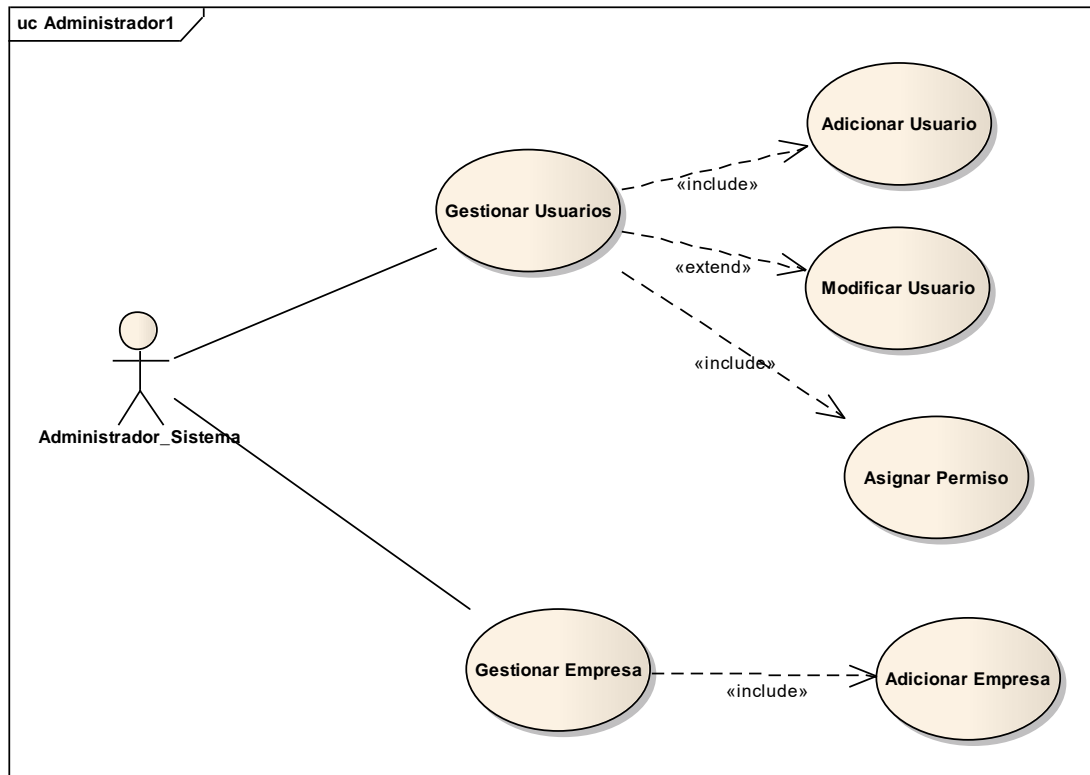
3.2.3 Diagrama de casos de uso Negocio



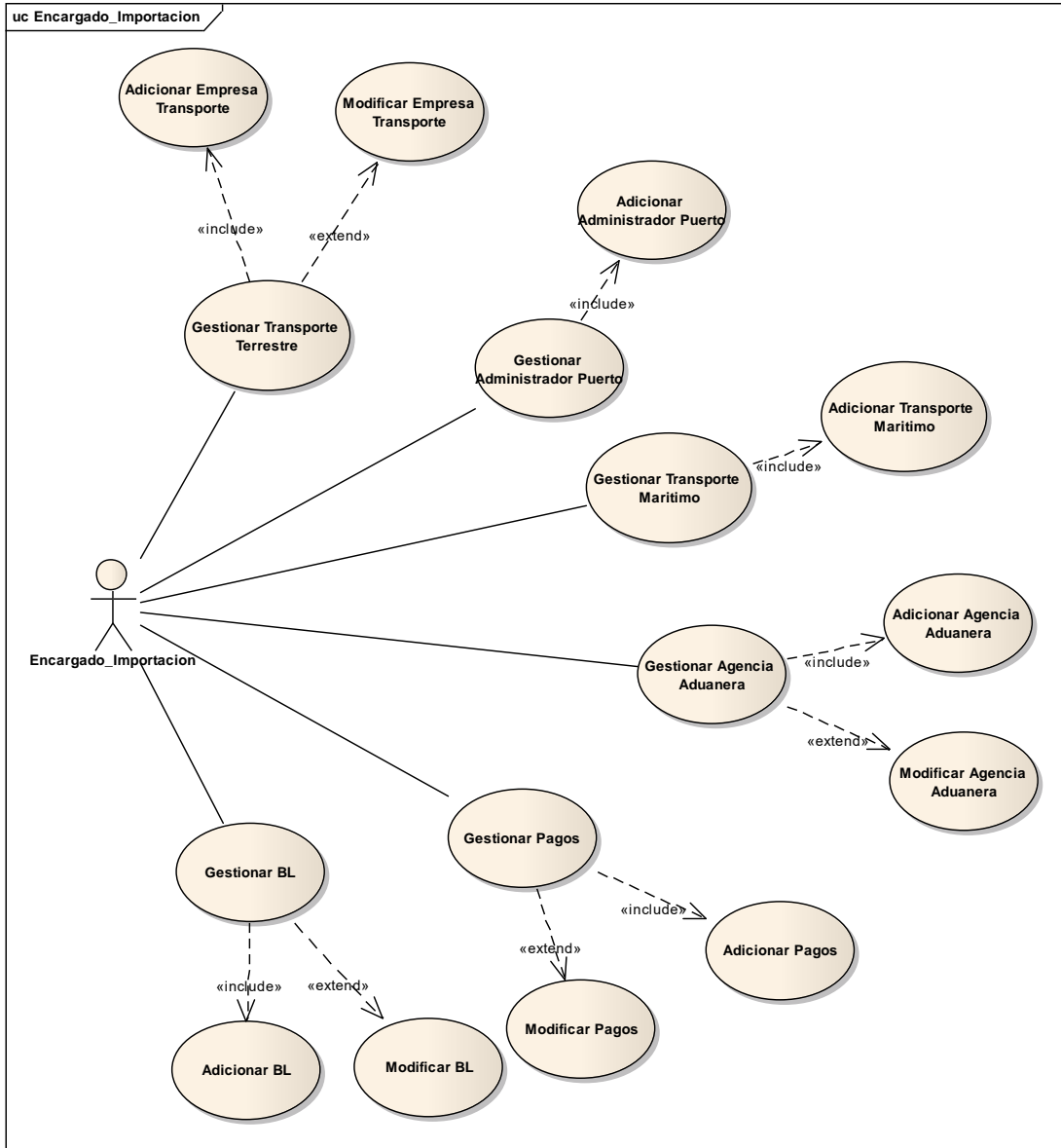
3.2.4 Diagrama de Casos de Uso: Conectarse

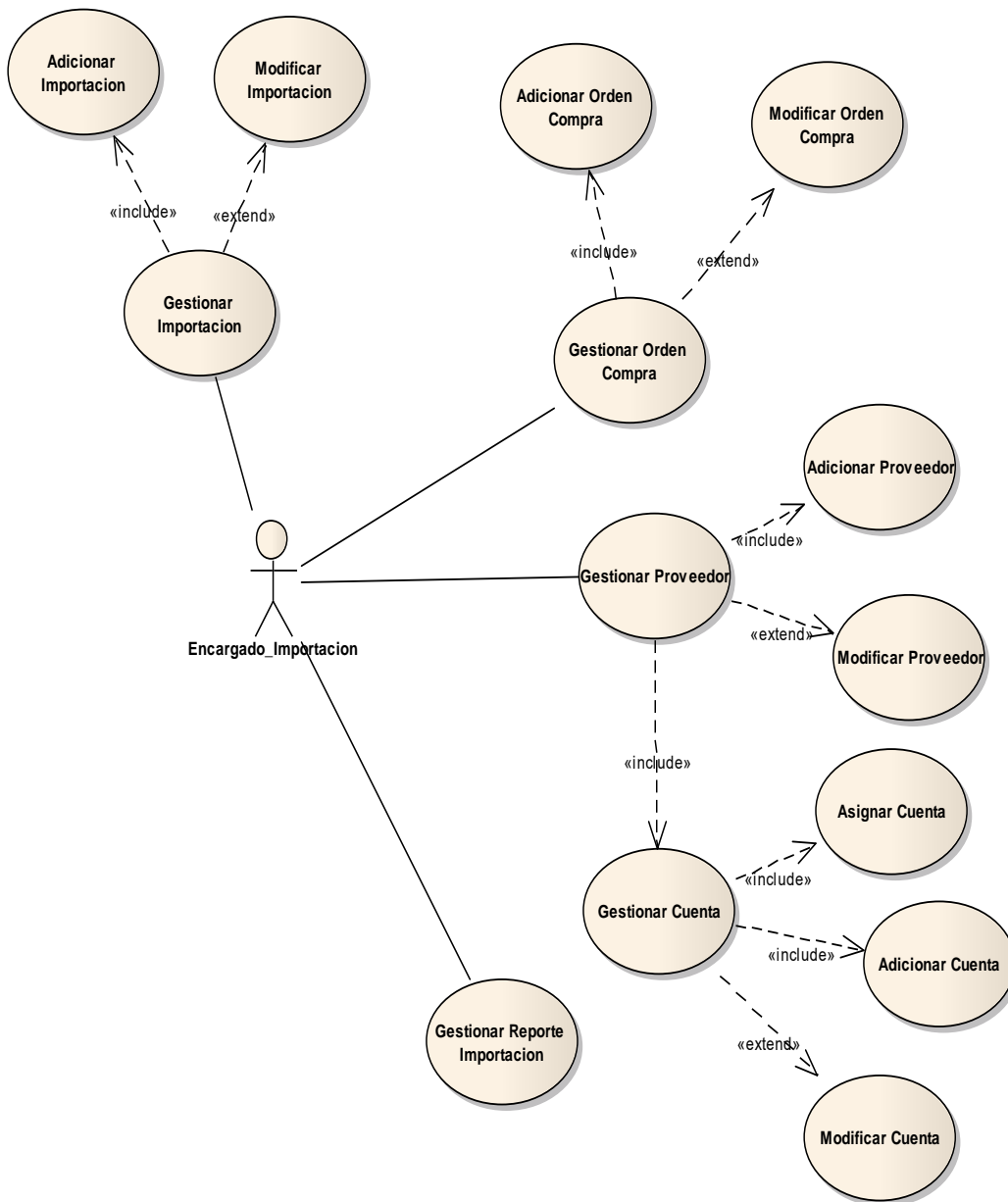


3.2.5 **Diagrama de Casos de Uso: Administrador Sistema.-** Es el encargado de adicionar, modificar y asignar usuarios, adicionar datos de una empresa.



3.2.6 **Diagrama de Casos de Uso: Encargado Importación.-** Es el encargado de adicionar, modificar empresa transporte, administrador puerto, adicionar transporte marítimo, adicionar, modificar agencia aduanera, adicionar pagos, además de generar reportes.





3.2.7 Especificaciones de los Casos de Uso

Casos de Uso	Conectarse.
Descripción	El usuario ingresa su usuario y clave.
Actor Principal	Usuario.
Actor secundario	
Tipo	Primario.
Precondición	<ul style="list-style-type: none">✓ Tener un usuario.✓ Ingresado al sistema.
Flujo Normal	<ol style="list-style-type: none">1. El sistema despliega el formulario para introducir el usuario y clave.2. El usuario ingresa el usuario y clave.3. El usuario hace click en el botón ingresar.
Flujo Alternativo	<ol style="list-style-type: none">1. El usuario hace click en cerrar cancelando el ingreso al sistema.2. El usuario no cuenta con usuario y hace click en registrar usuario.
Pos condición	Validar Usuario.

Casos de Uso	Gestionar Empresa.
Descripción	El Administrador Sistema ingresa los datos de la empresa.
Actor	Administrador Sistema.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar de empresa.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra los datos de la empresa. 2. El Administrador Sistema selecciona Adicionar. 3. El sistema le muestra el formulario para Adicionar. 4. El Administrador Sistema introduce los datos de la empresa. 5. El Administrador Sistema hace click en Aceptar. 6. El sistema guarda en la base de datos el registro. 7. El sistema muestra mensaje datos registrados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El Administrador Sistema hace click en el botón cancelar cancelando la adición. 2. El administrador sistema introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Usuarios.
Descripción	El Administrador Sistema registrar, modificar o eliminar los datos de los usuarios.
Actor	Administrador Sistema.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Usuarios.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de los usuarios. 2. El Administrador sistema selecciona registrar. 3. El sistema le muestra el formulario para registrar. 4. El Administrador Sistema introduce los datos de los usuarios. 5. El Administrador Sistema hace click en Aceptar. 6. El Sistema guarda los datos en la base de datos. 7. El Sistema muestra un mensaje datos registrados correctamente. 8. El Administrador Sistema selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El Administrador Sistema introduce los datos a modificar. 11. El Administrador Sistema hace click en Aceptar. 12. El Sistema guarda los datos en la base de datos. 13. El Sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El Administrador Sistema hace click en el botón cancelar cancelando el registro, modificación. 2. El Administrador Sistema introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Empresa Transporte Terrestre.
Descripción	El encargado de importación adiciona, modificar, los datos de la empresa de transporte terrestre.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Empresa Transporte Terrestre.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de empresa de transporte terrestre. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de las empresas de transporte. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los datos de la empresa de transporte terrestre. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.

Pos condición	Adicionar Acción.
----------------------	-------------------

Casos de Uso	Gestionar Empresa Transporte Marítimo.
Descripción	El encargado de importación adiciona, los datos de la empresa de transporte marítimo.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Empresa Transporte Marítimo.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de empresa de transporte. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de las empresas de transporte. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, de los datos de la empresa de transporte marítimo. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Administrador de Puerto.
Descripción	El encargado de importación adiciona, los datos del administrador de puerto.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar administrador de puerto.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado del administrador de puerto. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos del administrador de puerto. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, de los datos del administrador de puerto. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Agencia Aduanera.
Descripción	El encargado de importación adiciona, modifica, los datos de la agencia aduanera.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Agencia Aduanera.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de agencia aduanera. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de la agencia aduanera. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los datos de la agencia aduanera. 2. El encargado de importación introduce datos inválidos. 4. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Proveedor.
Descripción	El encargado de importación adiciona, modifica, los datos del proveedor.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Proveedor.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de proveedores. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de los proveedores. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los datos los proveedores. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Orden de Compra.
Descripción	El encargado de importación adiciona, modifica, los datos de la orden de compra.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Orden de Compra.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de proveedores. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de la orden de compra. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los datos de una compra. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar Pagos.
Descripción	El encargado de importación adiciona, modifica, los pagos en una orden de compra.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Pagos.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de orden de compra. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos del pago relacionado a una orden de compra. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los pagos relacionado a una orden de compra. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

Casos de Uso	Gestionar BL.
Descripción	El encargado de importación adiciona, modifica, los datos de un BL que está relacionado con una orden de compra.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar BL.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de orden de compra. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos del BL relacionado a una orden de compra. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación del BL relacionado a una orden de compra. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

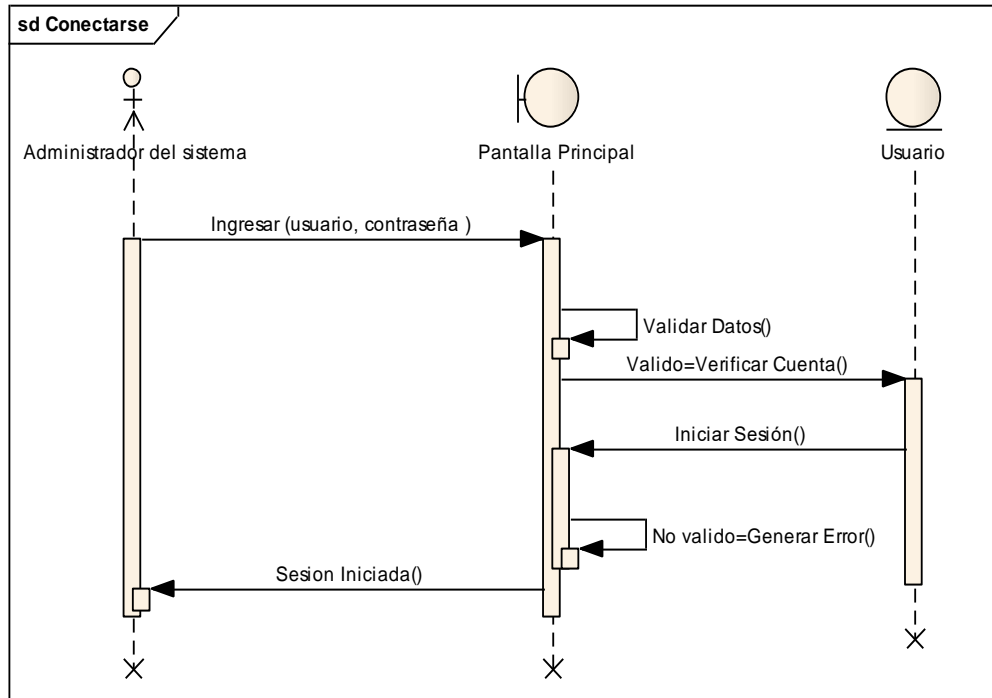
Casos de Uso	Gestionar Cuenta.
Descripción	El encargado de importación adiciona, modifica, las cuentas que tiene un proveedor.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Cuentas.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de proveedores. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de la cuenta de un proveedor. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de las cuentas. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.

Casos de Uso	Gestionar Importación.
Descripción	El encargado de importación adiciona, modifica, los datos de la hoja de importación.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Importación.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra un listado de importaciones. 2. El encargado de importación selecciona adicionar. 3. El sistema le muestra el formulario para registrar. 4. El encargado de importación introduce los datos de la hoja de importación. 5. El encargado de importación hace click en Aceptar. 6. El sistema guarda los datos en la base de datos. 7. El sistema muestra un mensaje datos registrados correctamente. 8. El encargado de importación selecciona modificar. 9. El sistema le muestra el formulario con los datos a modificar. 10. El encargado de importación introduce los datos a modificar. 11. El encargado de importación hace click en Aceptar. 12. El sistema guarda los datos en la base de datos. 13. El sistema muestra un mensaje datos modificados correctamente.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar cancelando la adición, modificación de los datos de la hoja de importación. 2. El encargado de importación introduce datos inválidos. 3. El sistema muestra mensaje de error.
Pos condición	Adicionar Acción.

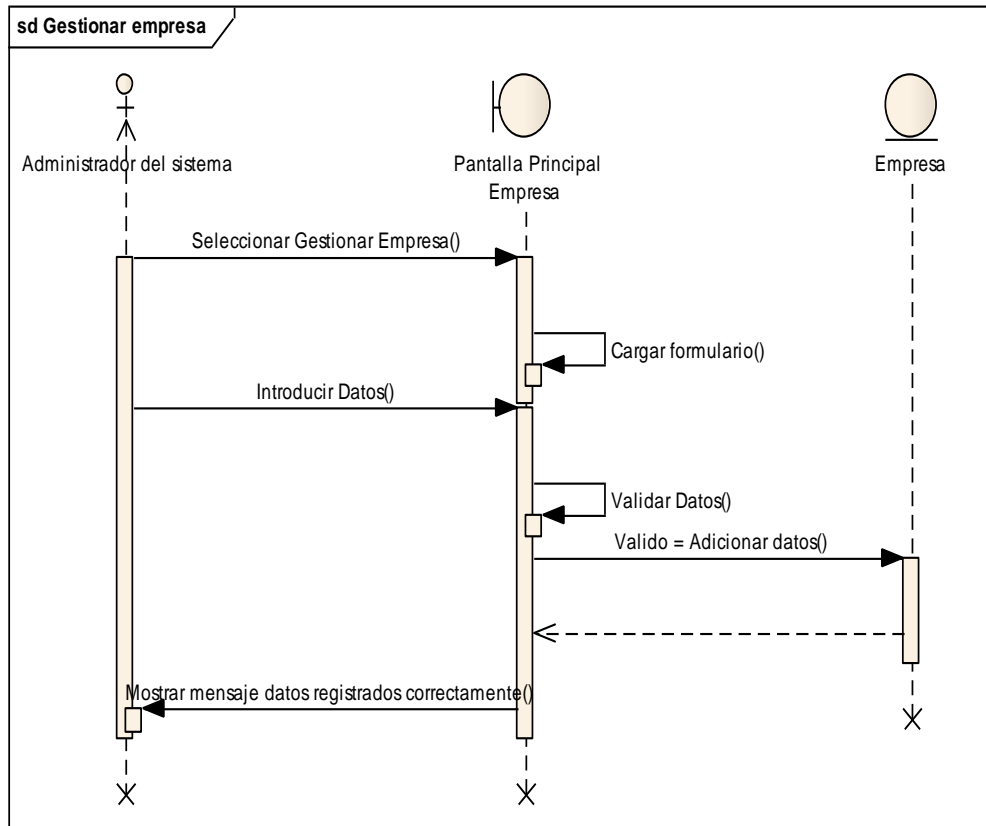
Caso de Uso	Gestionar reporte.
Descripción	El sistema le muestra el reporte de la hoja de importación.
Actor	Encargado Importación.
Tipo	Primario.
Precondición	<ul style="list-style-type: none"> ✓ Conectarse. ✓ Seleccionar Gestionar Reporte.
Flujo Normal	<ol style="list-style-type: none"> 1. El sistema le muestra formulario para elegir el reporte. 2. El encargado de importación elige el reporte deseado hace click en Aceptar. 3. El sistema muestra el reporte elegido. 4. El encargado de importación selecciona imprimir. 5. El sistema visualiza el reporte a imprimir. 6. El encargado de registro imprime el reporte deseado.
Flujo Alternativo	<ol style="list-style-type: none"> 1. El encargado de importación hace click en el botón cancelar.
Pos condición	Validar Datos.

3.3 Diagramas de Secuencia

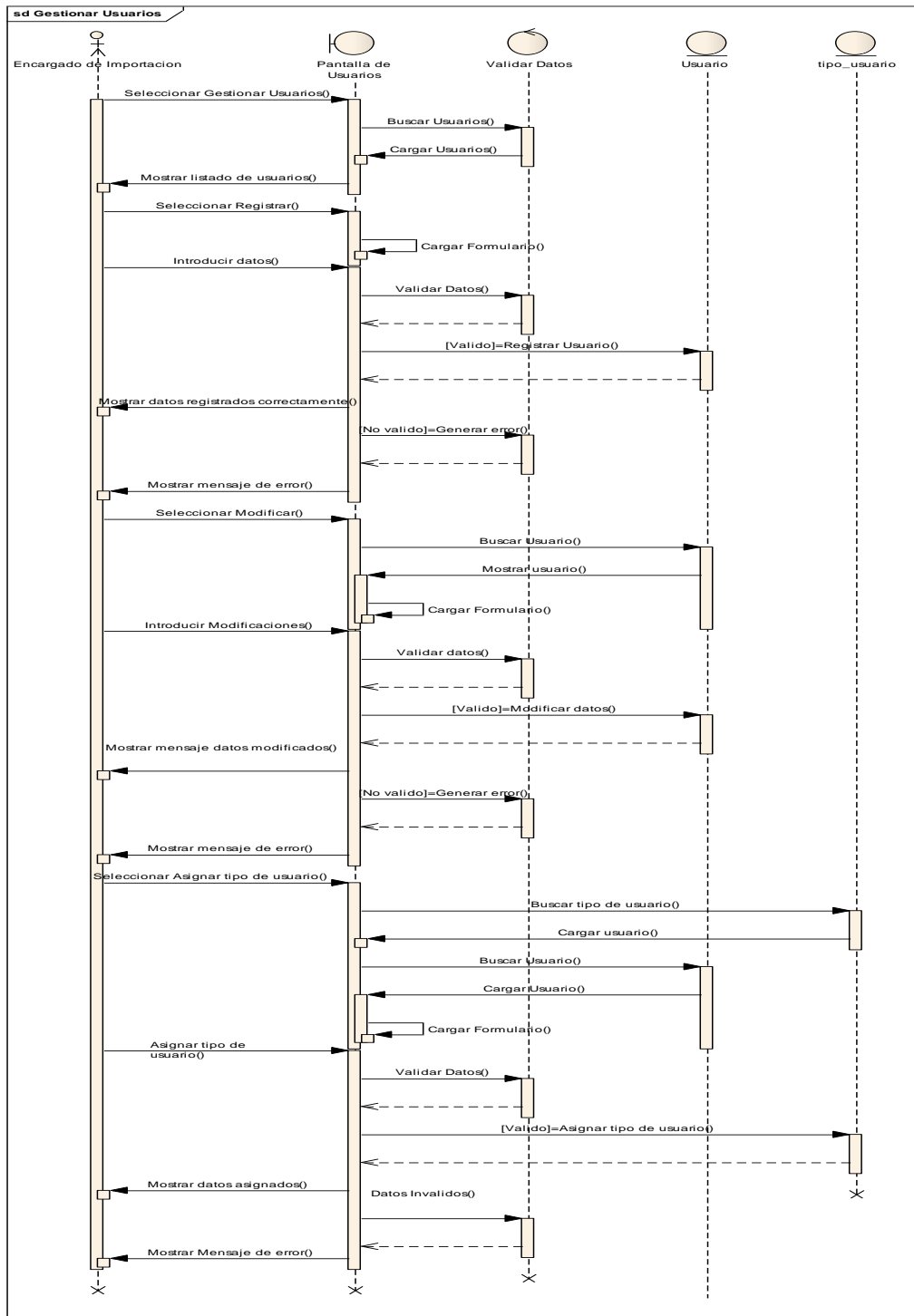
3.3.1 Diagrama de Secuencia: Conectarse



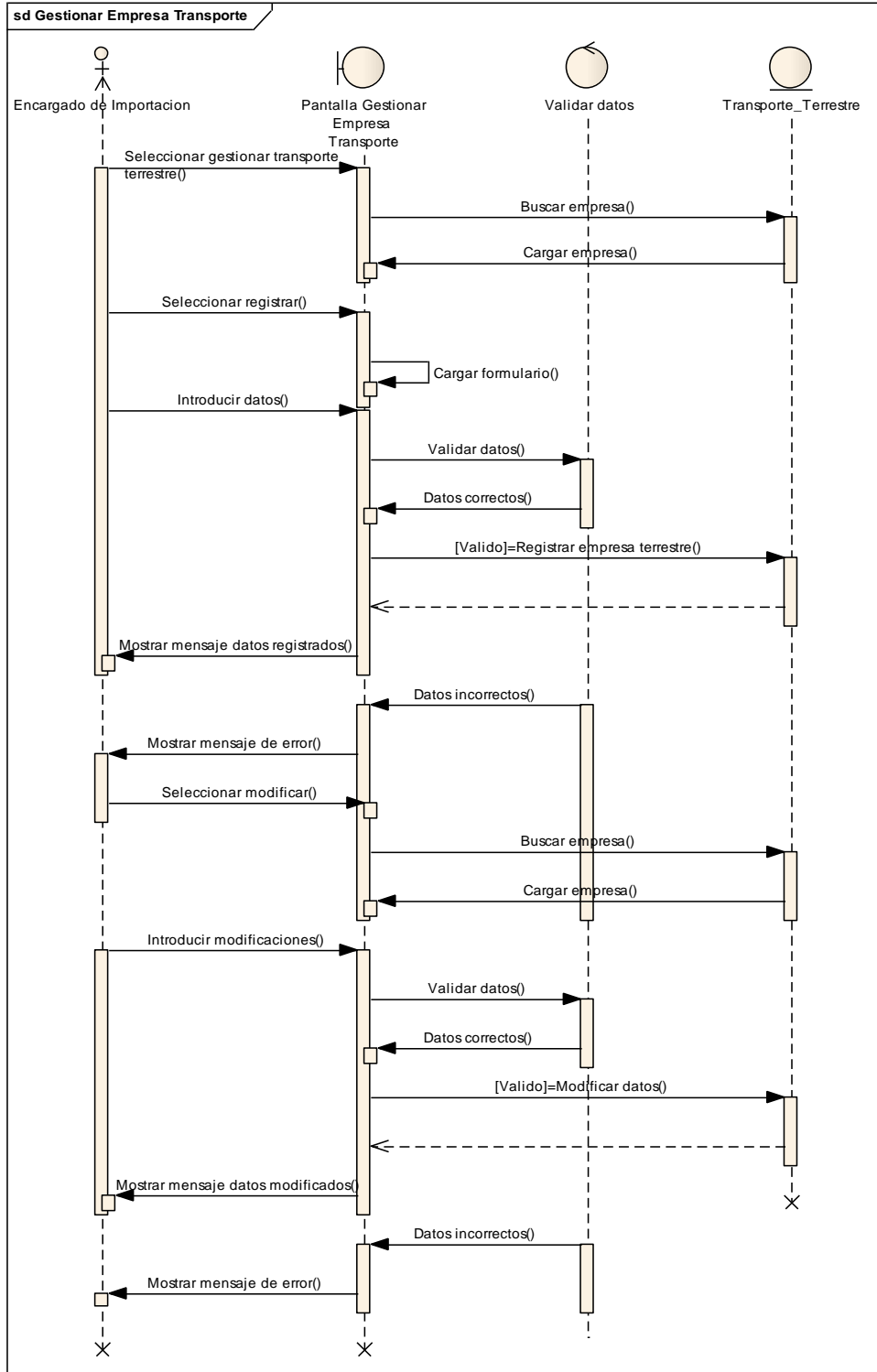
3.3.2 Diagrama de Secuencia: Gestionar Empresa



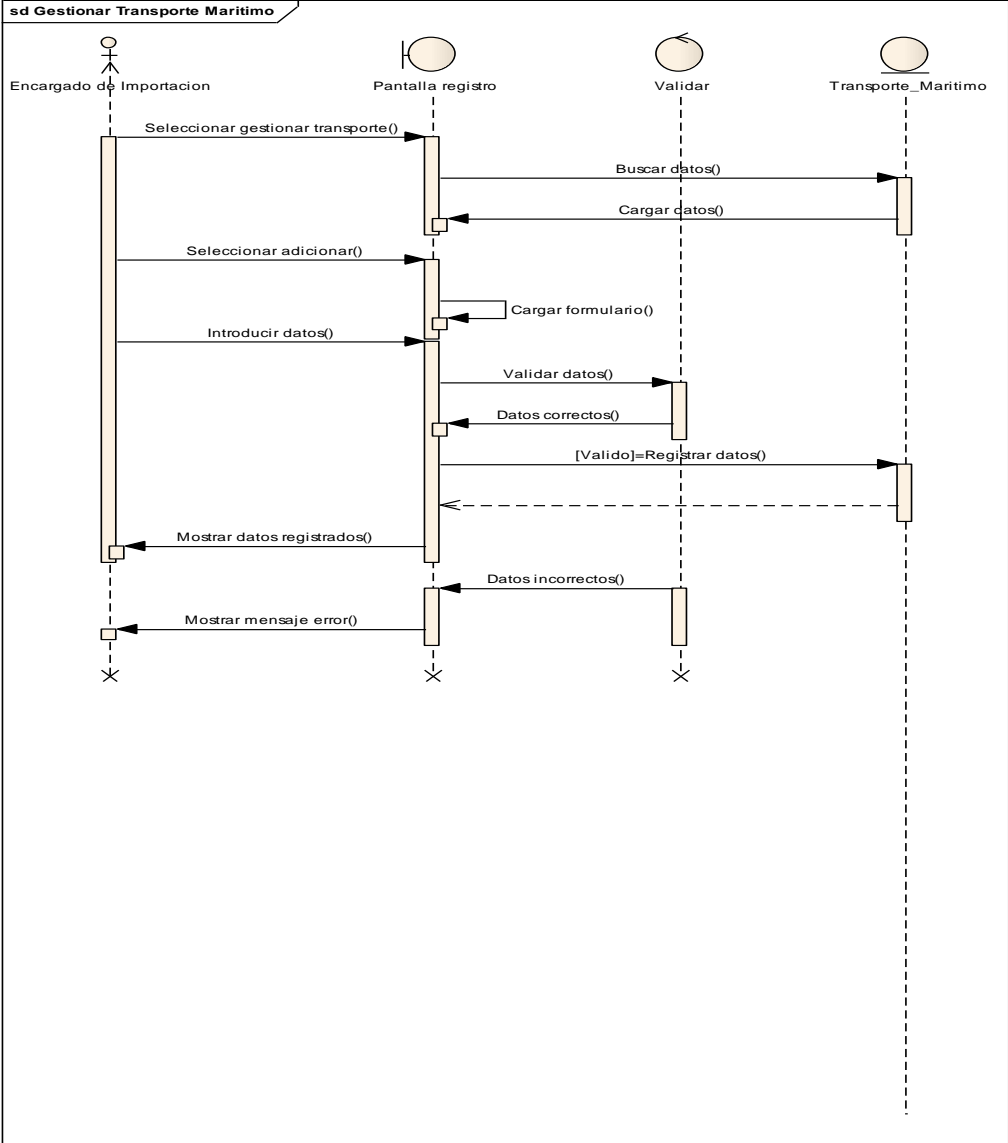
3.3.3 Diagrama de Secuencia: Gestionar Usuarios



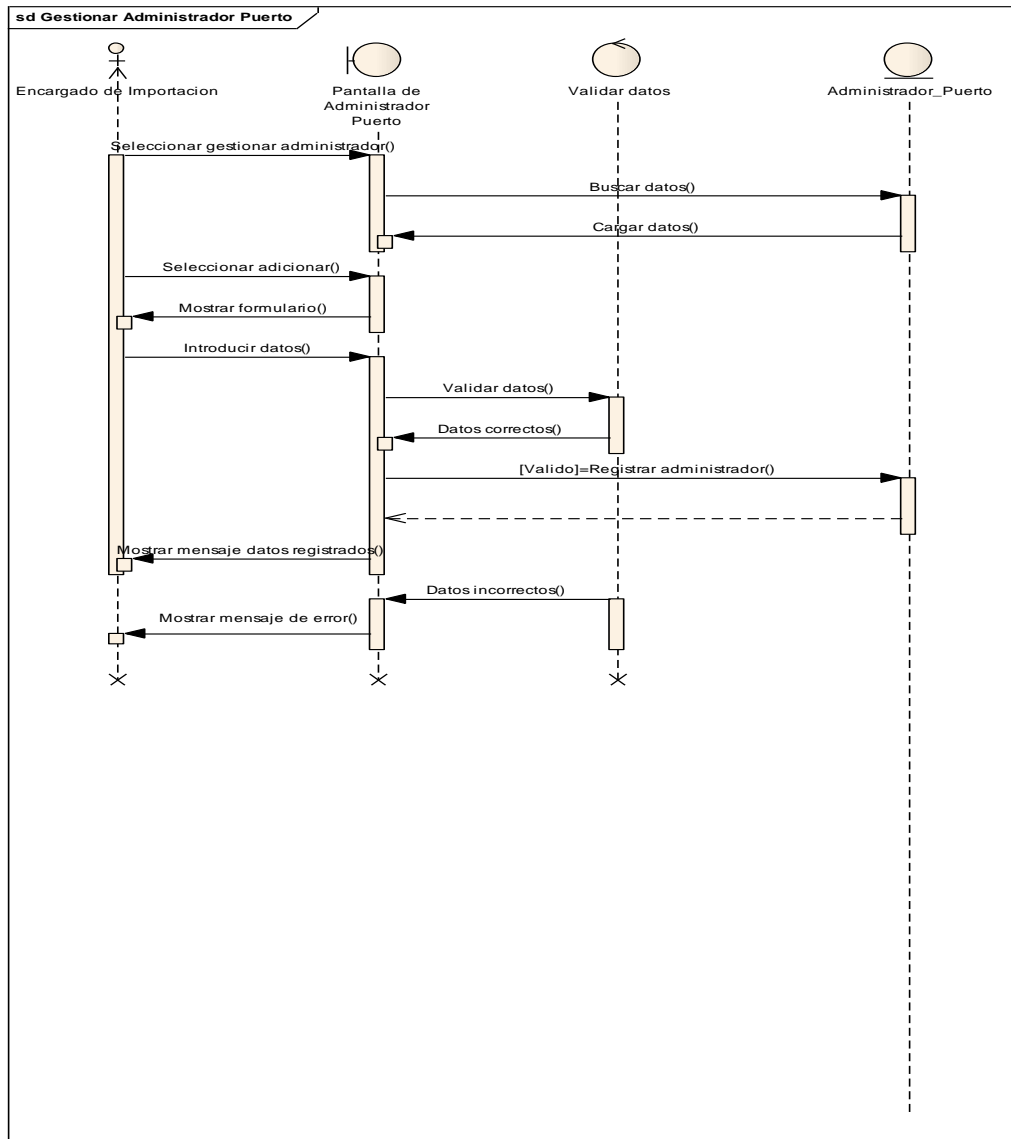
3.3.4 Diagrama de Secuencia: Gestionar Empresa Terrestre



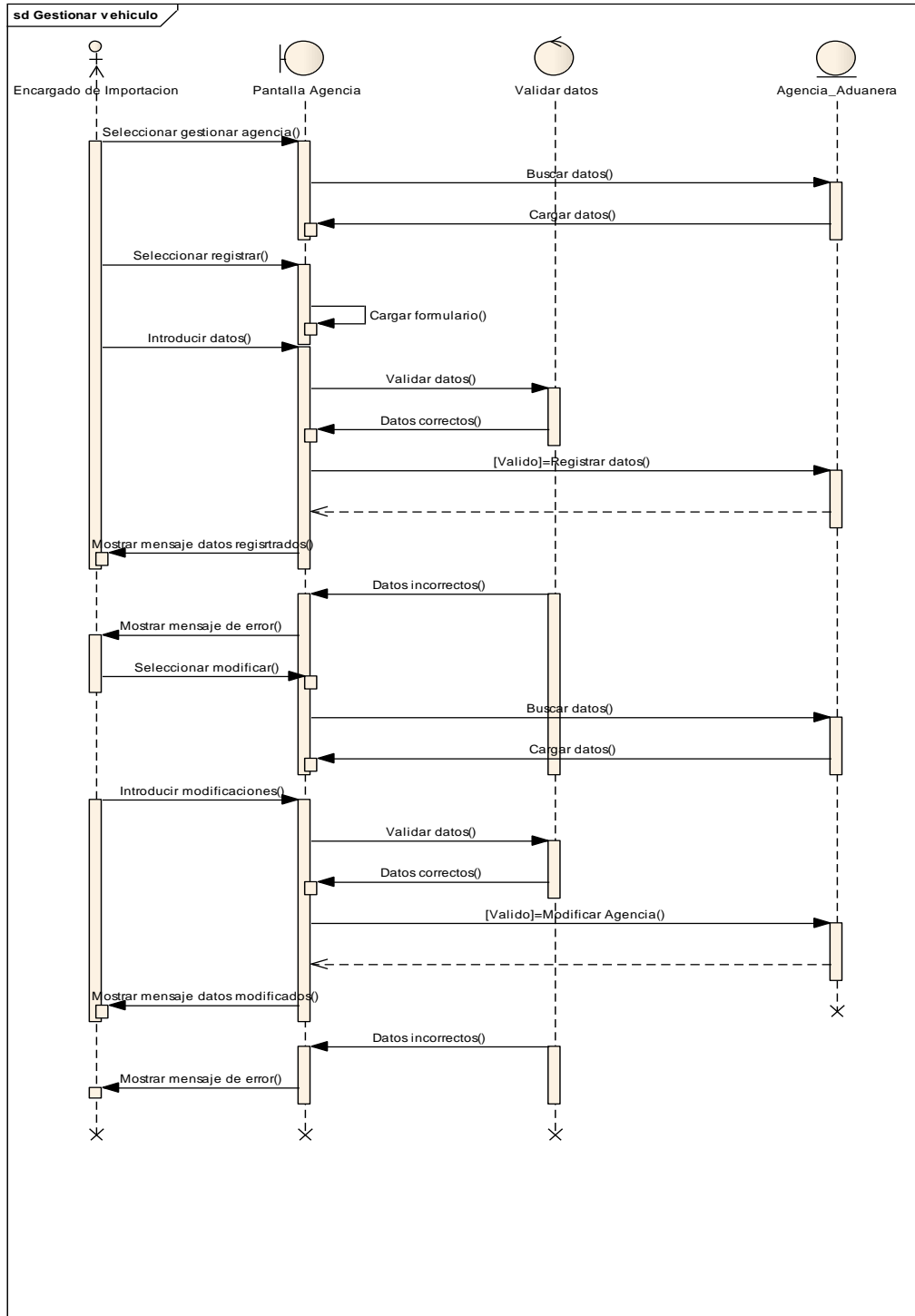
3.3.5 Diagrama de Secuencia: Gestionar Transporte Marítimo



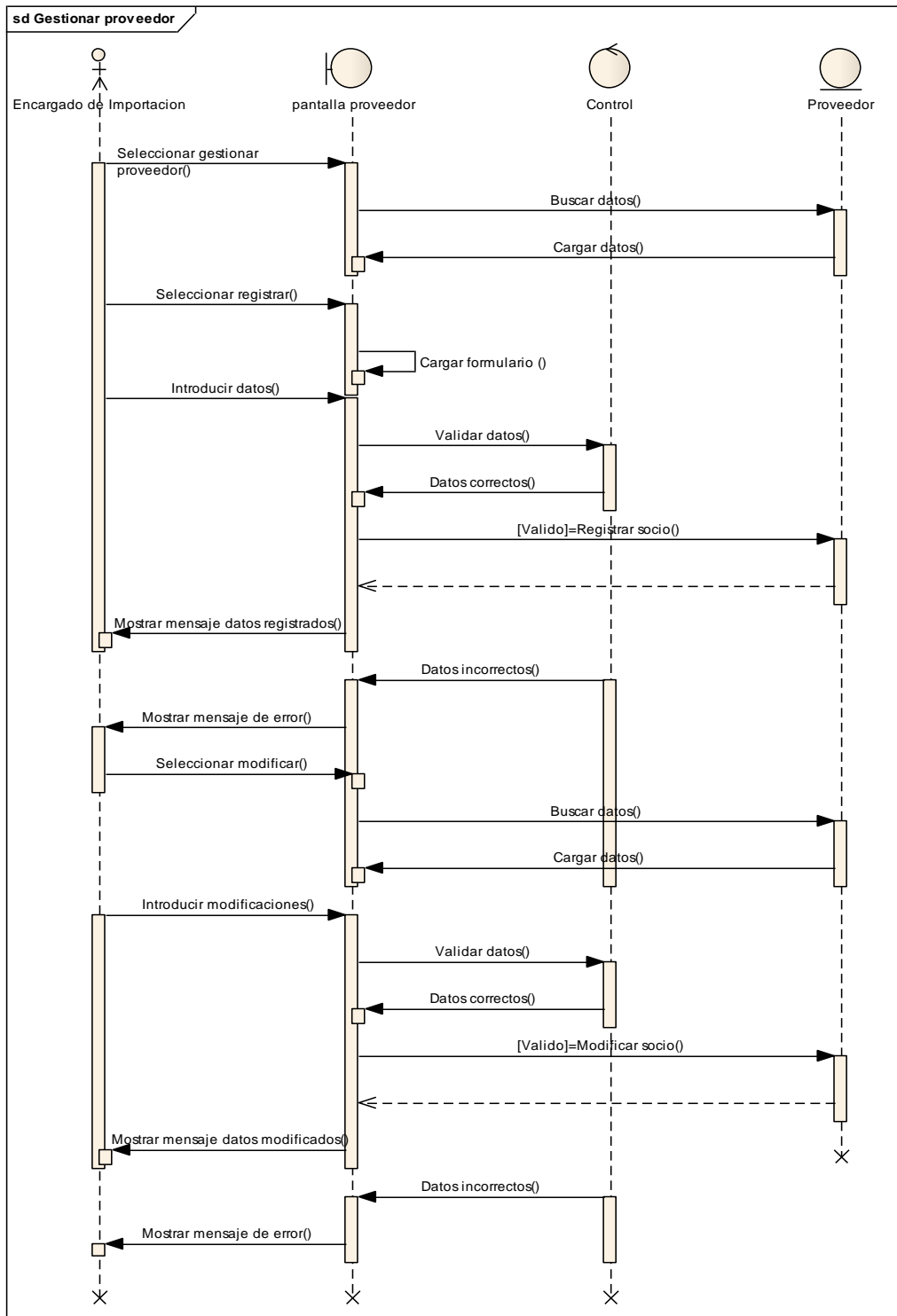
3.3.6 Diagrama de Secuencia: Gestionar Administrador Puerto



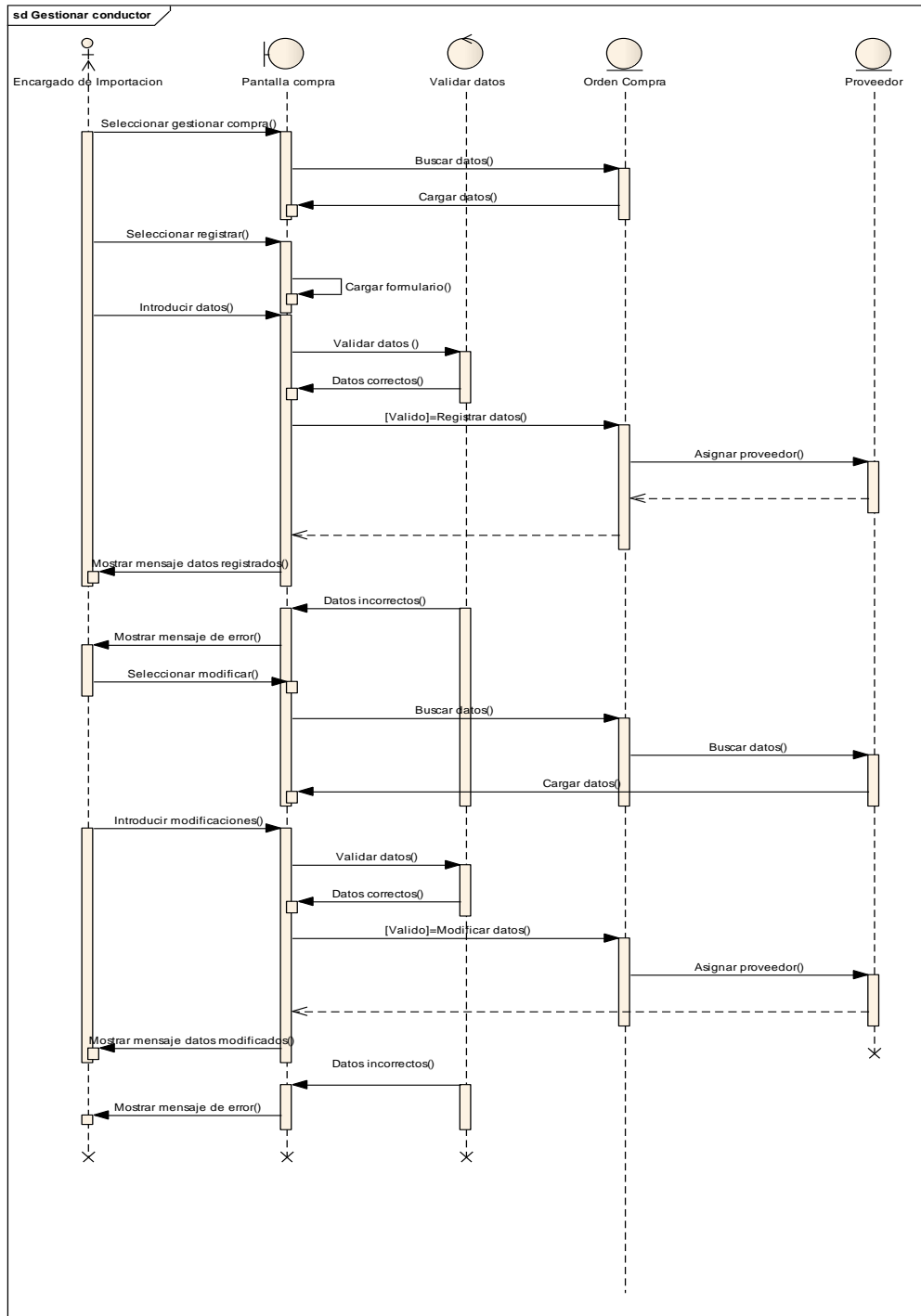
3.3.7 Diagrama de Secuencia: Gestionar Agencia Aduanera



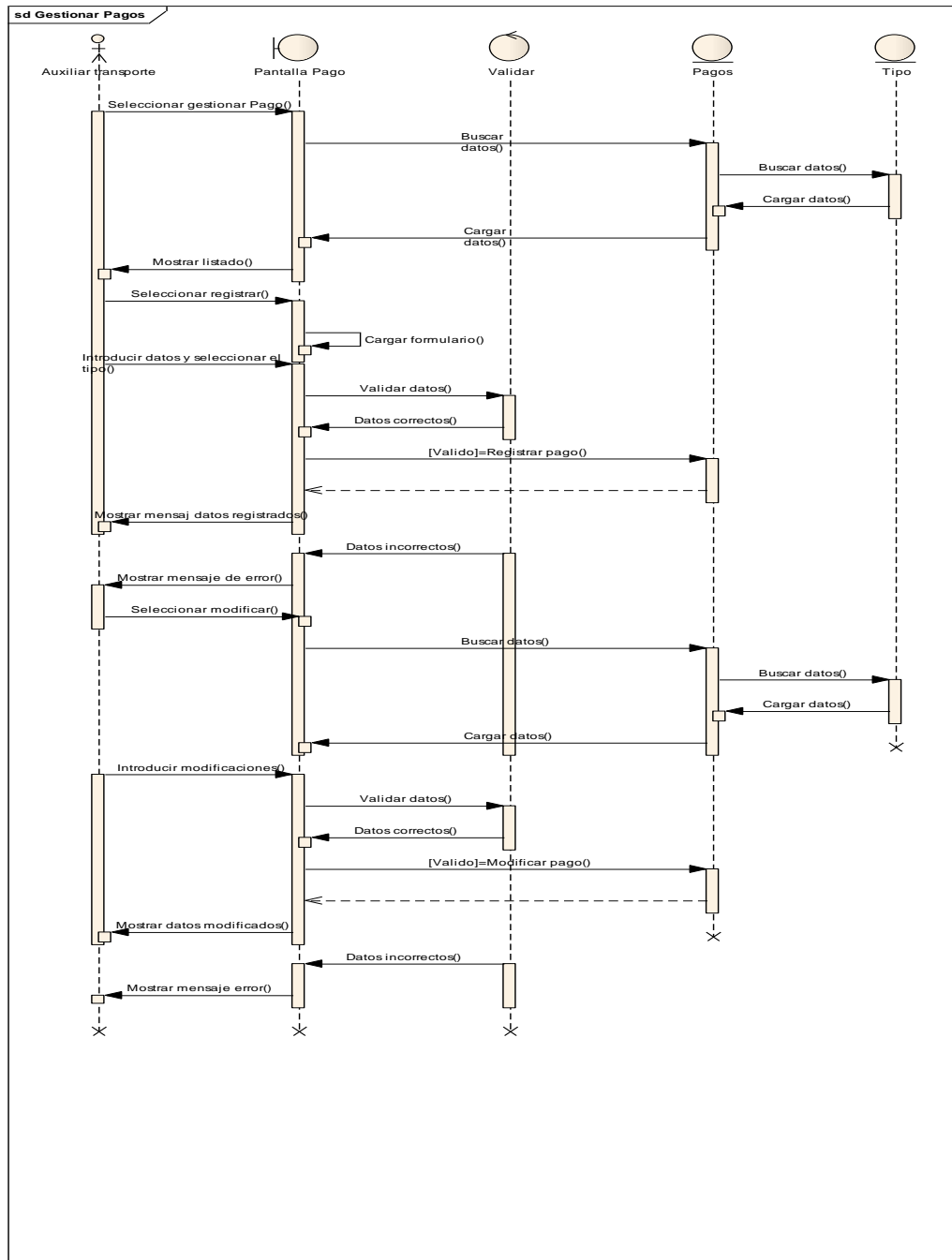
3.3.8 Diagrama de Secuencia: Gestionar Proveedor



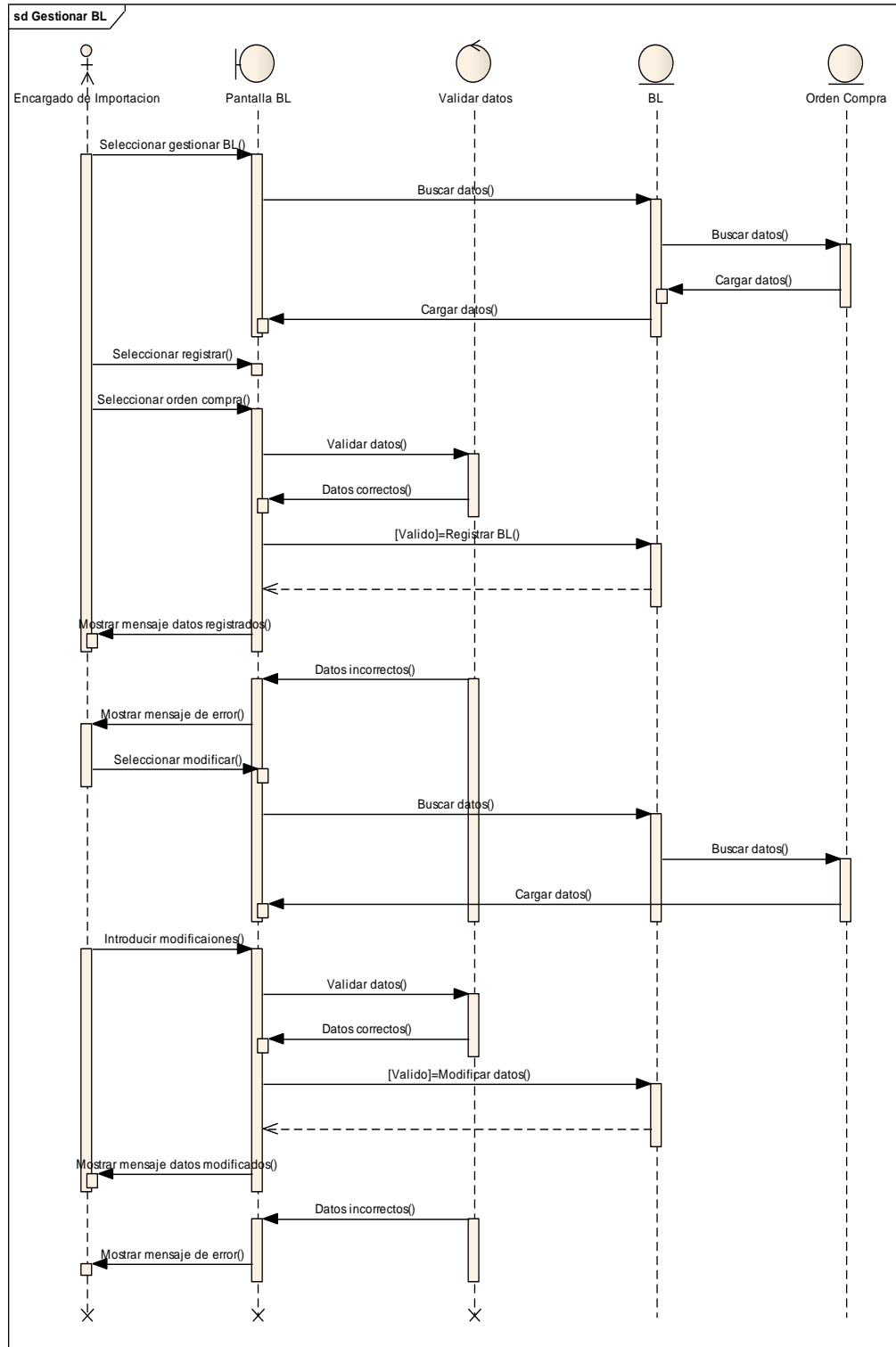
3.3.9 Diagrama de Secuencia: Gestionar Orden de Compra



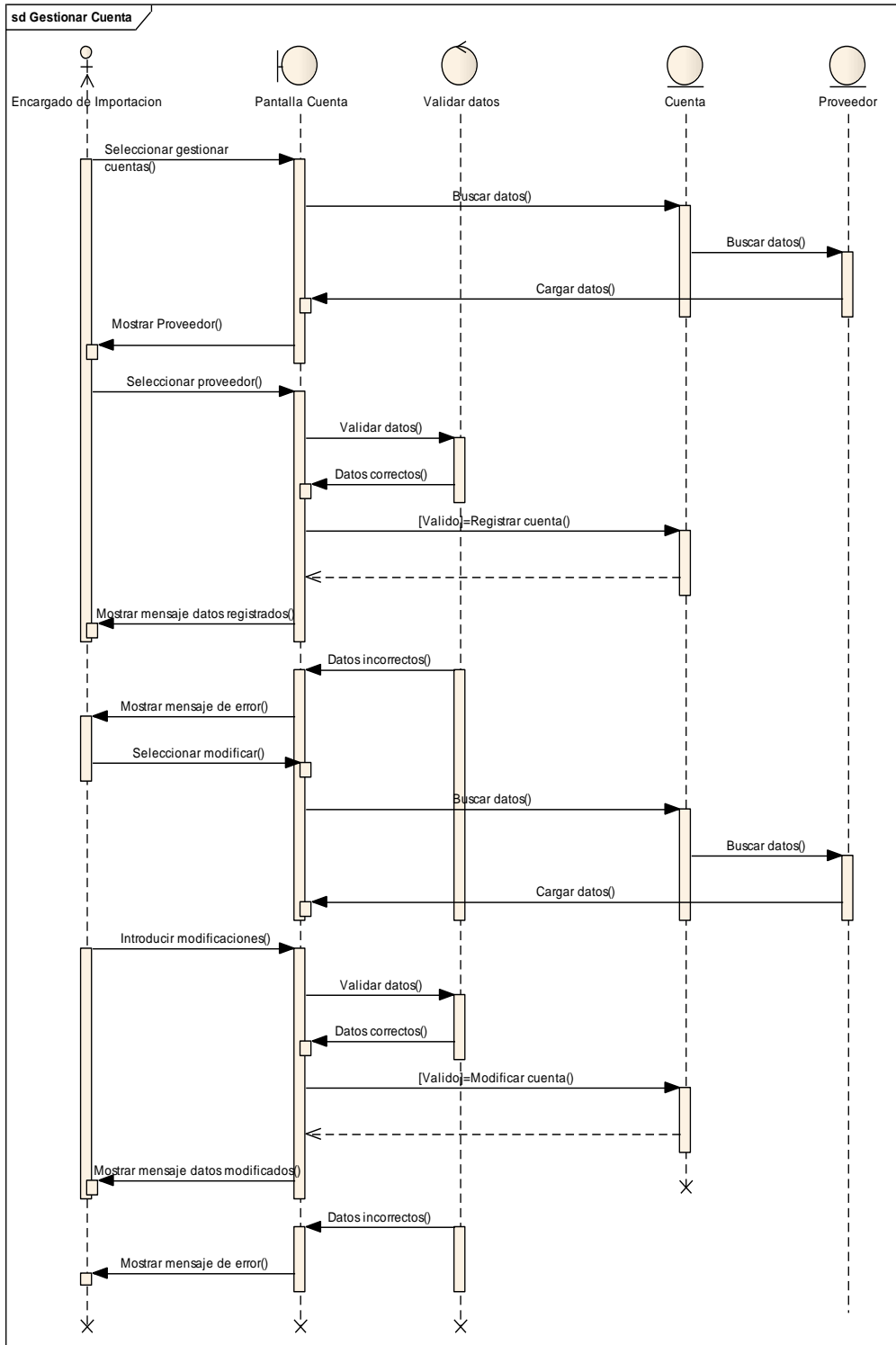
3.3.10 Diagrama de Secuencia: Gestionar Pagos



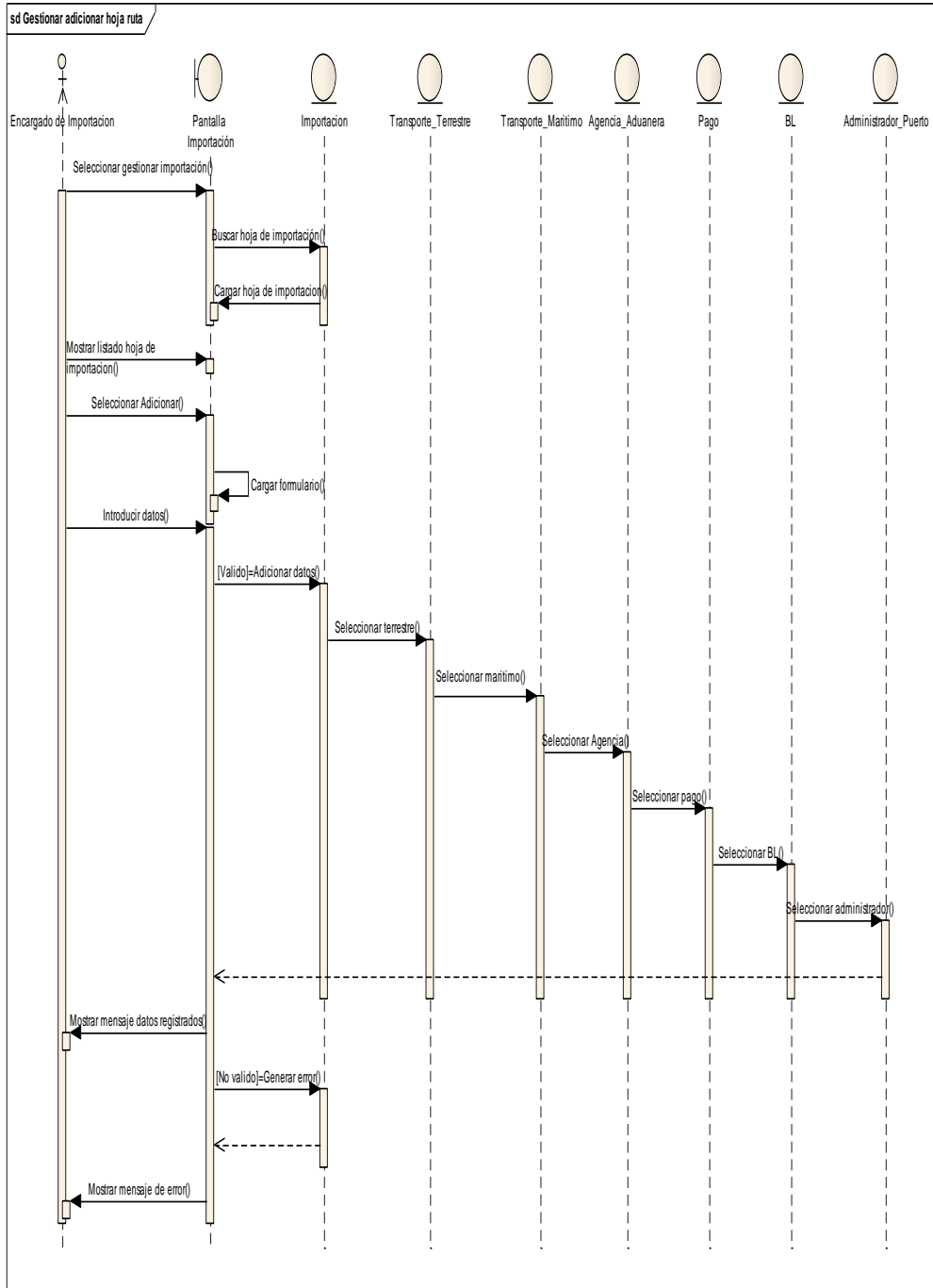
3.3.11 Diagrama de Secuencia: Gestionar BL



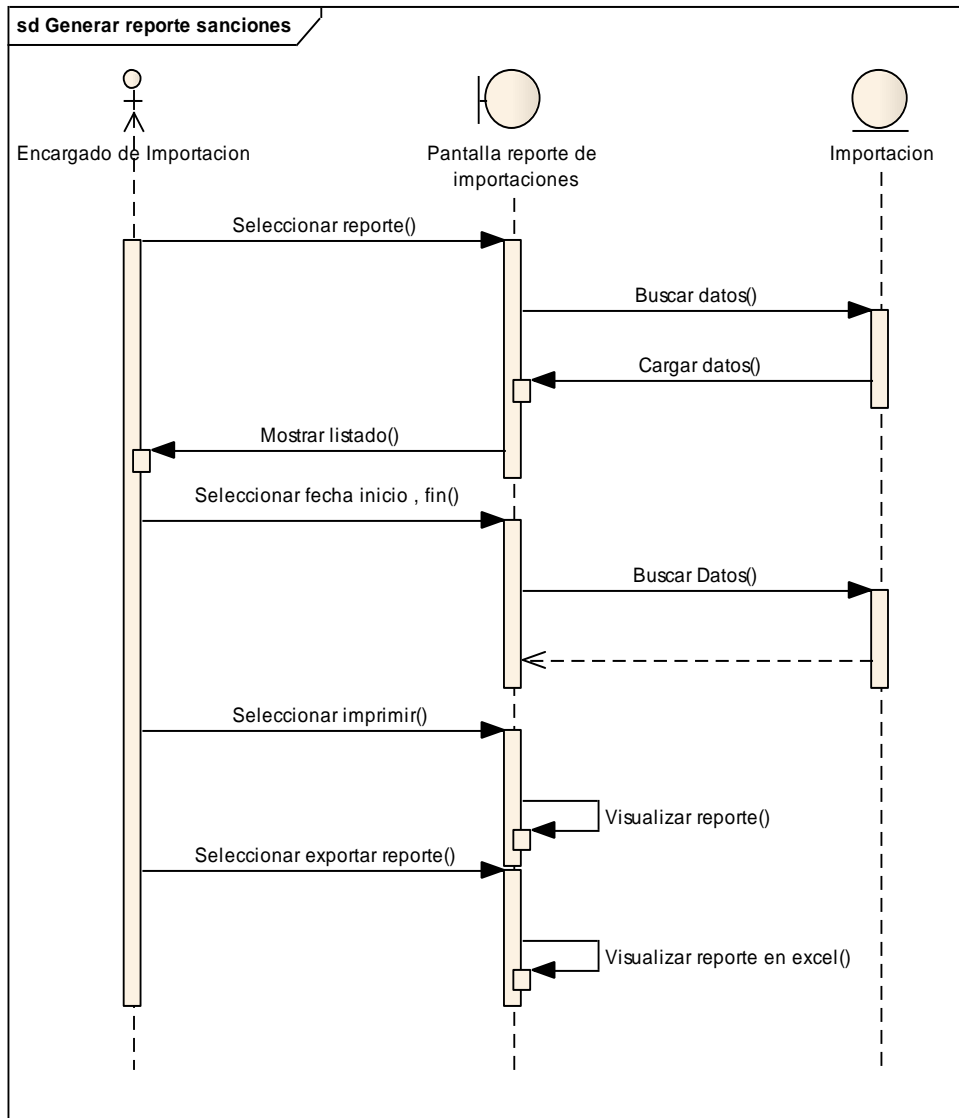
3.3.12 Diagrama de Secuencia: Gestionar Cuenta



3.3.13 Diagrama de Secuencia: Gestionar Importación

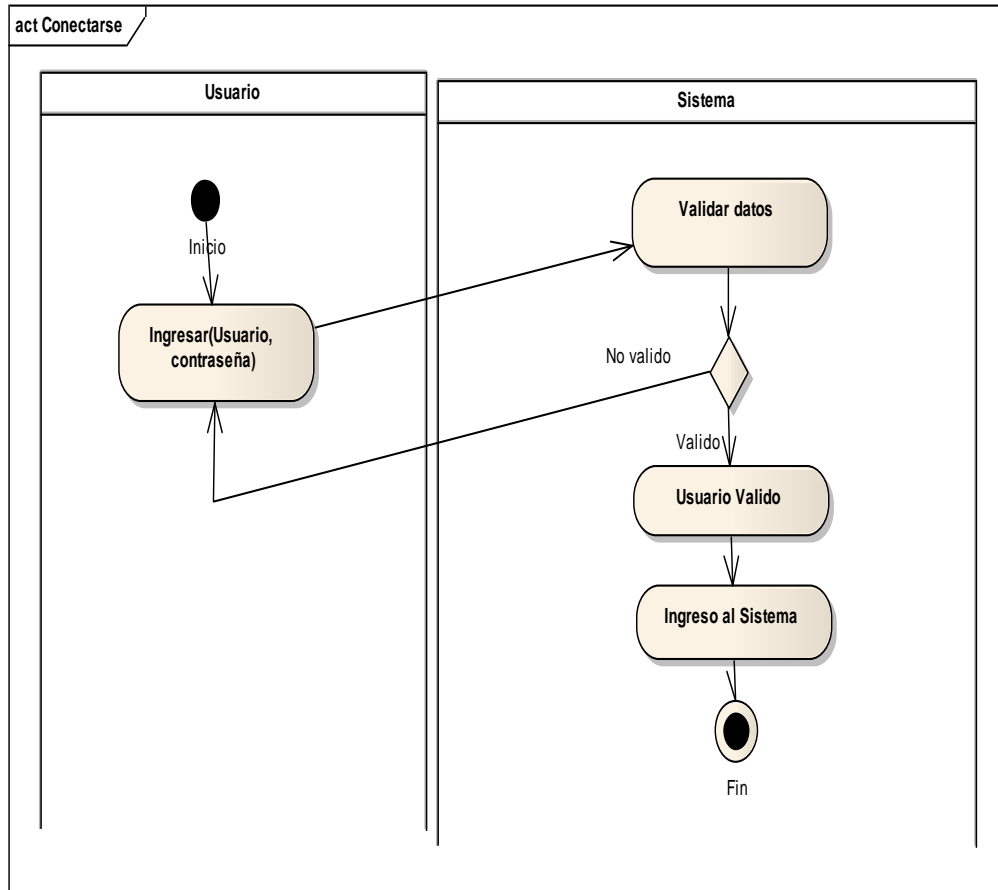


3.3.14 Diagrama de Secuencia: Generar Reporte Importación

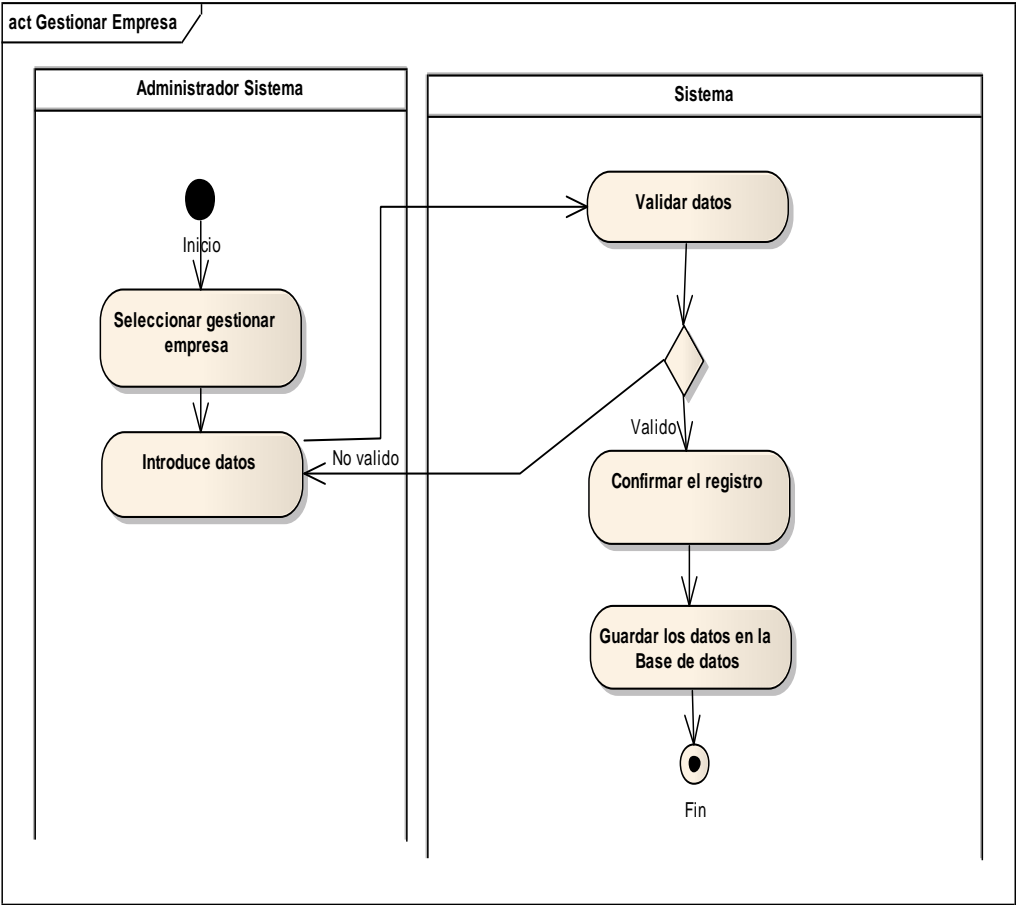


3.4 Diagramas de Actividades

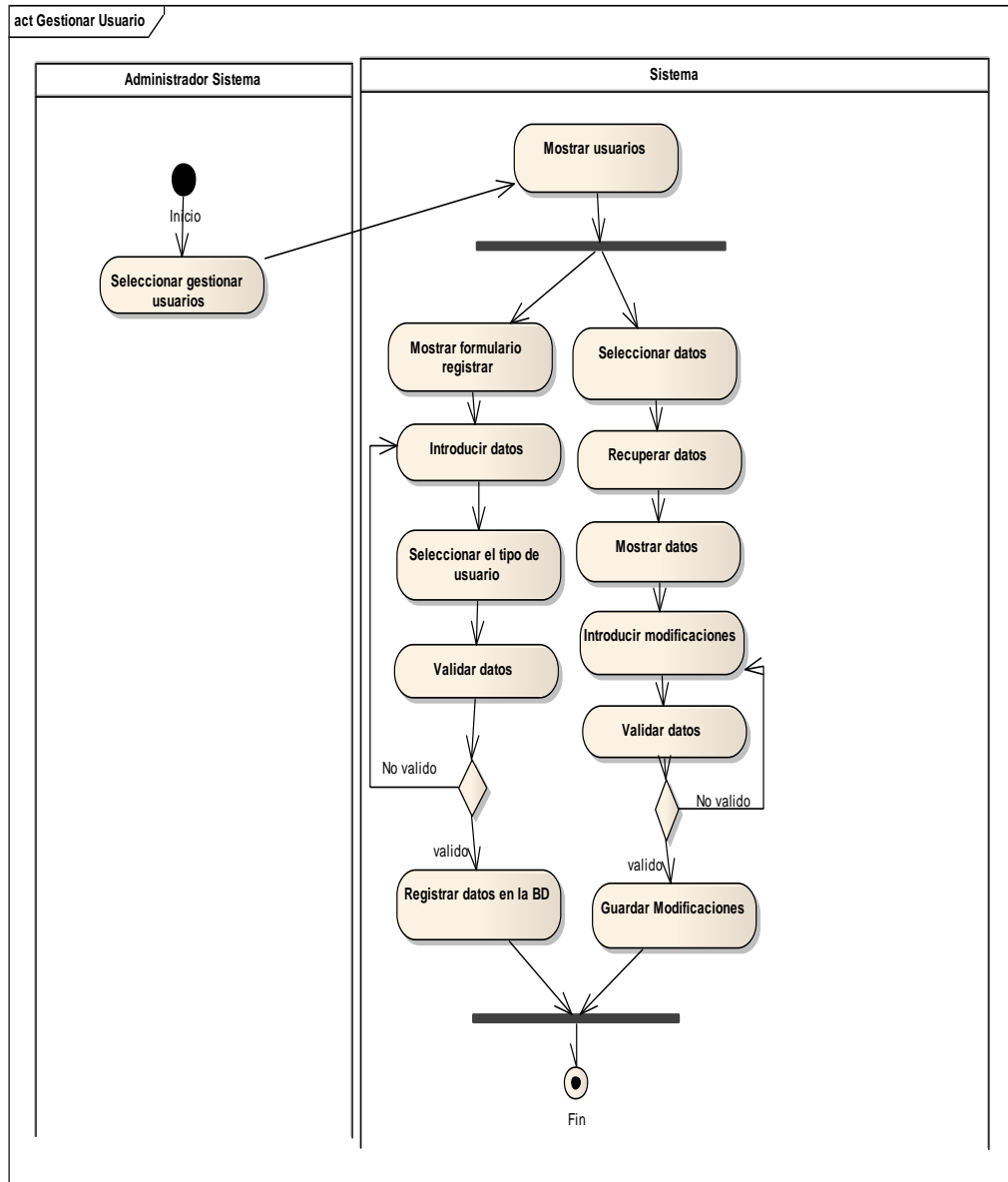
3.4.1 Diagrama de Actividades: Conectarse



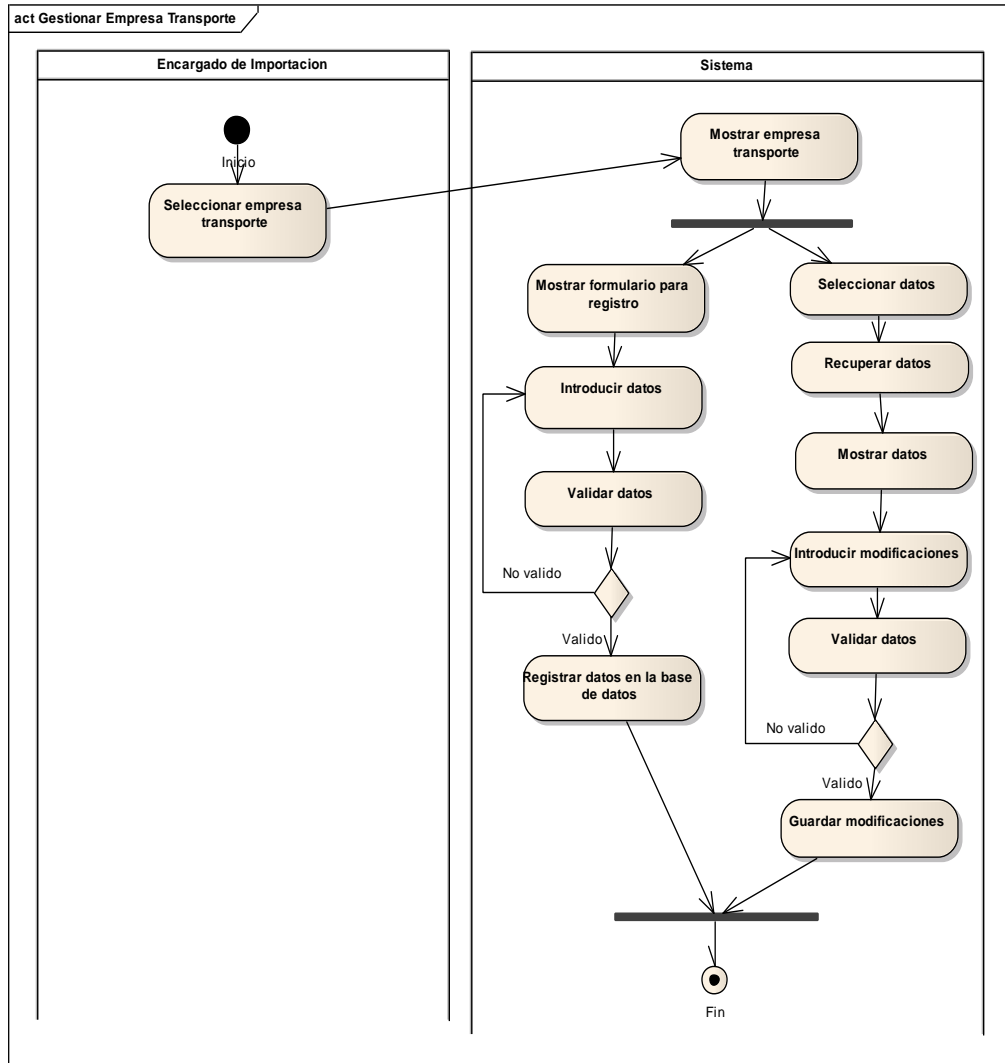
3.4.2 Diagrama de Actividades: Gestionar Empresa



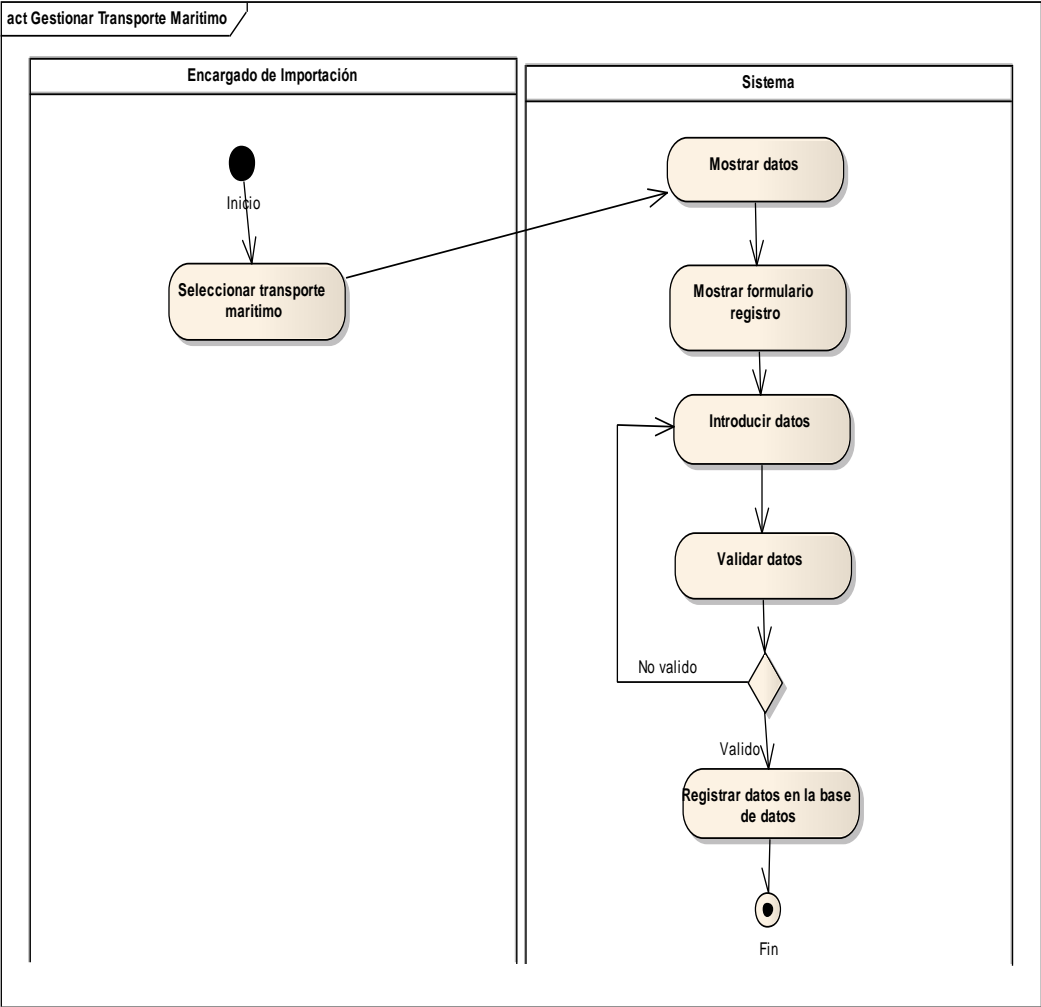
3.4.3 Diagrama de Actividades: Gestionar Usuarios



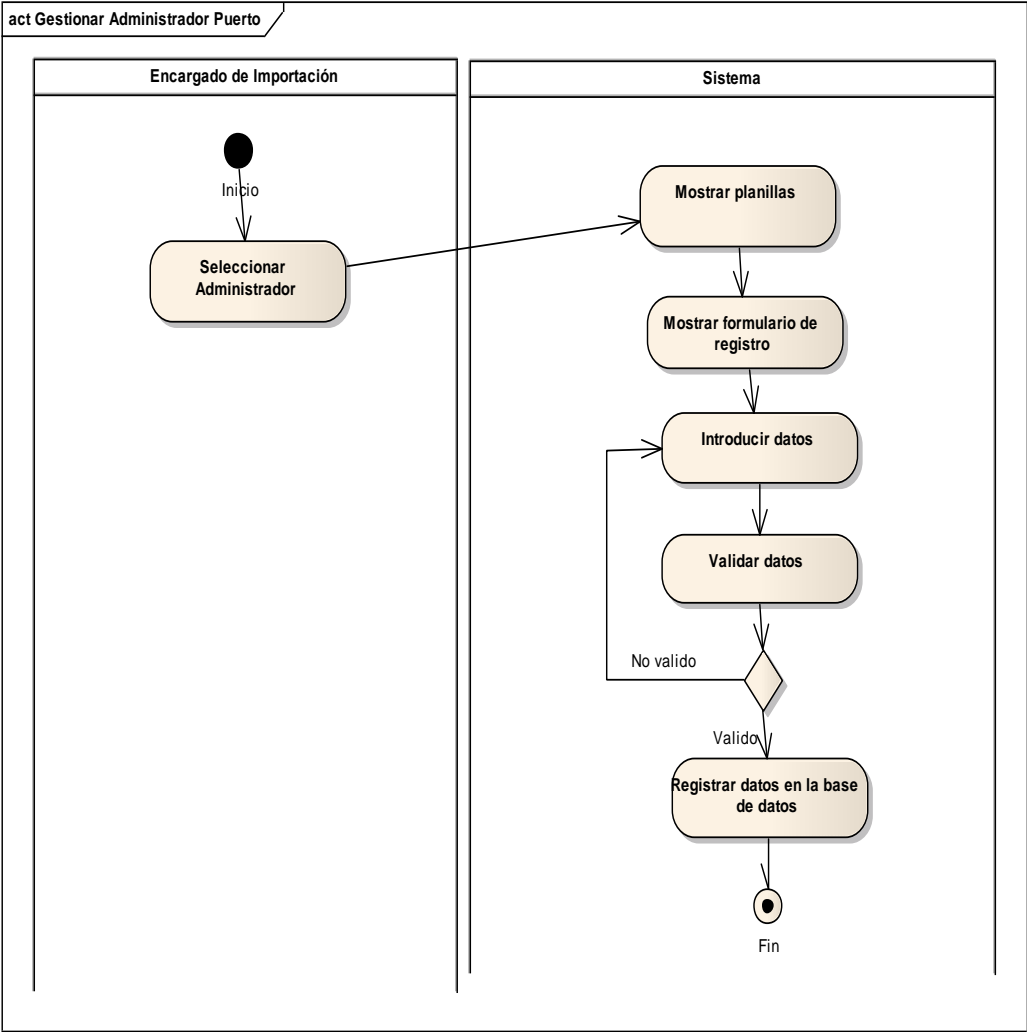
3.4.4 Diagrama de Actividades: Gestionar Empresa de Transporte Terrestre



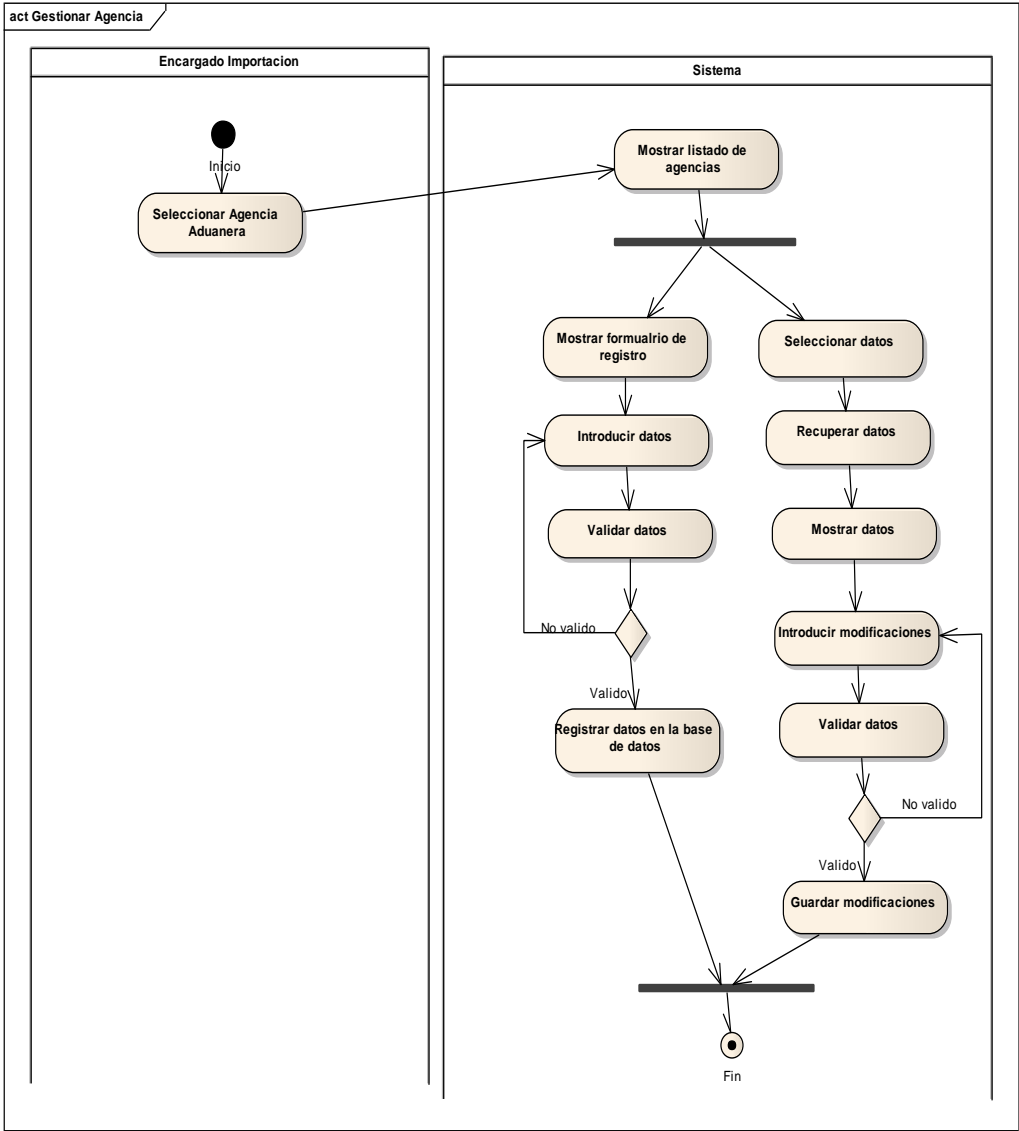
3.4.5 Diagrama de Actividades: Gestionar Transporte Marítimo



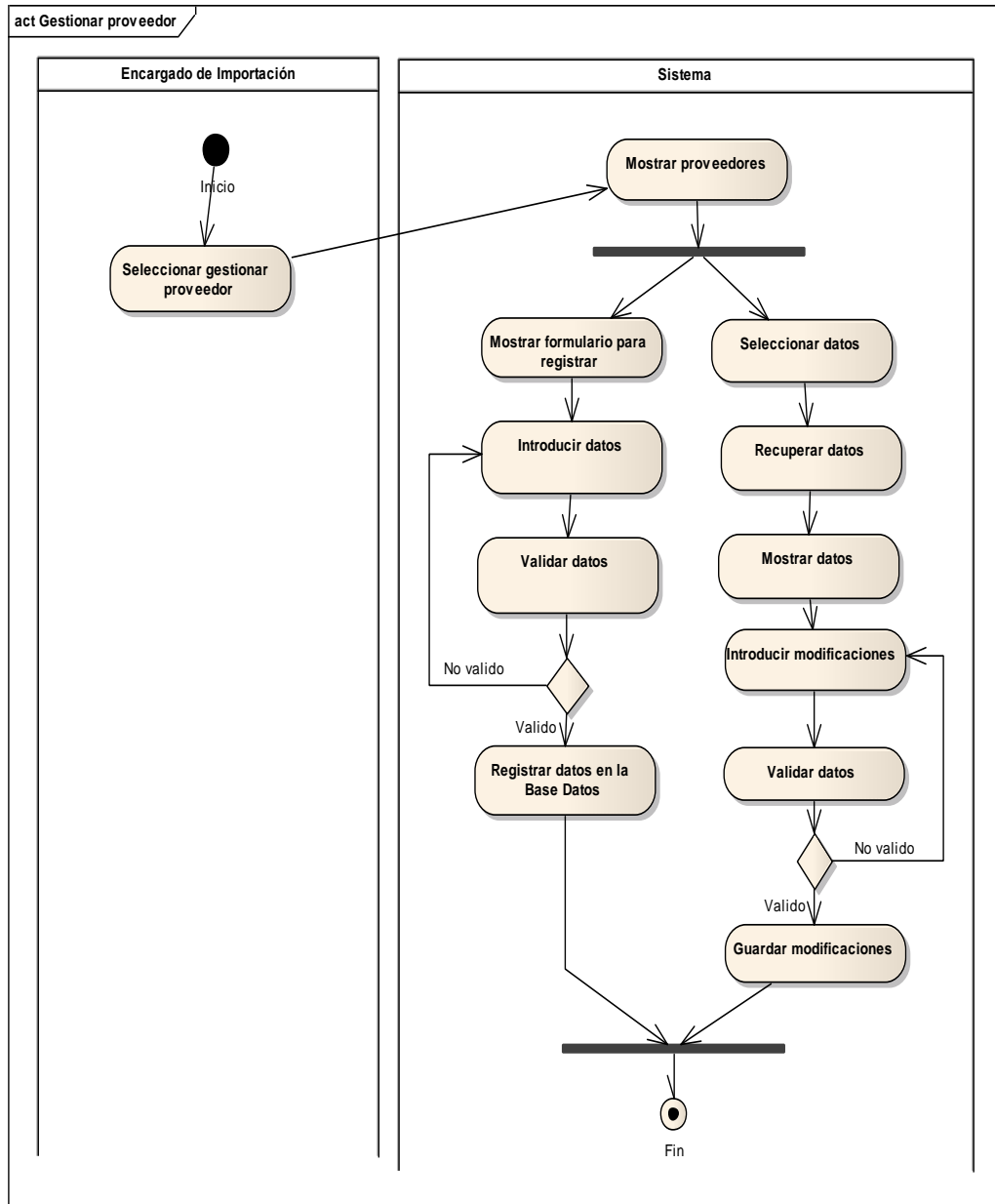
3.4.6 Diagrama de Actividades: Gestionar Administrador de Puerto



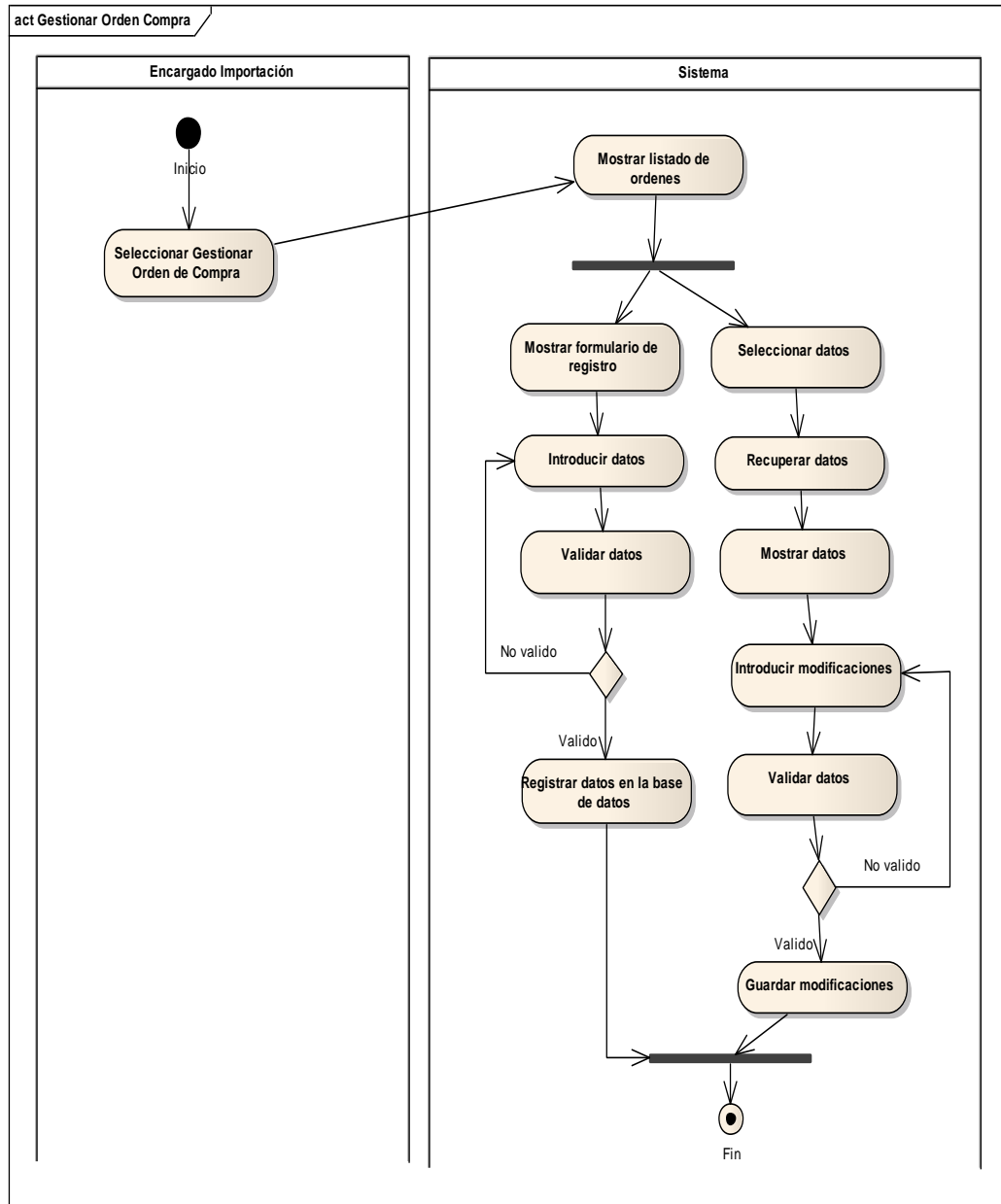
3.4.7 Diagrama de Actividades: Gestionar Agencia Aduanera



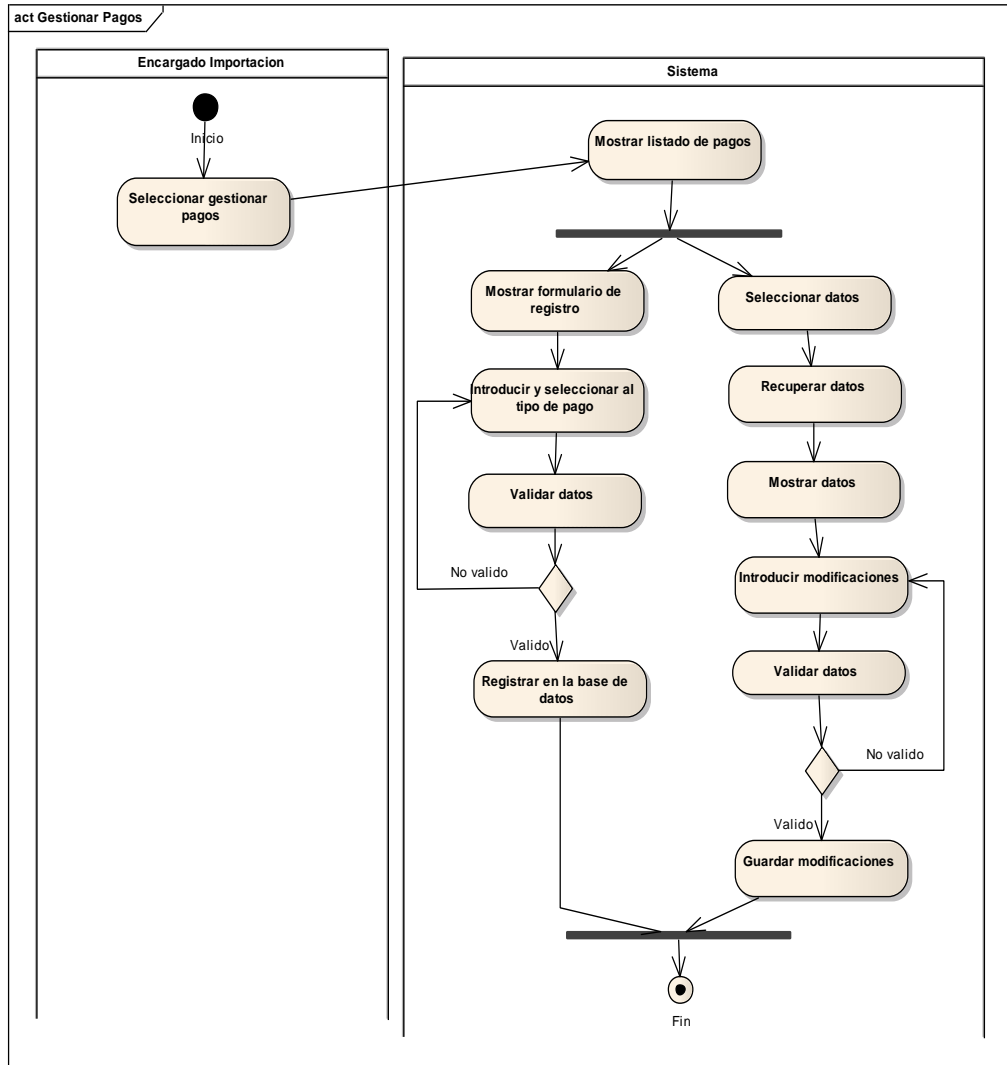
3.4.8 Diagrama de Actividades: Gestionar Proveedor



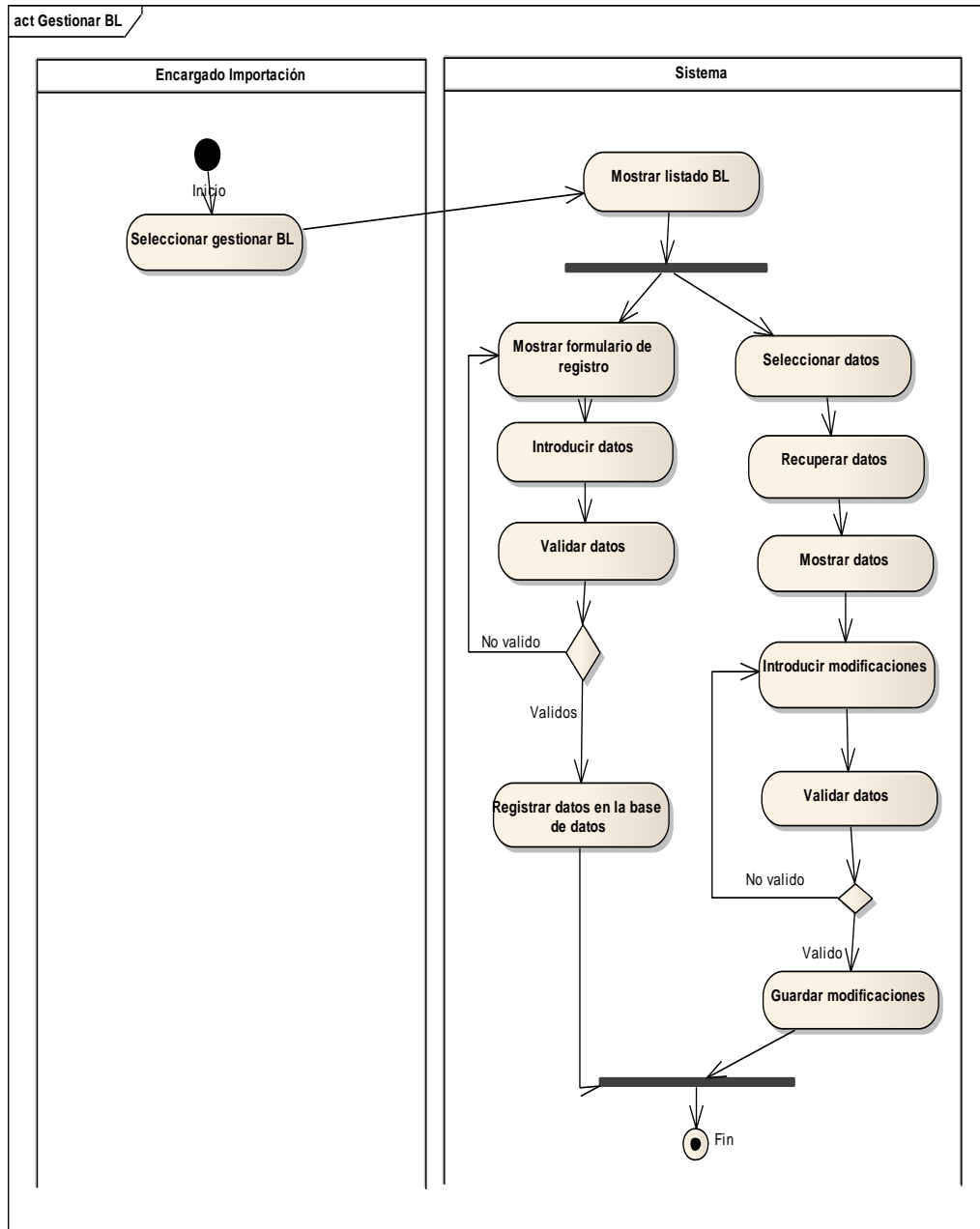
3.4.9 Diagrama de Actividades: Gestionar Orden de Compra



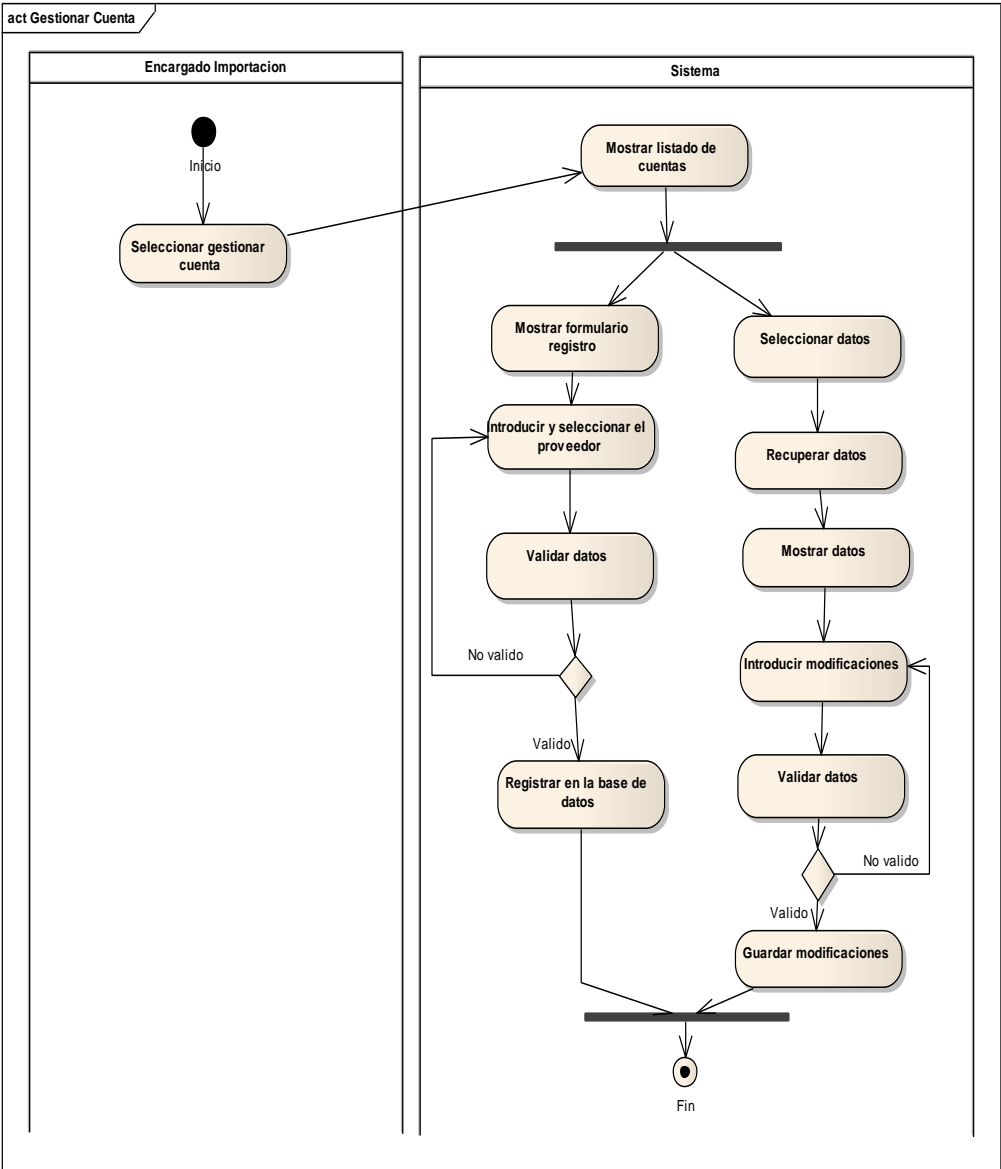
3.4.10 Diagrama de Actividades: Gestionar Pagos



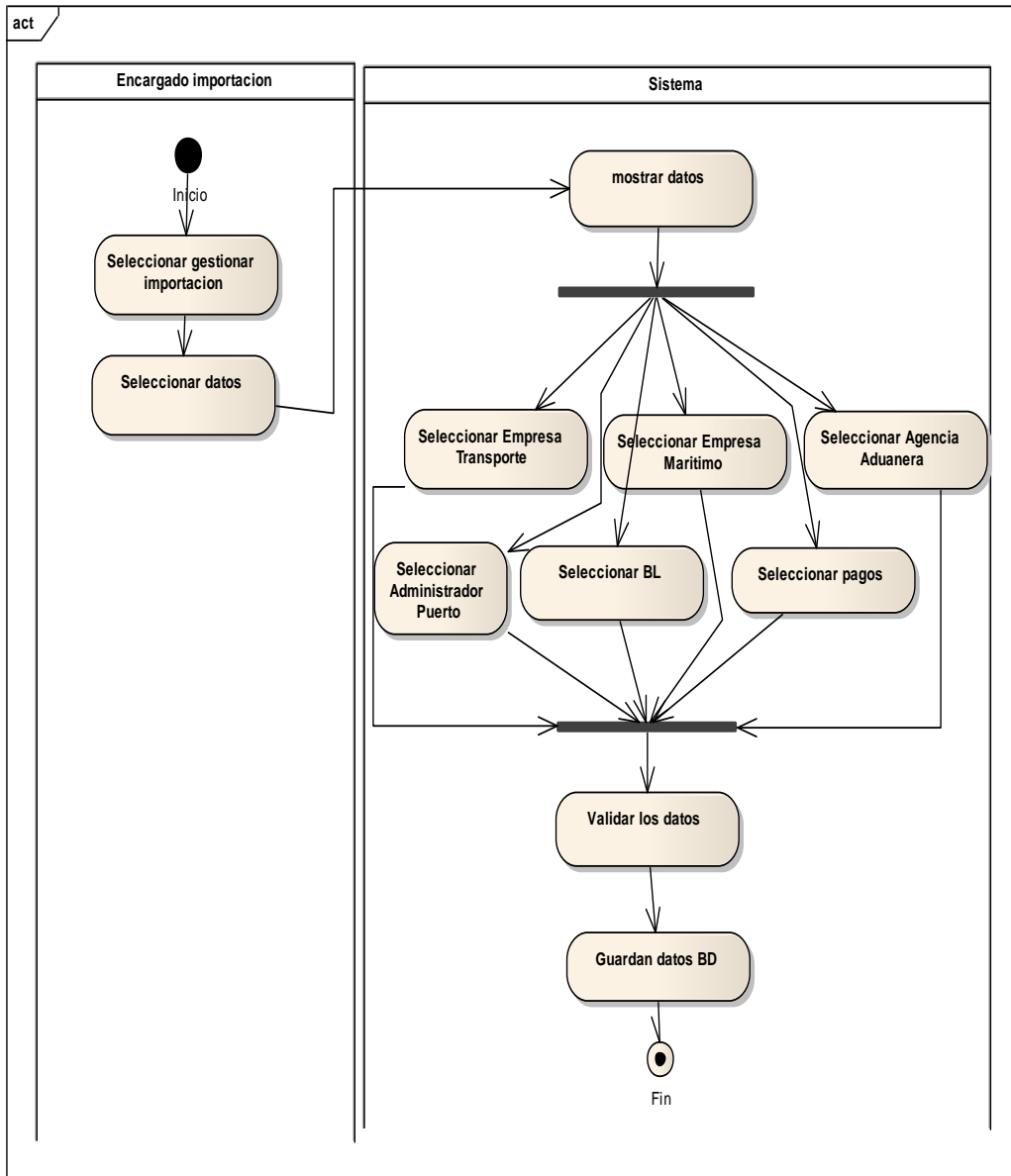
3.4.11 Diagrama de Actividades: Gestionar BL



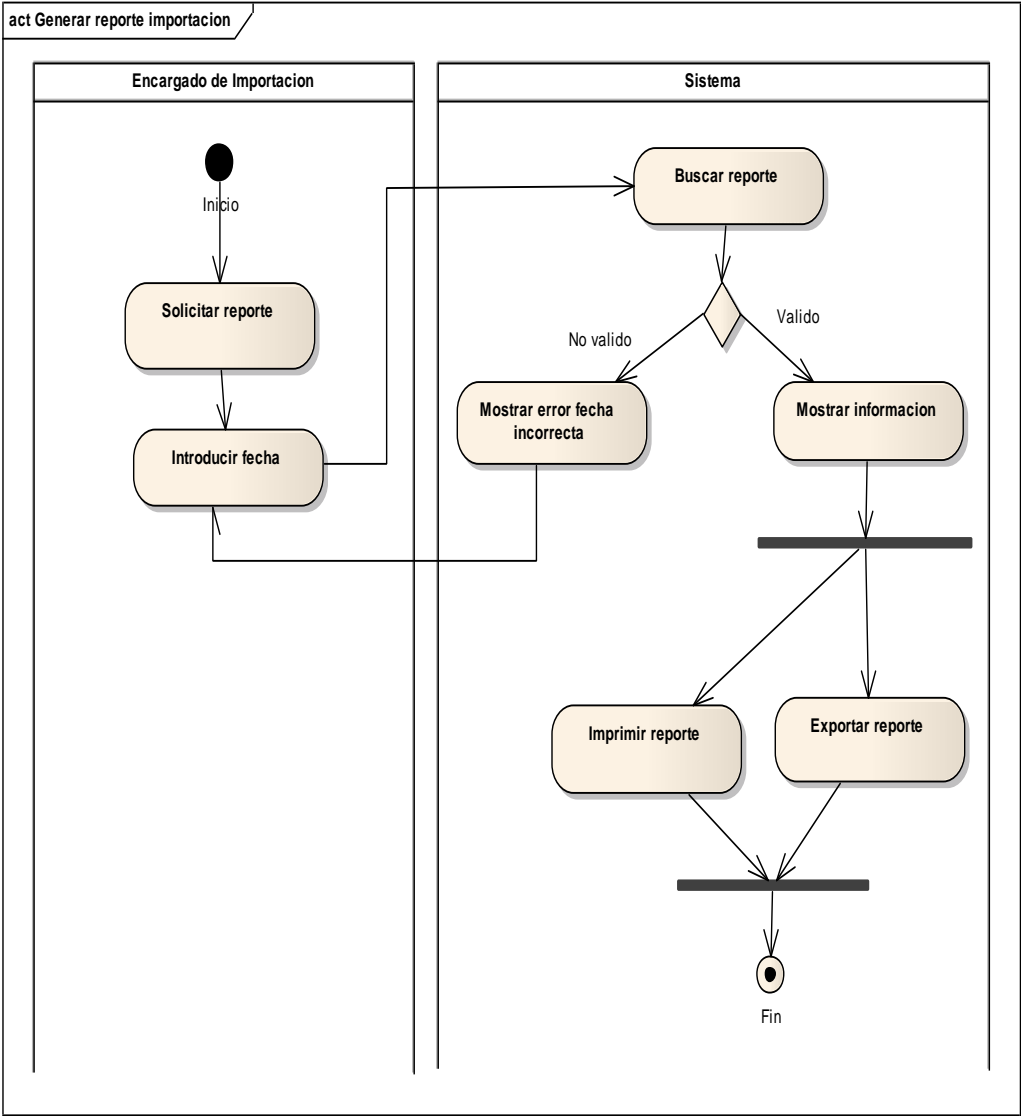
3.4.12 Diagrama de Actividades: Gestionar Cuentas



3.4.13 Diagrama de Actividades: Gestionar Importación

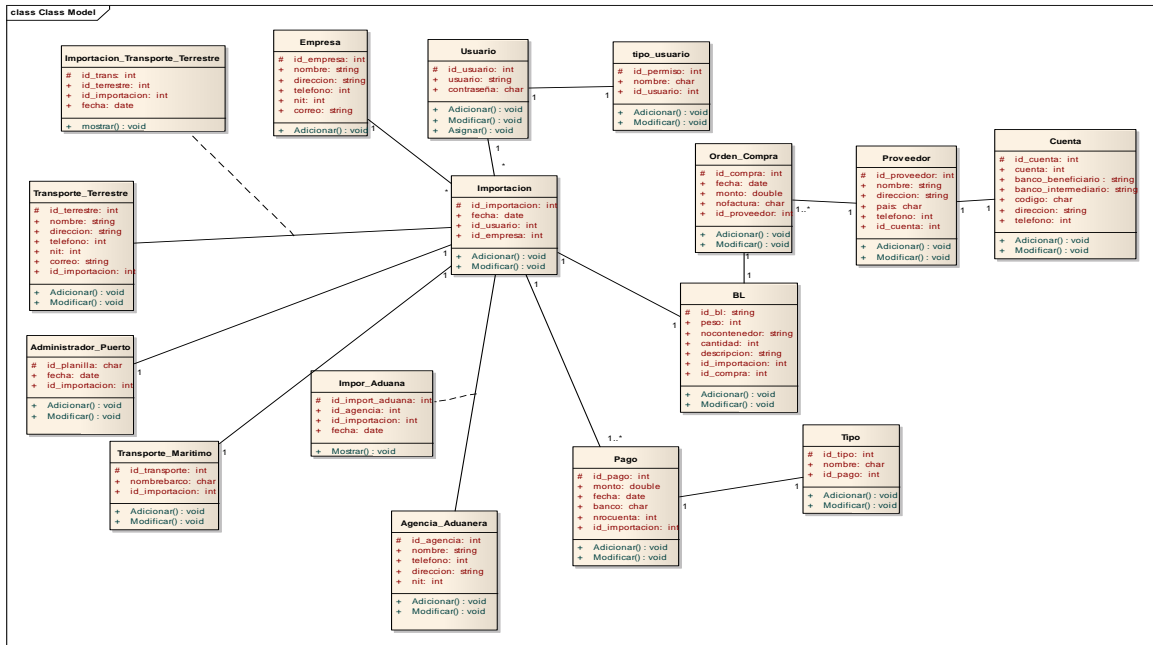


3.4.14 Diagrama de Actividades: Generar Reporte Importación

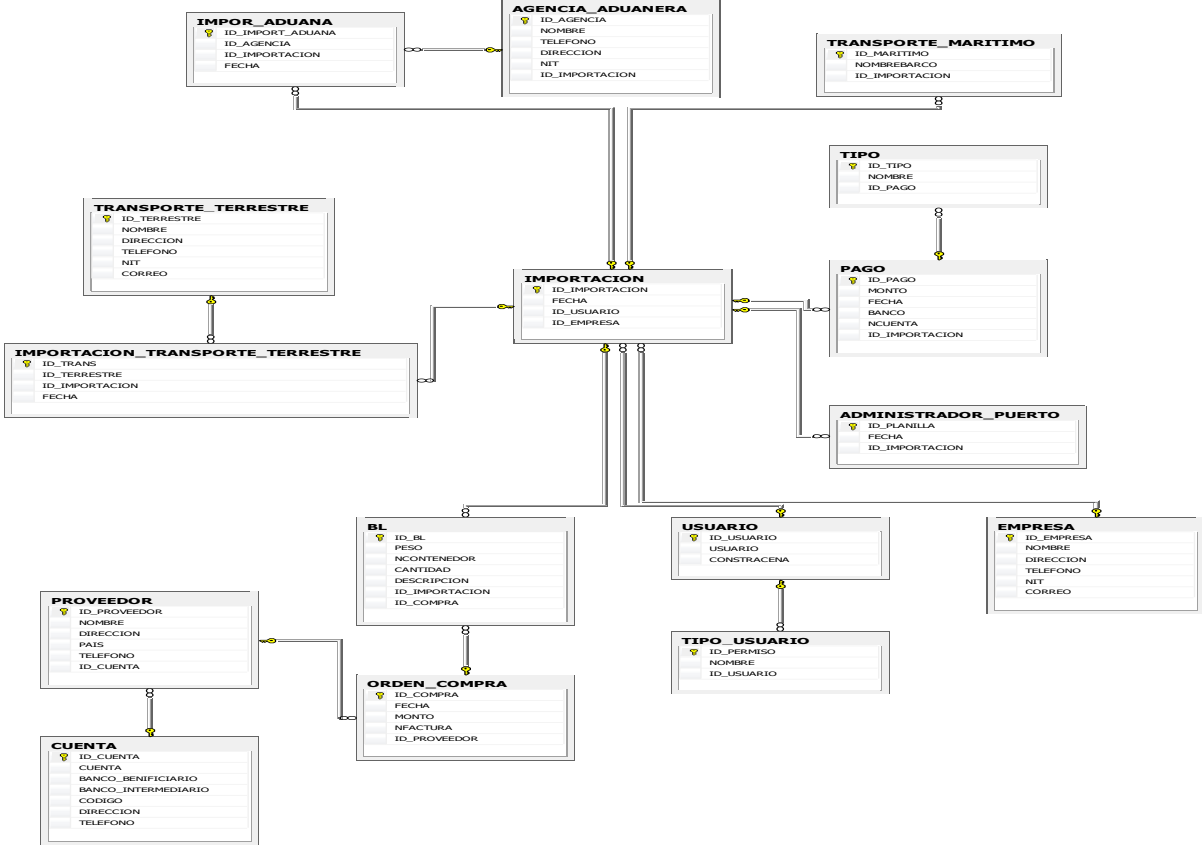


4. Descripción de la Base de Datos

4.1 Diagrama de Clases



4.2 Diseño de la Base Datos



4.3 Descripción de las tablas

Nombre de la Tabla		USUARIO			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_USUARIO	INT	YES	NO	NO
NO	USUARIO	VARCHAR(50)	NO	NO	NO
NO	CONSTRACENA	VARCHAR(50)	NO	NO	NO

Nombre de la Tabla		TIPO_USUARIO			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_PERMISO	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(10)	NO	NO	NO
NO	ID_USUARIO	INT	NO	NO	YES

Nombre de la Tabla		EMPRESA			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_EMPRESA	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(50)	NO	NO	NO
NO	DIRECCIÓN	VARCHAR(50)	NO	NO	NO
NO	TELÉFONO	INT	NO	NO	NO
NO	NIT	INT	NO	NO	NO
NO	CORREO	VARCHAR(50)	NO	NO	NO

Nombre de la Tabla		IMPORTACIÓN			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_IMPORTACIÓN	INT	YES	NO	NO
NO	FECHA	DATE	NO	NO	NO
NO	ID_USUARIO	INT	NO	NO	NO
NO	ID_EMPRESA	INT	NO	NO	YES

Nombre de la Tabla		TRANSPORTE_TERRESTRE			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_TERRESTRE	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(50)	NO	NO	NO
NO	DIRECCIÓN	VARCHAR(50)	NO	NO	NO
NO	TELÉFONO	INT	NO	NO	NO
NO	NIT	INT	NO	NO	NO
NO	CORREO	VARCHAR(50)	NO	NO	NO

Nombre de la Tabla		IMPORTACIÓN_TRANSPORTE_TERRESTRE			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_TRANS	INT	YES	NO	NO
NO	ID_TERRESTRE	INT	NO	NO	YES
NO	ID_IMPORTACIÓN	INT	NO	NO	YES
NO	FECHA	DATE	NO	NO	NO

Nombre de la Tabla		ADMINISTRADOR_PUERTO			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_PLANILLA	VARCHAR(10)	YES	NO	NO
NO	FECHA	DATE	NO	NO	NO
NO	ID_IMPORTACIÓN	INT	NO	NO	YES

Nombre de la Tabla		TRANSPORTE_MARÍTIMO			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_MARÍTIMO	INT	YES	NO	NO
NO	NOMBREBARCO	VARCHAR(50)	NO	NO	NO
NO	ID_IMPORTACIÓN	INT	NO	NO	YES

Nombre de la Tabla	AGENCIA_ADUANERA				
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_AGENCIA	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(50)	NO	NO	NO
NO	TELÉFONO	INT	NO	NO	NO
NO	DIRECCIÓN	VARCHAR(50)	NO	NO	NO
NO	NIT	INT	NO	NO	NO

Nombre de la Tabla	IMPOR_ADUANA				
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_IMPORT_ADUANA	INT	YES	NO	NO
NO	ID_AGENCIA	INT	NO	NO	YES
NO	ID_IMPORTACIÓN	INT	NO	NO	YES
NO	FECHA	DATE	NO	NO	NO

Nombre de la Tabla	PAGO				
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_PAGO	INT	YES	NO	NO
NO	MONTO	DECIMAL	NO	NO	NO
NO	FECHA	DATE	NO	NO	NO

NO	BANCO	VARCHAR(50)	NO	NO	NO
NO	NCUENTA	BIGINT	NO	NO	NO
NO	ID_IMPORTACIÓN	INT	NO	NO	YES

Nombre de la Tabla	TIPO				
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_TIPO	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(50)	NO	NO	NO
NO	ID_PAGO	INT	NO	NO	YES

Nombre de la Tabla	CUENTA				
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_CUENTA	INT	YES	NO	NO
NO	CUENTA	INT	NO	NO	NO
NO	BANCO_BENIFICIARIO	VARCHAR(50)	NO	NO	NO
NO	BANCO_INTERMEDIARIO	VARCHAR(50)	NO	NO	NO
NO	CODIGO	VARCHAR(10)	NO	NO	NO
NO	DIRECCIÓN	VARCHAR(100)	NO	NO	NO
NO	TELÉFONO	INT	NO	NO	NO

Nombre de la Tabla		PROVEEDOR			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_PROVEEDOR	INT	YES	NO	NO
NO	NOMBRE	VARCHAR(50)	NO	NO	NO
NO	DIRECCIÓN	VARCHAR(50)	NO	NO	NO
NO	PAIS	VARCHAR(50)	NO	NO	NO
NO	TELÉFONO	INT	NO	NO	NO
NO	ID_CUENTA	INT	NO	NO	YES

Nombre de la Tabla		ORDEN_COMPRA			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_COMPRA	INT	YES	NO	NO
NO	FECHA	DATE	NO	NO	NO
NO	MONTO	DECIMAL	NO	NO	NO
NO	NFACTURA	VARCHAR(10)	NO	NO	NO
NO	DESCRIPCIÓN	VARCHAR(100)	NO	NO	NO
NO	ID_PROVEEDOR	INT	NO	NO	YES

Nombre de la Tabla		BL			
Llave Primaria	Campos	Tipo de Dato	No Nulo	Único	Llave Foránea
YES	ID_BL	INT	YES	NO	NO
NO	PESO	INT	NO	NO	NO
NO	NCONTENEDOR	VARCHAR(50)	NO	NO	NO
NO	CANTIDAD	INT	NO	NO	NO
NO	DESCRIPCIÓN	VARCHAR(200)	NO	NO	NO
NO	ID_IMPORTACIÓN	INT	NO	NO	YES
NO	ID_COMPRA	INT	NO	NO	YES

4.4 Creación de la Base de Datos

```
CREATE DATABASE ADUANA
USE ADUANA
CREATE TABLE USUARIO(
    ID_USUARIO INT IDENTITY(1,1) NOT NULL,
    USUARIO VARCHAR(50) NOT NULL,
    CONSTRACENA VARCHAR(10) NOT NULL,
    CONSTRAINT PK_USUARIO PRIMARY KEY(ID_USUARIO)
)
CREATE TABLE TIPO_USUARIO(
    ID_PERMISO INT IDENTITY(1,1) NOT NULL,
    NOMBRE VARCHAR(10) NOT NULL, /*VERIFICAR*/
    ID_USUARIO INT NOT NULL,
    CONSTRAINT PK_TIPO_USUARIO PRIMARY KEY(ID_PERMISO),
    CONSTRAINT FK_TIPO_USU_USUARIO FOREIGN KEY(ID_USUARIO)
    REFERENCES USUARIO(ID_USUARIO) ON DELETE CASCADE ON
UPDATE CASCADE
)
CREATE TABLE EMPRESA(
    ID_EMPRESA INT IDENTITY(1,1) NOT NULL,
    NOMBRE VARCHAR(50) NOT NULL,
    DIRECCION VARCHAR(100) NOT NULL,
    TELEFONO INT NOT NULL,
    NIT INT NOT NULL,
    CORREO VARCHAR(50) NOT NULL,
    CONSTRAINT PK_EMPRESA PRIMARY KEY(ID_EMPRESA)
)
CREATE TABLE IMPORTACION(
    ID_IMPORTACION INT IDENTITY(1,1) NOT NULL,
```

```

FECHA DATE NOT NULL,
ID_USUARIO INT NOT NULL,
ID_EMPRESA INT NOT NULL,
CONSTRAINT PK_IMPORTACION PRIMARY KEY(ID_IMPORTACION),
CONSTRAINT FK_IMPORTACION_USUARIO FOREIGN KEY(ID_USUARIO)
REFERENCES USUARIO(ID_USUARIO) ON DELETE CASCADE ON
UPDATE CASCADE,
CONSTRAINT FK_IMPORTACION_EMPRESA FOREIGN
KEY(ID_EMPRESA)
REFERENCES EMPRESA(ID_EMPRESA) ON DELETE CASCADE ON
UPDATE CASCADE
)
CREATE TABLE TRANSPORTE_TERRESTRE(
ID_TERRESTRE INT IDENTITY(1,1) NOT NULL,
NOMBRE VARCHAR(50) NOT NULL,
DIRECCION VARCHAR(100) NOT NULL,
TELEFONO INT NOT NULL,
NIT INT NOT NULL,
CORREO VARCHAR(50) NOT NULL,
CONSTRAINT PK_TRANSPORTE_TERRESTRE PRIMARY
KEY(ID_TERRESTRE)
)
CREATE TABLE IMPORTACION_TRANSPORTE_TERRESTRE
(
ID_TRANS INT IDENTITY(1,1) NOT NULL,
ID_TERRESTRE INT NOT NULL,
ID_IMPORTACION INT NOT NULL,
FECHA DATE NOT NULL,
CONSTRAINT PK_IMPORTACION_TRANSPORTE_TERRESTRE PRIMARY
KEY(ID_TRANS),

```



```

        CONSTRAINT FK_IMPORTACION_TRANSPORTE_TERRESTRE FOREIGN
KEY(ID_TERRESTRE)
        REFERENCES TRANSPORTE_TERRESTRE(ID_TERRESTRE) ON DELETE
CASCADE ON UPDATE CASCADE,
        CONSTRAINT FK_TRANSPORTE_TERRESTRE_IMPORTACION FOREIGN
KEY(ID_IMPORTACION)
        REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE
)
CREATE TABLE ADMINISTRADOR_PUERTO(
        ID_PLANILLA VARCHAR(10) NOT NULL,/*VERIFICAR*/
        FECHA DATE NOT NULL,
        ID_IMPORTACION INT,
        CONSTRAINT PK_ADMINISTRADOR_PUERTO PRIMARY
KEY(ID_PLANILLA),
        CONSTRAINT FK_ADMINISTRADOR_PUERTO FOREIGN
KEY(ID_IMPORTACION)
        REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE
)
CREATE TABLE TRANSPORTE_MARITIMO(
        ID_MARITIMO INT IDENTITY(1,1) NOT NULL,
        NOMBREBARCO VARCHAR(50) NOT NULL,
        ID_IMPORTACION INT,
        CONSTRAINT PK_TRANSPORTE_MARITIMO PRIMARY
KEY(ID_MARITIMO),
        CONSTRAINT FK_TRANSPORTE_MARITIMO_IMPORTACION FOREIGN
KEY(ID_IMPORTACION)
        REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE
)

```

```

CREATE TABLE AGENCIA_ADUANERA(
    ID_AGENCIA INT IDENTITY(1,1) NOT NULL,
    NOMBRE VARCHAR(50) NOT NULL,
    TELEFONO INT NOT NULL,
    DIRECCION VARCHAR(100) NOT NULL,
    NIT INT NOT NULL,
    CONSTRAINT PK_AGENCIA_ADUANERA PRIMARY KEY(ID_AGENCIA)
)

CREATE TABLE IMPOR_ADUANA(
    ID_IMPORT_ADUANA INT IDENTITY(1,1) NOT NULL,
    ID_AGENCIA INT NOT NULL,
    ID_IMPORTACION INT NOT NULL,
    FECHA DATE NOT NULL,
    CONSTRAINT PK_IMPORT_ADUANA PRIMARY
KEY(ID_IMPORT_ADUANA),
    CONSTRAINT FK_IMPORT_ADUANA_AGENCIA_ADUANERA FOREIGN
KEY(ID_AGENCIA)
    REFERENCES AGENCIA_ADUANERA(ID_AGENCIA) ON DELETE
CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_IMPORT_ADUANA_IMPORTACION FOREIGN
KEY(ID_IMPORTACION)
    REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE
)

CREATE TABLE PAGO(
    ID_PAGO INT IDENTITY(1,1) NOT NULL,
    MONTO DECIMAL(10,2) NOT NULL,
    FECHA DATE NOT NULL,
    BANCO VARCHAR(50) NOT NULL,
    NCUENTA BIGINT NOT NULL,
    ID_IMPORTACION INT,

```

```

        CONSTRAINT PK_PAGO PRIMARY KEY(ID_PAGO),
        CONSTRAINT FK_PAGO_IMPORTACION FOREIGN
KEY(ID_IMPORTACION)
        REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE
    )
CREATE TABLE TIPO(
    ID_TIPO INT IDENTITY(1,1) NOT NULL,
    NOMBRE VARCHAR(50) NOT NULL,
    ID_PAGO INT NOT NULL,
    CONSTRAINT PK_TIPO PRIMARY KEY(ID_TIPO),
    CONSTRAINT FK_TIPO_PAGO FOREIGN KEY(ID_PAGO)
REFERENCES PAGO(ID_PAGO) ON DELETE CASCADE ON UPDATE
CASCADE
)
CREATE TABLE CUENTA(
    ID_CUENTA INT IDENTITY(1,1) NOT NULL,
    CUENTA INT NOT NULL,
    BANCO_BENIFICIARIO VARCHAR(50) NOT NULL,
    BANCO_INTERMEDIARIO VARCHAR(50) NOT NULL,
    CODIGO VARCHAR(10) NOT NULL,
    DIRECCION VARCHAR(100) NOT NULL,
    TELEFONO INT NOT NULL,
    CONSTRAINT PK_CUENTA PRIMARY KEY(ID_CUENTA)
)
CREATE TABLE PROVEEDOR(
    ID_PROVEEDOR INT IDENTITY(1,1) NOT NULL,
    NOMBRE VARCHAR(50) NOT NULL,
    DIRECCION VARCHAR(100) NOT NULL,
    PAIS VARCHAR(50) NOT NULL,
    TELEFONO INT NOT NULL,

```

```

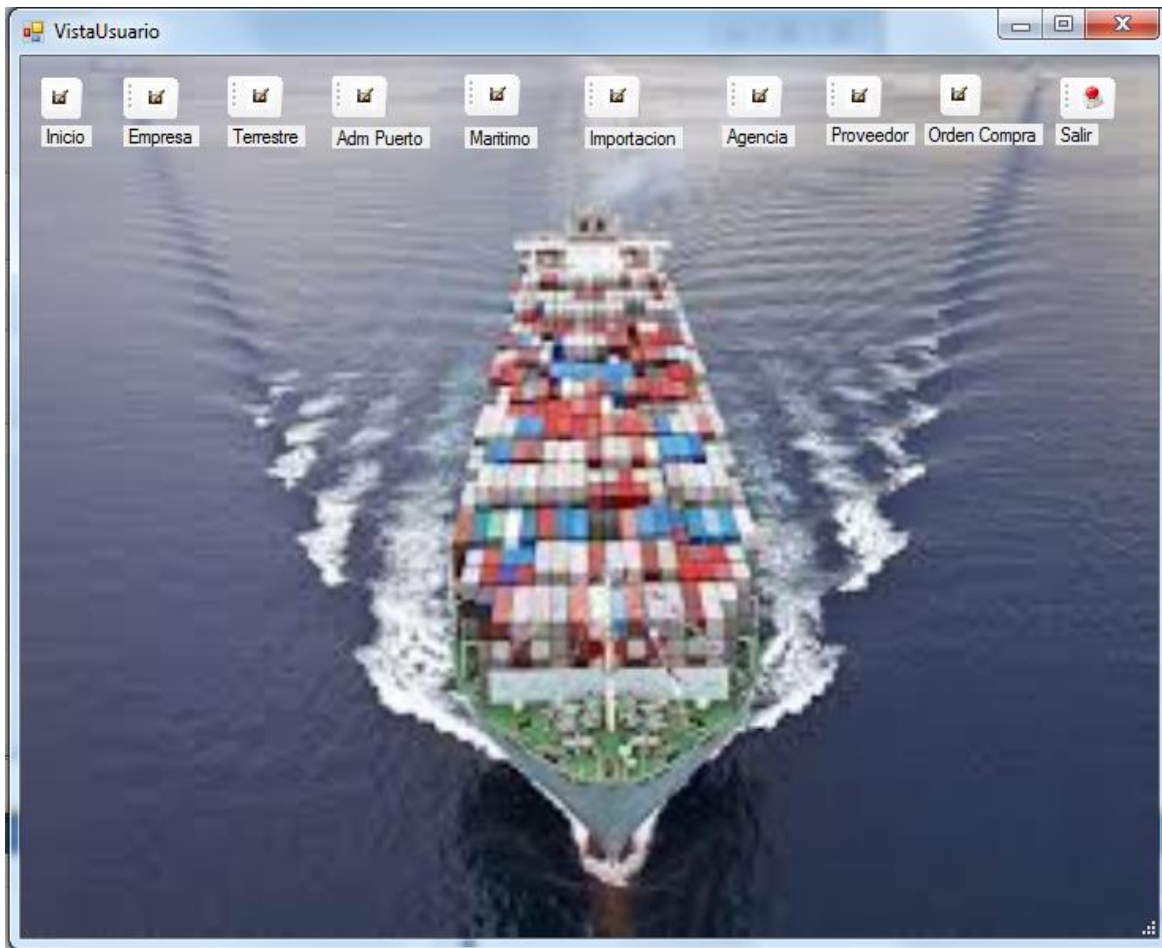
ID_CUENTA INT NOT NULL,
CONSTRAINT PK_PROVEEDOR PRIMARY KEY(ID_PROVEEDOR),
CONSTRAINT FK_PROVEEDOR_CUENTA FOREIGN KEY(ID_CUENTA)
REFERENCES CUENTA(ID_CUENTA) ON DELETE CASCADE ON UPDATE
CASCADE
)
CREATE TABLE ORDEN_COMPRA(
    ID_COMPRA INT IDENTITY(1,1) NOT NULL,
    FECHA DATE NOT NULL,
    MONTO DECIMAL(10,2) NOT NULL,
    NFACTURA VARCHAR(10) NOT NULL,
    ID_PROVEEDOR INT NOT NULL,
    DESCRIPCION VARCHAR(100) NOT NULL,
    CONSTRAINT PK_ORDEN_COMPRA PRIMARY KEY(ID_COMPRA),
    CONSTRAINT FK_ORDEN_COMPRA_PROVEEDOR FOREIGN
KEY(ID_PROVEEDOR)
REFERENCES PROVEEDOR(ID_PROVEEDOR) ON DELETE CASCADE ON
UPDATE CASCADE
)
CREATE TABLE BL(
    ID_BL INT IDENTITY(1,1) NOT NULL,
    PESO INT NOT NULL,
    NCONTENEDOR VARCHAR(50) NOT NULL,
    CANTIDAD INT NOT NULL,
    DESCRIPCION VARCHAR(200) NOT NULL,
    ID_IMPORTACION INT NOT NULL,
    ID_COMPRA INT NOT NULL,
    CONSTRAINT PK_BL PRIMARY KEY(ID_BL),
    CONSTRAINT FK_BL_IMPORTACION FOREIGN KEY(ID_IMPORTACION)
REFERENCES IMPORTACION(ID_IMPORTACION) ON DELETE CASCADE
ON UPDATE CASCADE,

```

```
CONSTRAINT FK_BL_ORDEN_COMPRA FOREIGN KEY(ID_COMPRA)
REFERENCES ORDEN_COMPRA(ID_COMPRA) ON DELETE CASCADE ON
UPDATE CASCADE
)
```

4.5 Diseño de Interfaces

Pantalla n°1: Pantalla Principal



Pantalla donde se muestra los diferentes módulos para poder navegar dependiendo el tipo de usuario.

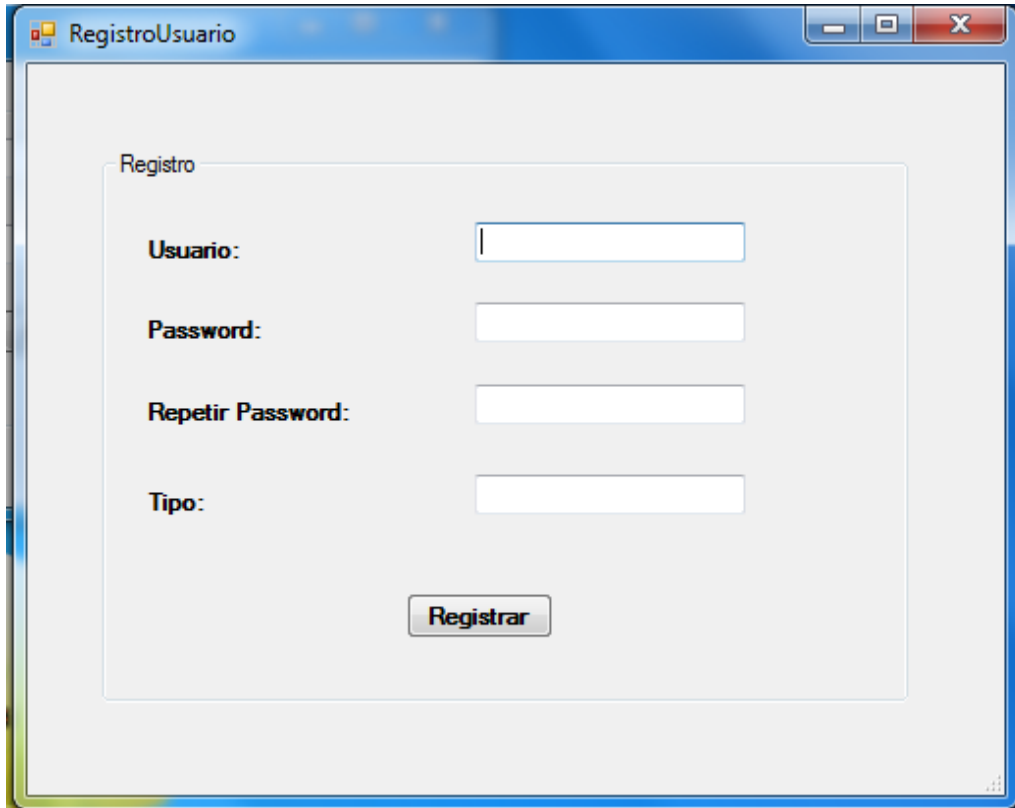
Pantalla n°2: Login y Password



The image shows a screenshot of a Windows-style window titled "Form1". Inside the window, there is a login form. On the left side of the form, there is an icon of a man in a suit holding a large golden key. To the right of the icon, there are two input fields. The first is labeled "Usuario:" and contains the text "admin". The second is labeled "Contraseña:" and contains six asterisks "*****". Below these fields is a button labeled "Ingresar". At the bottom right of the form area, there is a blue hyperlink that reads "Registrar Nuevo Usuario".

Pantalla donde se muestra el formulario para la autenticación del usuario para verificar su existencia.

Pantalla n° 3: Gestionar Usuario

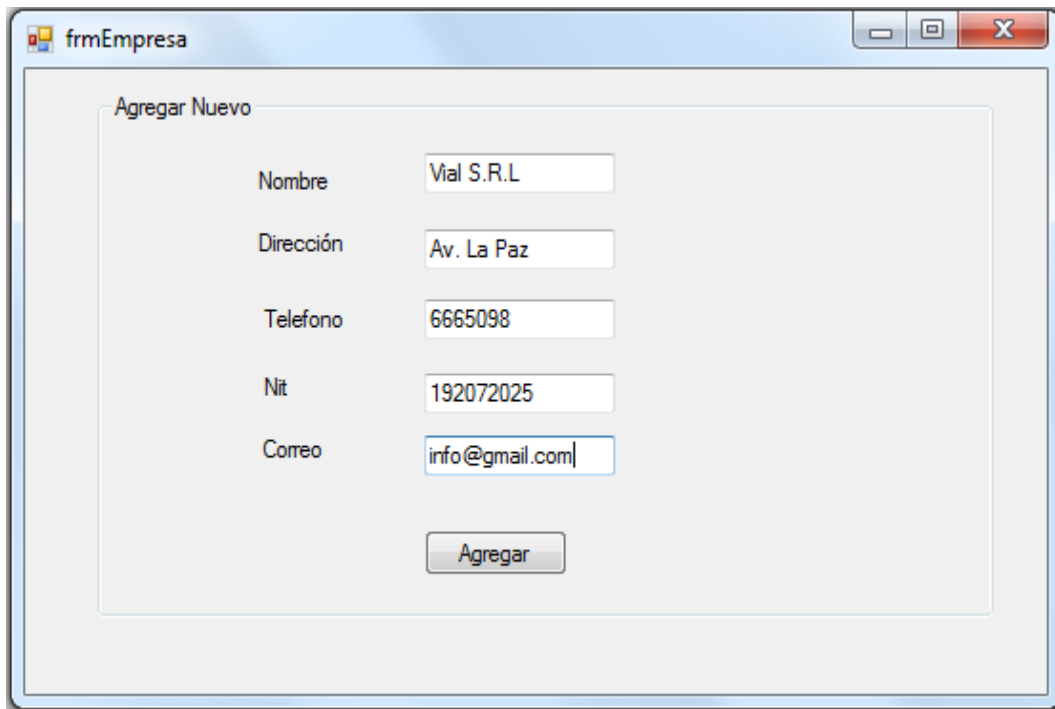


The image shows a software window titled "RegistroUsuario" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a registration form with the following fields and a button:

- Registro** (Section header)
- Usuario:** Text input field
- Password:** Text input field
- Repetir Password:** Text input field
- Tipo:** Text input field
- Registrar** (Submit button)

Pantalla que permite registrar un usuario introduciendo sus datos personales.

Pantalla n°4: Gestionar Empresa



The screenshot shows a window titled "frmEmpresa" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a form titled "Agregar Nuevo" (Add New). The form contains five text input fields, each with a label to its left:

- Nombre: Vial S.R.L.
- Dirección: Av. La Paz
- Telefono: 6665098
- Nit: 192072025
- Correo: info@gmail.com|

Below the input fields is a button labeled "Agregar".

Pantalla que permite registrar una empresa y llenar los datos de la misma.

Pantalla n°5: Gestionar Empresa de Transporte

frmtransporteTerrestre

Agregar Nuevo

Nombre: COTRIBER

Dirección: Av Panamericana

Telefono: 6961999

Nit: 192072015

Correo: cotriber@gmail.com

Agregar

Modificar

	ECCION	TELEFONO	NIT	CORREO
▶	anamericana	6961999	192072025	cotriber@
*				

Pantalla que permite ingresar los datos de la empresa de transporte terrestre encargada de traer la mercancía.

Pantalla n°6: Gestionar Administrador de Puerto

Nuevo

Planilla: 1511234TJA

Fecha: lunes, 07 de diciembre de 2015

Insertar

Modificar

Planilla:

	PLANILLA	FECHA
▶	1511234TJA	07/12/20...
*		

Pantalla donde se registra la planilla que se genera cuando el transporte marítimo llega a puerto de Bolivia.

Pantalla n°7: Gestionar Transporte Marítimo

frmTransporteMaritimo

Agregar

Barco

VESSEL SANTA BARBARA

Insertar

Nombre

ID	Nombre Barco
1	VESSEL SANTA ...
*	

Pantalla donde se registra los datos del barco que realizó el transporte marítimo de la mercancía.

Pantalla n°8: Gestionar Agencia Aduanera

frmAgenciaAduanera

Agregar

Nombre: Agencia Oriental

Dirección: Av Panamericana

Telefono: 2234786

Nit: 132178023

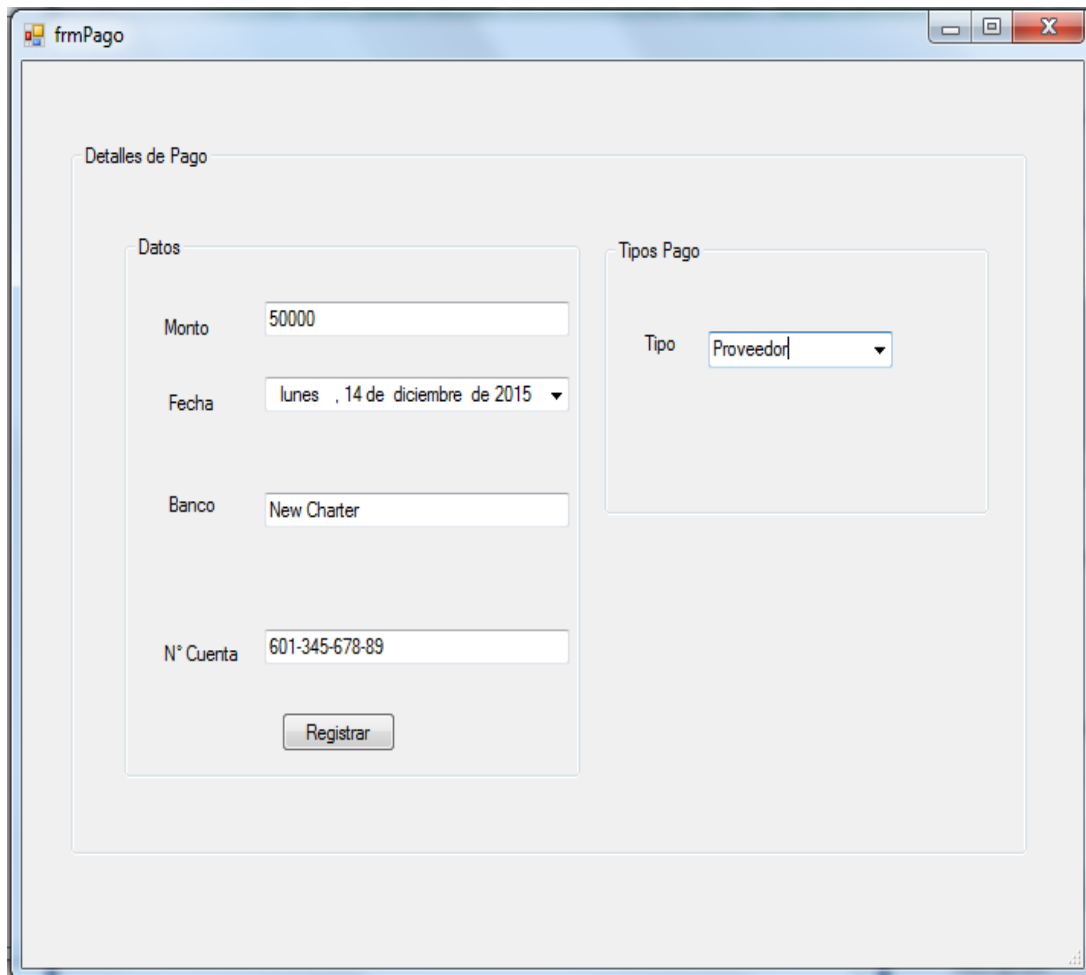
Modificar

Nombre:

	Nombre	Direccion
▶	Agencia Oriental	Av Panamericana
*		

Pantalla donde se registra los datos de la agencia aduanera que realizó los tramites de aduana para legalizar la mercadería.

Pantalla n°9: Gestionar Pagos

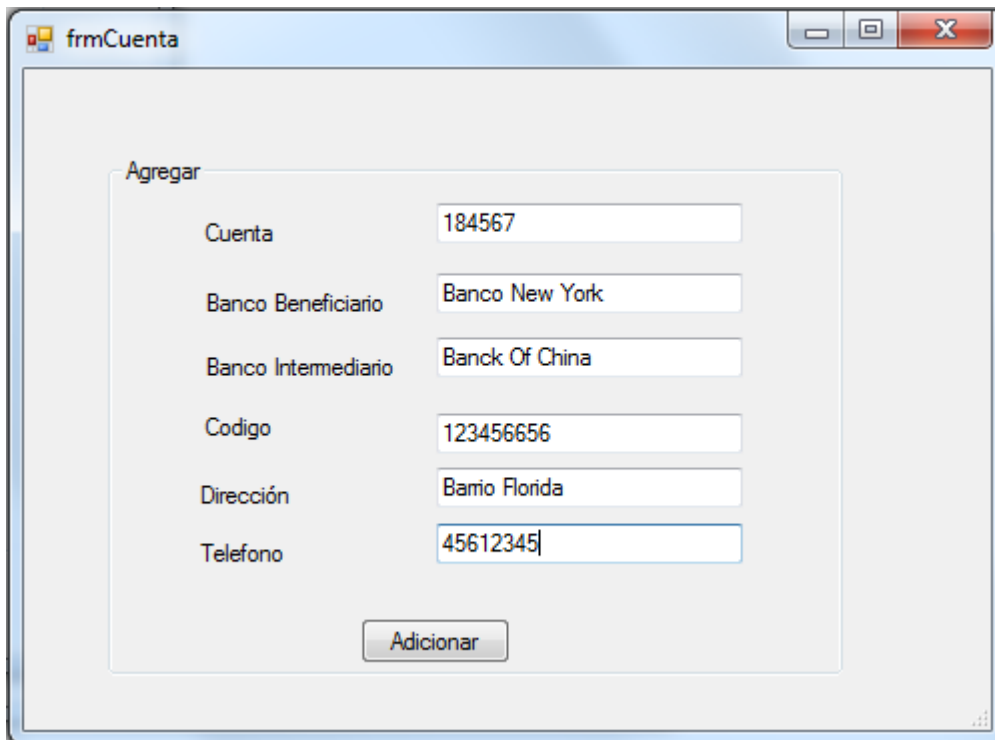


The screenshot shows a Windows-style application window titled 'frmPago'. Inside the window, there is a section titled 'Detalles de Pago'. This section is divided into two main areas: 'Datos' and 'Tipos Pago'. The 'Datos' area contains four text input fields: 'Monto' with the value '50000', 'Fecha' with a date picker showing 'lunes , 14 de diciembre de 2015', 'Banco' with the value 'New Charter', and 'N° Cuenta' with the value '601-345-678-89'. Below these fields is a 'Registrar' button. The 'Tipos Pago' area contains a 'Tipo' dropdown menu with 'Proveedor' selected.

Detalles de Pago	
Datos	
Monto	50000
Fecha	lunes , 14 de diciembre de 2015
Banco	New Charter
N° Cuenta	601-345-678-89
<input type="button" value="Registrar"/>	
Tipos Pago	
Tipo	Proveedor

Pantalla donde se registra los pagos a proveedores, agencia aduaneras, empresa de transporte terrestre, empresas de transporte marítimo de una importación.

Pantalla n°10: Gestionar Cuenta



The screenshot shows a window titled 'frmCuenta' with a standard Windows-style title bar. Inside the window, there is a form titled 'Agregar' (Add) with the following fields and values:

Label	Value
Cuenta	184567
Banco Beneficiario	Banco New York
Banco Intermediario	Bank Of China
Codigo	123456656
Dirección	Barrio Florida
Telefono	45612345

At the bottom of the form is a button labeled 'Adicionar'.

Pantalla donde se registra las cuentas para realizar los giros o transferencias a los proveedores.

Pantalla n°11: Gestionar Proveedor

The screenshot shows a software window titled 'frmProveedor'. It is divided into two main sections: 'Agregar' (Add) and 'Modificar' (Modify).

Agregar Section:

- Nombre: TRIMAR SRL
- Direccion: Av Londres
- Pais: Ecuador
- Telefono: 35446789
- Cuenta: 184567 (dropdown menu)
- Adicionar button
- [Agregar Cuenta](#) link

Modificar Section:

- Nombre: [Empty text box]
- Table with columns: N°, NOMBR, DIRECCION, TELEFO
- Table content:

N°	NOMBR	DIRECCION	TELEFO
2	TRIM...	Av Londres	3544678
*			

Pantalla donde se registra los datos de los proveedores de mercadería a la empresa.

Pantalla n°12: Gestionar Orden de Compra

The screenshot shows a software window titled 'frmOrden'. It is divided into two main sections: 'Agregar' (Add) on the left and 'Modificar' (Modify) on the right.

Agregar Section:

- Fecha: martes , 08 de diciembre de 2015 (dropdown menu)
- Monto: 45000,00 (text input)
- N° Factura: 3456 (text input)
- Descripcion: Giferia Mezclada (text input)
- Proveedor: TRIMAR SRL (dropdown menu)
- Adicionar (button)

Modificar Section:

- N° Factura (text input)
- Table with columns: ID, FECHA, MONTO

	ID	FECHA	MONTO
▶	2	08/12/2015 12:0...	45000,00
*			

Pantalla donde se registra los datos la orden de compra detalle de la mercadería a importar.

Pantalla n°13: Gestionar Importación

The screenshot shows a software interface for managing imports. It is divided into several sections:

- Seleccionar Transporte Terrestre:** A search box for 'Nombre' and a table with columns 'N°', 'Nombre', and 'Direccion'. The first row is selected, showing '1', 'COTRIBER LTDA', and 'Av Panamericana'.
- Seleccionar Agencia Aduanera:** A search box for 'Agencia' and a table with columns 'N°', 'Nombre', and 'Direccion'. The first row is selected, showing '1', 'Agencia Oriental', and 'Av Panamericana'.
- Seleccionar Proveedor:** A search box for 'Nombre' and a table with columns 'N°', 'Nombre', and 'Direccion'. The first row is selected, showing '1', 'EDIFICA S.R.L.', and 'Av La Pa'.
- Datos:** A date selector showing 'martes , 22 de diciembre de 2015'.
- Datos del Transporte:** 'Nombre: COTRIBER LTDA', 'Dirección: Av Panamericana'.
- Datos Agencia:** 'Nombre: Agencia Oriental', 'Dirección: Av Panamericana'.
- Datos Proveedor:** 'Nombre: EDIFICA S.R.L.', 'Dirección: Av La Paz'.
- Barco:** Radio buttons for 'Si' (checked) and 'No', and a dropdown menu showing 'VESSEL SANTA BAF'.
- Planilla:** A dropdown menu showing '1511234TJA' and a 'Registrar' button.

Pantalla donde se relaciona los datos de proveedores, Agencia Aduanera, Empresa Transporte, Barco, Planilla a una importación y registrar en la base de datos.

Pantalla n°14: Gestionar Reporte

ReportL.rdl - Vista previa del informe

de 1 de 1

100%

Buscar | Siguiente

INFORME DE IMPORTACIONES

17/01/2016 0:00:00

NOMBRE	DIRECCION	TELEFONO	NIT	CORREO	EMPRESA	AGENCIA	TELEFONO	DIRECCIÓN	NIT
COTRIBER LTDA	Av Panamericana	6961999	192072025	cotriber@hotm ail.com	EDIFICA S.R.L	Agencia Oriental	2234786	Av Panamericana	132178023

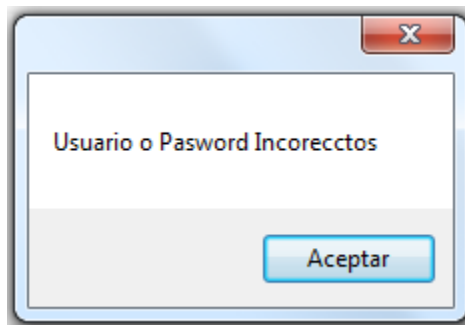
Página 1

Pantalla donde se muestra reporte de las importaciones de acuerdo a la fecha para realizar un seguimiento de la información.

Pantalla n°15: Pantalla de Errores en el Ingreso

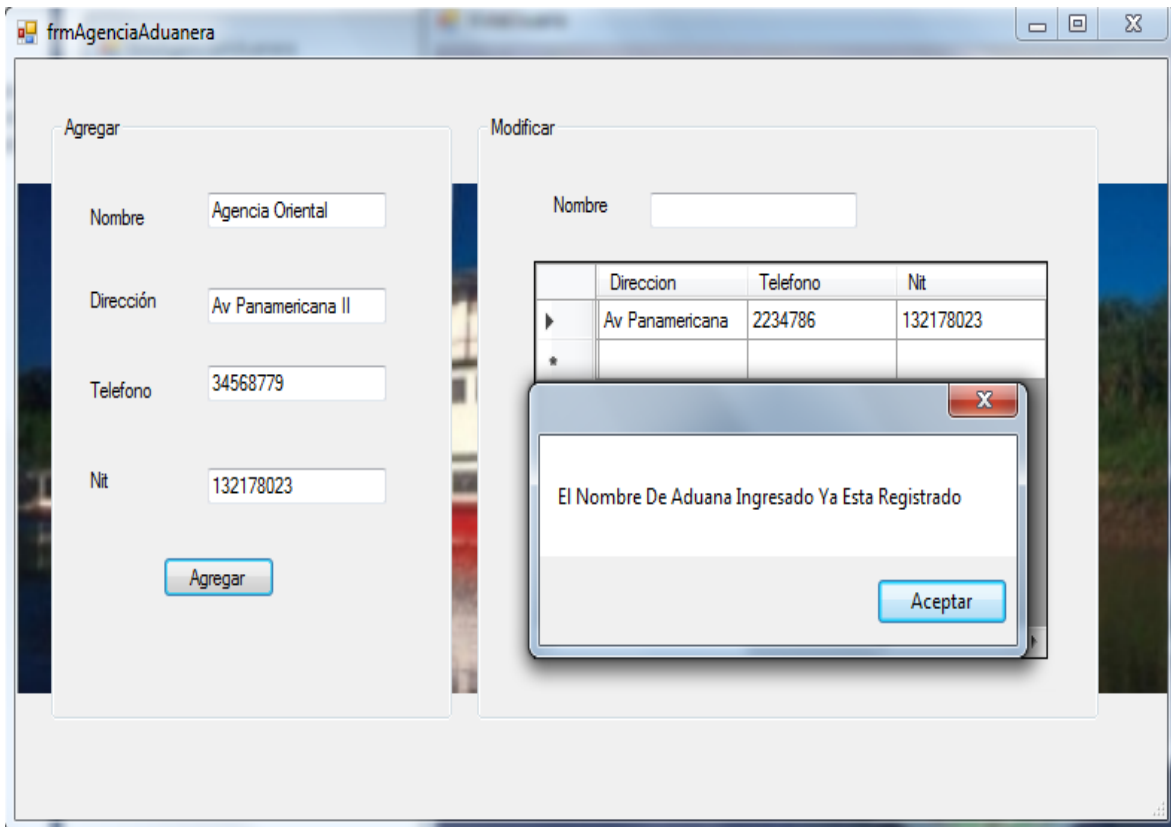


The screenshot shows a window titled "Form1" with a standard Windows-style title bar. Inside the window, there is a login form. On the left side of the form is an icon of a person in a suit holding a key. To the right of the icon, there are two input fields: "Usuario:" with the text "usuario" and "Contraseña:" with masked characters "*****". Below these fields is a blue button labeled "Ingresar". At the bottom right of the form area, there is a blue hyperlink that says "Registrar Nuevo Usuario".



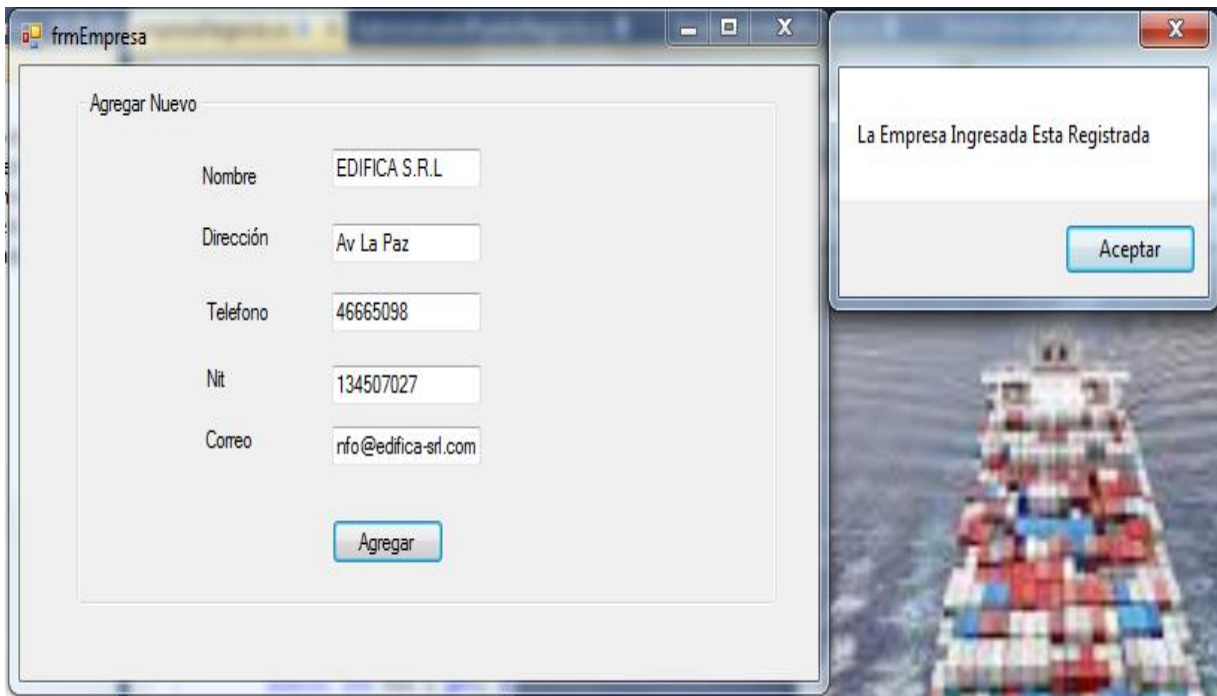
Pantalla donde se muestra un mensaje de error por un ingreso erróneo de los datos de usuario y contraseña.

Pantalla n°16: Pantalla de Errores en el Ingreso de Agencia Aduanera



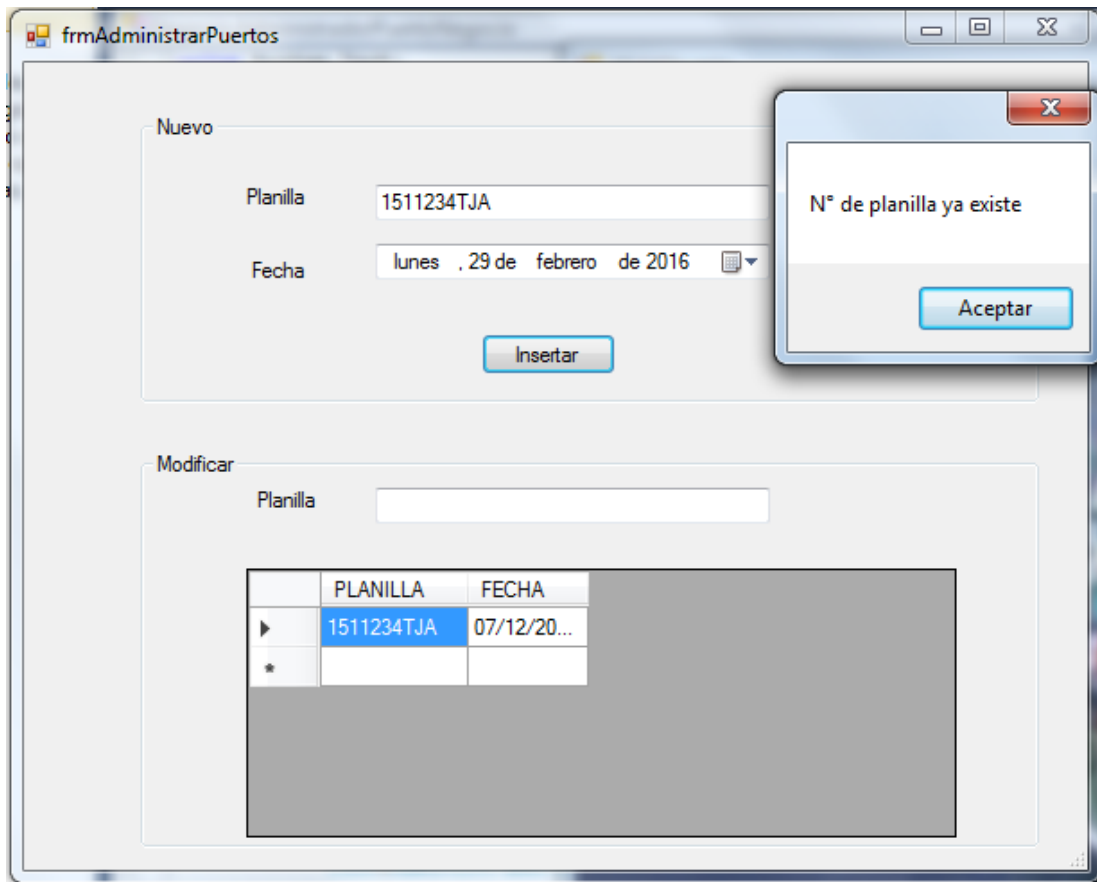
Pantalla donde se muestra un mensaje de error cuando se intenta ingresar el nombre de una agencia que ya está registrada.

Pantalla n°17: Pantalla de Errores en el Ingreso de Empresa



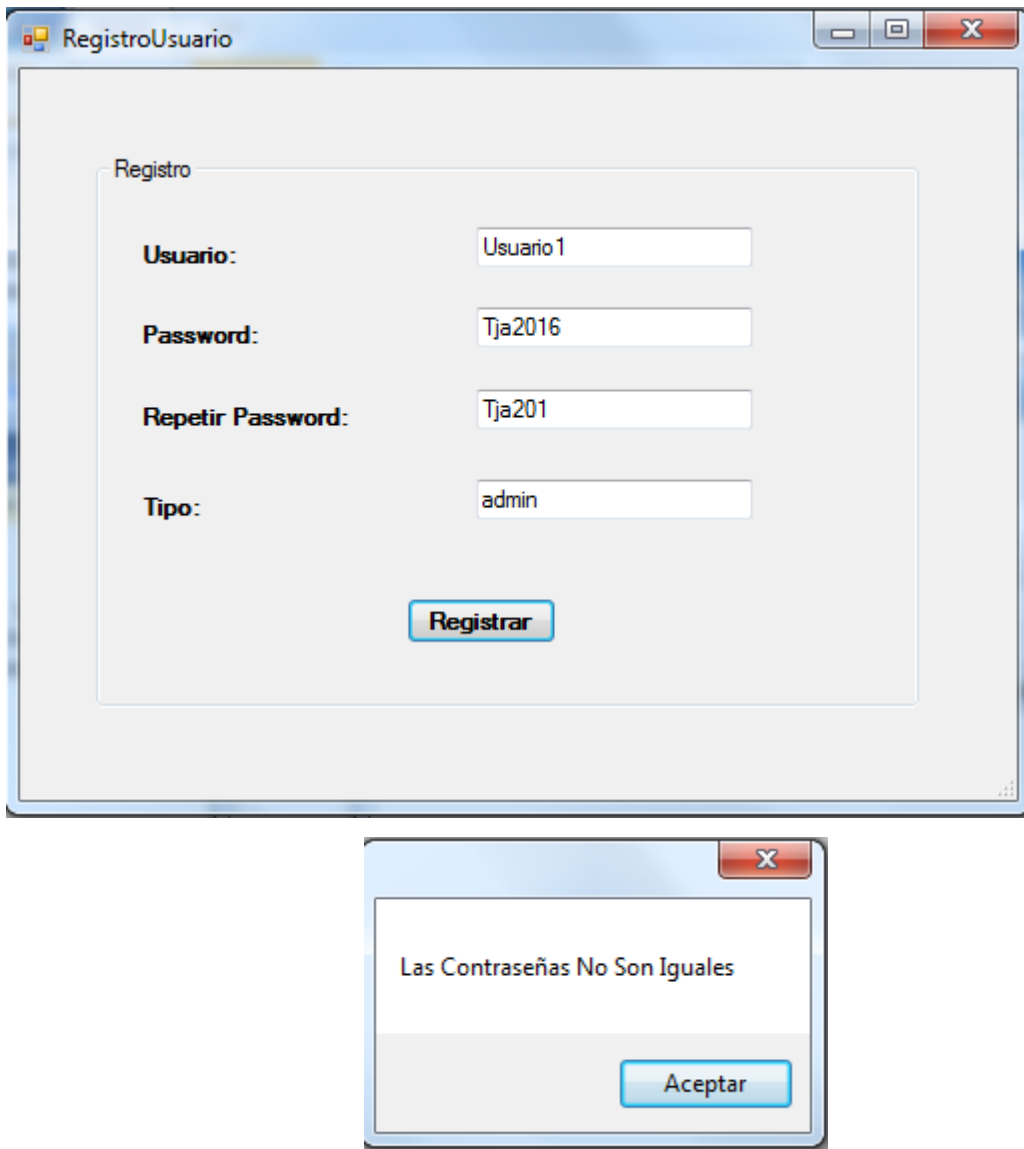
Pantalla donde se muestra un mensaje de error cuando se intenta ingresar el nombre de una empresa que ya está registrada.

Pantalla n°18: Pantalla de Errores en el Ingreso de Planilla



Pantalla donde se muestra un mensaje de error cuando se intenta ingresar el nombre de una planilla que ya está registrada.

Pantalla n°19: Pantalla de Errores en el Ingreso de Usuario



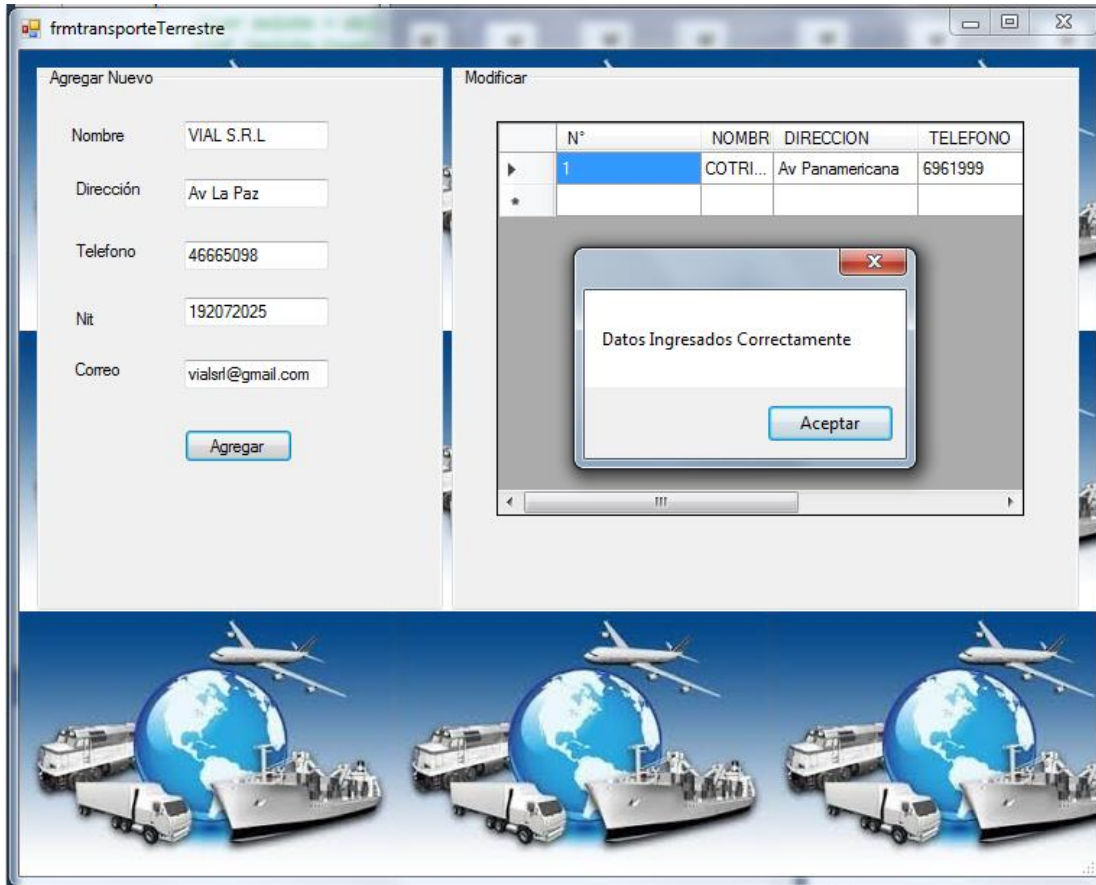
The image shows two overlapping windows from a software application. The top window, titled "RegistroUsuario", contains a registration form with the following fields and values:

Label	Value
Usuario:	Usuario 1
Password:	Tja2016
Repetir Password:	Tja201
Tipo:	admin

Below the form is a "Registrar" button. A smaller error dialog box is positioned in front of the bottom part of the registration window. It contains the text "Las Contraseñas No Son Iguales" and an "Aceptar" button.

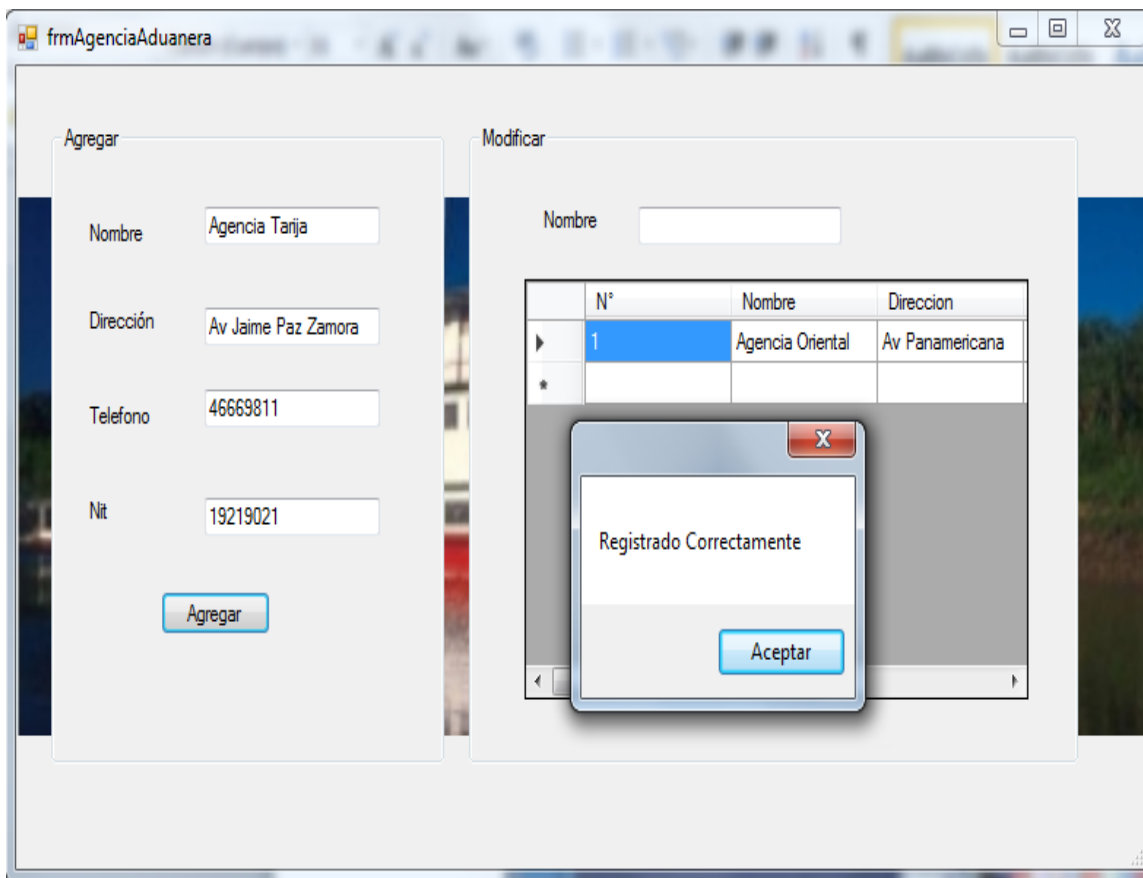
Pantalla donde se muestra un mensaje de error cuando se intenta registrar un usuario y las contraseñas no son iguales.

Pantalla n°20: Pantalla de mensaje en el ingreso empresa de transporte



Pantalla donde se muestra un mensaje correcto cuando se registra una empresa de transporte.

Pantalla n°21: Pantalla de mensaje en el ingreso agencia aduanera



Pantalla donde se muestra un mensaje correcto cuando se registra una agencia aduanera.

Pantalla n°22: Pantalla de mensaje, orden de compra

The screenshot shows a software window titled 'frmOrden'. It is divided into two main sections: 'Agregar' (Add) on the left and 'Modificar' (Modify) on the right. The 'Agregar' section contains several input fields: 'Fecha' (Date) set to 'martes , 23 de febrero de 2016', 'Monto' (Amount) set to '23000', 'N° Factura' (Invoice No.) set to '45678', 'Descripcion' (Description) set to 'Tuberia', and 'Proveedor' (Provider) set to 'TRIMAR SRL'. Below these fields is an 'Adicionar' (Add) button. The 'Modificar' section has an 'N° Factura' input field and a table with three columns: 'ID', 'FECHA', and 'MONTO'. The table contains one row with the values '2', '08/12/2015 12:0...', and '45000,00'. A modal dialog box is overlaid on the table, displaying the message 'Datos Ingresados Correctamente' (Data Entered Correctly) and an 'Aceptar' (Accept) button.

ID	FECHA	MONTO
2	08/12/2015 12:0...	45000,00

Pantalla donde se muestra un mensaje correcto cuando se registra una orden de compra.

4.6 Diseño Lógico del prototipo

4.6.1 Pantalla de Administrador de puertos

```
public class AdministrarPuertoDatos
{
    conexion con = new conexion();
    public DataTable existe(string texto)
    {
        string sql = "select * from ADMINISTRADOR_PUERTO where ID_PLANILLA
=" + texto + """;
        return con.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string text)
    {
        string sql = "select * from ADMINISTRADOR_PUERTO where ID_PLANILLA
like " + text + "%' ";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrar()
    {
        string sql = "select * from ADMINISTRADOR_PUERTO";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrarD()
    {
        string sql = "select * from ADMINISTRADOR_PUERTO where
ID_IMPORTACION is NULL";
        return con.solicitar(sql).Tables[0];
    }
    public void insertar(string planilla, DateTime fecha)
    {
        string sql = "insert into ADMINISTRADOR_PUERTO (ID_PLANILLA,FECHA)
values(' + planilla + ',' + fecha.ToString("yyyy-MM-dd") + ')"
```

```

        con.ejecutar(sql);
    }
    public void modificar(string planilla, DateTime fecha)
    {
        string sql = ("update ADMINISTRADOR_PUERTO set FECHA =" +
        fecha.ToString("yyyy-MM-dd") + " WHERE ID_PLANILLA=" + planilla + "");
        con.ejecutar(sql);
    }
}

```

4.6.1.1 Pantalla de Administrador de puertos

```

public class AdministradorPuertoNegocio
{
    AdministrarPuertoDatos obj = new AdministrarPuertoDatos();
    public string planilla { get; set; }
    public DateTime fecha { get; set; }
    public void agregar()
    {
        var existe = obj.existe(this.planilla).Rows;
        if (existe.Count > 0)
        {
            MessageBox.Show("N° de planilla ya existe");
        }
        else
        {
            obj.insertar(this.planilla, this.fecha);
            MessageBox.Show("Agregado Correctamente");
        }
    }
}
public List<AdministradorPuertoNegocio> listar()

```

```

    {
        List<AdministradorPuertoNegocio> lista = new
List<AdministradorPuertoNegocio>();
        foreach (DataRow f in obj.MOstrar().Rows)
        {
            AdministradorPuertoNegocio objt = new AdministradorPuertoNegocio();
            objt.planilla = f["ID_PLANILLA"].ToString();
            objt.fecha = Convert.ToDateTime(f["FECHA"]);
            lista.Add(objt);
        }
        return lista;
    }
    public List<AdministradorPuertoNegocio> listar(string texto)
    {
        List<AdministradorPuertoNegocio> lista = new
List<AdministradorPuertoNegocio>();
        foreach (DataRow f in obj.buscar(texto).Rows)
        {
            AdministradorPuertoNegocio objt = new AdministradorPuertoNegocio();
            objt.planilla = f["ID_PLANILLA"].ToString();
            objt.fecha = Convert.ToDateTime(f["FECHA"]);
            lista.Add(objt);
        }
        return lista;
    }
    public List<AdministradorPuertoNegocio> listarC()
    {
        List<AdministradorPuertoNegocio> lista = new
List<AdministradorPuertoNegocio>();
        foreach (DataRow f in obj.MOstrarD().Rows)
        {

```

```

        AdministradorPuertoNegocio objt = new AdministradorPuertoNegocio();
        objt.planilla = f["ID_PLANILLA"].ToString();
        objt.fecha = Convert.ToDateTime(f["FECHA"]);
        lista.Add(objt);
    }
    return lista;
}
public void modificar()
{
    obj.modificar(this.planilla, this.fecha);
    MessageBox.Show("Modificado Correctamente");
}
}

```

4.6.1.2 Pantalla de Administrador de puertos

```

public frmAdministrarPuertos()
{
    InitializeComponent();
    dataGridView1.Columns.Add("PLANILLA", "PLANILLA");
    dataGridView1.Columns.Add("FECHA", "FECHA");
    dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.ColumnHeader;
    actualizar();
}

AdministradorPuertoNegocio obj = new AdministradorPuertoNegocio();

private void actualizar()
{
    dataGridView1.Rows.Clear();
}

```

```

int i = 0;
foreach (AdministradorPuertoNegocio f in obj.listar())
{
    dataGridView1.Rows.Add(f.planilla,f.fecha);
    dataGridView1.Rows[i].Tag = f;
    i++;
}
}
private void actu( string texto)
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (AdministradorPuertoNegocio f in obj.listar(texto))
    {
        dataGridView1.Rows.Add(f.planilla, f.fecha);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
    actu(txtbuscar.Text);
}

private void button1_Click(object sender, EventArgs e)
{
    obj.planilla = txtplanilla.Text;
    obj.fecha = Convert.ToDateTime(dtfecha.Text);
    obj.agregar();
    actualizar();
}

```



```
}
```

4.6.2 Pantalla de Agencia Aduanera

```
public class AgenciaAduaneraDatos
{
    conexion con = new conexion();
    public void insertar(string nombre, string direccion, int telefono, int nit)
    {
        string sql = "insert into AGENCIA_ADUANERA
(NOMBRE,TELEFONO,DIRECCION,NIT)values('" + nombre + "', " + telefono + ", '" +
direccion + "', " + nit + ")";
        con.ejecutar(sql);
    }
    public DataTable existe(string nombre)
    {
        string sql = "select * from AGENCIA_ADUANERA where NOMBRE =" +
nombre + """;
        return con.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string nombre)
    {
        string sql = "select * from AGENCIA_ADUANERA where NOMBRE like '" +
nombre + "%' ";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrar()
    {
        string sql = "select * from AGENCIA_ADUANERA";
        return con.solicitar(sql).Tables[0];
    }
}
```

```

public void modificar(string nombre, string direccion, int telefono, int nit, int id)
{
    string sql = ("update AGENCIA_ADUANERA set NOMBRE=" + nombre + ",
TELEFONO=" + telefono + ", DIRECCION=" + direccion + ", NIT=" + nit + " where
ID_AGENCIA=" + id + "");
    con.ejecutar(sql);
}
}

```

4.6.2.1 Pantalla de Agencia Aduanera

```

public class AgenciaAduaneraNegocio
{
    AgenciaAduaneraDatos objdatos = new AgenciaAduaneraDatos();
    public int id { get; set; }
    public string nombre { get; set; }
    public string direccion { get; set; }
    public int telefono { get; set; }
    public int nit { get; set; }
    public void insertar()
    {
        var existe = objdatos.existe(this.nombre).Rows;
        if (existe.Count > 0)
        {
            MessageBox.Show("El Nombre De Aduana Ingresado Ya Esta Registrado");
        }
        else
        {
            objdatos.insertar(this.nombre, this.direccion, this.telefono, this.nit);
            MessageBox.Show("Registrado Correctamente");
        }
    }
}

```

```

}
public void modificar()
{
    var existe = objdatos.existe(this.nombre).Rows;
    if (existe.Count > 0)
    {
        MessageBox.Show("El Nombre De Aduana Ingresado Ya Esta Registrado");
    }
    else
    {
        objdatos.modificar(this.nombre, this.direccion, this.telefono, this.nit, this.id);
        MessageBox.Show("Modificado Correctamente");
    }
}
public List<AgenciaAduaneraNegocio> listar()
{
    List<AgenciaAduaneraNegocio> lista = new List<AgenciaAduaneraNegocio>();
    foreach (DataRow f in objdatos.MOstrar().Rows)
    {
        AgenciaAduaneraNegocio objt = new AgenciaAduaneraNegocio();
        objt.id = Convert.ToInt32(f["ID_AGENCIA"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
        objt.telefono = Convert.ToInt32(f["TELEFONO"]);
        objt.nit = Convert.ToInt32(f["NIT"]);
        lista.Add(objt);
    }
    return lista;
}
public List<AgenciaAduaneraNegocio> listar(string texto)
{

```

```

List<AgenciaAduaneraNegocio> lista = new List<AgenciaAduaneraNegocio>();
foreach (DataRow f in objdatos.buscar(texto).Rows)
{
    AgenciaAduaneraNegocio objt = new AgenciaAduaneraNegocio();
    objt.id = Convert.ToInt32(f["ID_AGENCIA"]);
    objt.nombre = f["NOMBRE"].ToString();
    objt.direccion = f["DIRECCION"].ToString();
    objt.telefono = Convert.ToInt32(f["TELEFONO"]);
    objt.nit = Convert.ToInt32(f["NIT"]);
    lista.Add(objt);
}
return lista;
}
}

```

4.6.2.2 Pantalla de Agencia Aduanera

```

public partial class frmAgenciaAduanera : Form
{
    AgenciaAduaneraNegocio objA = new AgenciaAduaneraNegocio();
    public frmAgenciaAduanera()
    {
        InitializeComponent();
        dataGridView1.Columns.Add("N°", "N°");
        dataGridView1.Columns.Add("Nombre", "Nombre");
        dataGridView1.Columns.Add("Direccion", "Direccion");
        dataGridView1.Columns.Add("Telefono", "Telefono");
        dataGridView1.Columns.Add("Nit", "Nit");
        actualizar();
    }
}

```

```
private void button1_Click(object sender, EventArgs e)
```

```
{  
    objA.nombre = txtnombre.Text;  
    objA.direccion = txtdireccion.Text;  
    objA.telefono = Convert.ToInt32(txttelefono.Text);  
    objA.nit = Convert.ToInt32(txtnit.Text);  
    objA.insertar();  
    actualizar();  
}
```

```
private void actualizar()
```

```
{  
    dataGridView1.Rows.Clear();  
    int i = 0;  
    foreach (AgenciaAduaneraNegocio f in objA.listar())  
    {  
        dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.nit);  
        dataGridView1.Rows[i].Tag = f;  
        i++;  
    }  
}
```

```
private void actu(string texto)
```

```
{  
    dataGridView1.Rows.Clear();  
    int i = 0;  
    foreach (AgenciaAduaneraNegocio f in objA.listar(texto))  
    {  
        dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.nit);  
        dataGridView1.Rows[i].Tag = f;  
        i++;  
    }  
}
```

```

    }
}

private void txtbuscar_KeyUp(object sender, KeyEventArgs e)
{
    actu(txtbuscar.Text);
}
}

```

4.6.3 Pantalla de Cuenta

```

public class CuentaDatos
{
    conexion conex = new conexion();
    public DataTable existe(int cuenta)
    {
        string sql = "select * from CUENTA where CUENTA =" + cuenta + """;
        return conex.solicitar(sql).Tables[0];
    }
    public void agregar(int cuenta,string beneficiario, string intermediario, int codigo,
string direccion, int telefono)
    {
        string sql = "insert into CUENTA values(" + cuenta + ", " + beneficiario + ", " +
intermediario + ", " + codigo + ", " + direccion + ","+telefono+"");
        conex.ejecutar(sql);
    }
    public DataTable MOstrar()
    {
        string sql = "select * from CUENTA";
        return conex.solicitar(sql).Tables[0];
    }
}

```

```

public DataTable buscar(string text)
{
    string sql = "select * from CUENTA where NOMBRE like '" + text + "%' ";
    return conex.solicitar(sql).Tables[0];
}
}

```

4.6.3.1 Pantalla de Cuenta

```

public class CuentaNegocio
{
    public int id { get; set; }
    public int cuenta { get; set; }
    public string beneficiario { get; set; }
    public string intermediario { get; set; }
    public int codigo { get; set; }
    public string direccion { get; set; }
    public int telefono { get; set; }

    CuentaDatos obj = new CuentaDatos();
    public void agregar()
    {
        obj.agregar(this.cuenta, this.beneficiario, this.intermediario, this.codigo,
this.direccion,this.telefono);
        MessageBox.Show("Datos Ingresados Correctamente");
    }
}

```

4.6.3.2 Pantalla de Cuenta

```

public partial class frmCuenta : Form
{
    public frmCuenta()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        CuentaNegocio onb = new CuentaNegocio();
        onb.cuenta = int.Parse(txtcuenta.Text);
        onb.beneficiario = txtbeneficiario.Text;
        onb.intermediario = txtintermediario.Text;
        onb.codigo = int.Parse(txtcodigo.Text);
        onb.direccion = txtdireccion.Text;
        onb.telefono = int.Parse(txttelefono.Text);
        onb.agregar();
    }
}

```

4.6.4 Pantalla de Empresa

```

public class empresaDatos
{
    conexion conex = new conexion();
    public DataTable existe(string nombre)
    {
        string sql = "select * from EMPRESA where NOMBRE=" + nombre + """;
        return conex.solicitar(sql).Tables[0];
    }
    public void agregar(string nombre, string direccion, int telefono, int nit, string correos)
    {

```



```

        string sql = "insert into EMPRESA values('" + nombre + "', '" + direccion + "', '" +
telefono + "', '" + nit + "', '" + correos + "')";
        conex.ejecutar(sql);
    }
    public DataTable MOstrar()
    {
        string sql = "select * from EMPRESA";
        return conex.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string text)
    {
        string sql = "select * from EMPRESA where NOMBRE like '" + text + "%' ";
        return conex.solicitar(sql).Tables[0];
    }
}

```

4.6.4.1 Pantalla de Empresa

```

public class empresaNegocio
{
    public int id { get; set; }
    public string nombre { get; set; }
    public string direccion { get; set; }
    public int telefono { get; set; }
    public int nit { get; set; }
    public string correo { get; set; }

    empresaDatos datos = new empresaDatos();
    public void agregar()
    {
        var exist = datos.existe(this.nombre).Rows;
        if (exist.Count > 0)

```

```

{
    MessageBox.Show("La Empresa Ingresada Esta Registrada");
}
else
{
    datos.agregar(this.nombre, this.direccion, this.telefono, this.nit, this.correo);
    MessageBox.Show("Empresa Agregada Correctamente");
}
}
public List<empresaNegocio> listar()
{
    List<empresaNegocio> lista = new List<empresaNegocio>();
    foreach (DataRow f in datos.MOstrar().Rows)
    {
        empresaNegocio objt = new empresaNegocio();
        objt.id = Convert.ToInt32(f["ID_EMPRESA"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
        objt.telefono = Convert.ToInt32(f["TELEFONO"]);
        objt.nit = Convert.ToInt32(f["NIT"]);
        objt.correo = f["CORREO"].ToString();
        lista.Add(objt);
    }
    return lista;
}
public List<empresaNegocio> listar(string texto)
{
    List<empresaNegocio> lista = new List<empresaNegocio>();
    foreach (DataRow f in datos.buscar(texto).Rows)
    {
        empresaNegocio objt = new empresaNegocio();

```

```

    objt.id = Convert.ToInt32(f["ID_EMPRESA"]);
    objt.nombre = f["NOMBRE"].ToString();
    objt.direccion = f["DIRECCION"].ToString();
    objt.telefono = Convert.ToInt32(f["TELEFONO"]);
    objt.nit = Convert.ToInt32(f["NIT"]);
    objt.correo = f["CORREO"].ToString();
    lista.Add(objt);
}
return lista;
}
}

```

4.6.4.2 Pantalla de Empresa

```

public partial class frmEmpresa : Form
{
    public frmEmpresa()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {

        empresaNegocio onb = new empresaNegocio();
        onb.nombre = txtnombre.Text;
        onb.direccion = txtdireccion.Text;
        onb.telefono = int.Parse(txttelefono.Text);
        onb.nit = int.Parse(txtnit.Text);
        onb.correo = txtcorreo.Text;
        onb.agregar();
    }
}

```

```
}
```

4.6.5 Pantalla de Importaciones

```
public class importacionDatos
{
    conexion conex = new conexion();
    public void agregar(DateTime fecha, int idUsuario, int idEmpresa, int idTerrestre,
string AdminPlanilla, int AgenciaAdu)
    {
        int IDImportacion = 0;
        string sqlImpotacion = "insert into IMPORTACION values('" +
fecha.ToString("yyyy-MM-dd") + "', '" + idUsuario + "', '" + idEmpresa + "')";
        conex.ejecutar(sqlImpotacion);
        string ultiImportacion = "select MAX(ID_IMPORTACION) as ULTIMO from
IMPORTACION";
        var ulti = conex.solicitar(ultiImportacion).Tables[0];
        foreach (DataRow item in ulti.Rows)
        {
            IDImportacion = Convert.ToInt32(item["ULTIMO"]);
            break;
        }
        string sqlImportacionTerrestre = "insert into
IMPORTACION_TRANSPORTE_TERRESTRE values('" + idTerrestre + "', '" +
IDImportacion + "', '" + fecha.ToString("yyyy-MM-dd") + "')";
        conex.ejecutar(sqlImportacionTerrestre);
        string sqlAdminPuerto = "update ADMINISTRADOR_PUERTO set
ID_IMPORTACION = '" + IDImportacion + "' where ID_PLANILLA = '" + AdminPlanilla
+ "'";
        conex.ejecutar(sqlAdminPuerto);
    }
}
```

```

        string sqlAgencia = "insert into IMPOR_ADUANA values('" + AgenciaAdu + "', '"
+ IDImportacion + "', '" + fecha.ToString("yyyy-MM-dd") + "')";
        conex.ejecutar(sqlAgencia);
    }

    public void agregar(DateTime fecha, int idUsuario, int idEmpresa, int idTerrestre,
string AdminPlanilla, int AgenciaAdu, string Barco)
    {
        int IDImportacion = 0;
        string sqlImpotacion = "insert into IMPORTACION values('" +
fecha.ToString("yyyy-MM-dd") + "', '" + idUsuario + "', '" + idEmpresa + "')";
        conex.ejecutar(sqlImpotacion);
        string ultiImportacion = "select MAX(ID_IMPORTACION) as ULTIMO from
IMPORTACION";
        var ulti = conex.solicitar(ultiImportacion).Tables[0];
        foreach (DataRow item in ulti.Rows)
        {
            IDImportacion = Convert.ToInt32(item["ULTIMO"]);
            break;
        }
        string sqlImportacionTerrestre = "insert into
IMPORTACION_TRANSPORTE_TERRESTRE values('" + idTerrestre + "', '" +
IDImportacion + "', '" + fecha.ToString("yyyy-MM-dd") + "')";
        conex.ejecutar(sqlImportacionTerrestre);
        string sqlAdminPuerto = "update ADMINISTRADOR_PUERTO set
ID_IMPORTACION = '" + IDImportacion + "' where ID_PLANILLA = '" + AdminPlanilla
+ "'";
        conex.ejecutar(sqlAdminPuerto);
        string sqlAgencia = "insert into IMPOR_ADUANA values('" + AgenciaAdu + "', '"
+ IDImportacion + "', '" + fecha.ToString("yyyy-MM-dd") + "')";
        conex.ejecutar(sqlAgencia);
    }

```

```

        string sqlBarco = "update TRANSPORTE_MARITIMO set ID_IMPORTACION =
''' + IDImportacion + ''' where NOMBREBARCO ='" + Barco + "'";
        conex.ejecutar(sqlBarco);
    }
    public DataTable MOstrar()
    {
        string sql = "";
        return conex.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string text)
    {
        string sql = "";
        return conex.solicitar(sql).Tables[0];
    }
}

```

4.6.5.1 Pantalla de Importaciones

```

public class importacionNegocio
{
    importacionDatos objImport = new importacionDatos();
    public int id { get; set; }
    public DateTime fecha { get; set; }
    public int idUsuario { get; set; }
    public int idEmpresa { get; set; }
    public int idTerrestre { get; set; }
    public string AdminPlanila { get; set; }
    public int AgenciaAdu { get; set; }
    public string barco { get; set; }
    public void agregar()
    {

```

```

        objImport.agregar(this.fecha, this.idUsuario, this.idEmpresa, this.idTerrestre,
this.AdminPlanila, this.AgenciaAdu);
        MessageBox.Show("Registrado");
    }
    public void agregarB()
    {
        objImport.agregar(this.fecha, this.idUsuario, this.idEmpresa, this.idTerrestre,
this.AdminPlanila, this.AgenciaAdu, this.barco);
        MessageBox.Show("Registrado");
    }
}

```

4.6.5.2 Pantalla de Importaciones

```

public partial class frmImportacion : Form
{
    MaritimoNegocios objmatitimo = new MaritimoNegocios();
    TerresteNegocio objTerrestre = new TerresteNegocio();
    AgenciaAduaneraNegocio objAgencia = new AgenciaAduaneraNegocio();
    empresaNegocio objEmpresa = new empresaNegocio();
    AdministradorPuertoNegocio objAdministradot = new
AdministradorPuertoNegocio();
    public frmImportacion()
    {

        InitializeComponent();
        dataGridView1.Columns.Add("N°", "N°");
        dataGridView1.Columns.Add("Nombre", "Nombre");
        dataGridView1.Columns.Add("Direccion", "Direccion");
        dataGridView2.Columns.Add("N°", "N°");
        dataGridView2.Columns.Add("Nombre", "Nombre");
    }
}

```

```

dataGridView2.Columns.Add("Direccion", "Direccion");
dataGridView3.Columns.Add("N°", "N°");
dataGridView3.Columns.Add("Nombre", "Nombre");
dataGridView3.Columns.Add("Direccion", "Direccion");
cargarDataTerrestre();
cargarDataAgencia();
cargarDataEmpresa();
cargarcombo();
cargarcombo2();
comboBarco.Visible = false;
lbIDUSUARIO.Visible = false;
lbIdAgencia.Visible = false;
lbIdEmpresa.Visible = false;
lbIdTerrestre.Visible = false;
}
#region cargarDatos
private void cargarDataTerrestre()
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (TerresteNegocio f in objTerrestre.listar())
    {
        dataGridView1.Rows.Add(f.id, f.nombre, f.direccion);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
private void BuscarTerrestre(string texto)
{
    dataGridView1.Rows.Clear();
    int i = 0;

```



```

foreach (TerresteNegocio f in objTerrestre.listarNombres(texto))
{
    dataGridView1.Rows.Add(f.id, f.nombre, f.direccion);
    dataGridView1.Rows[i].Tag = f;
    i++;
}
}
private void cargarDataAgencia()
{
    dataGridView2.Rows.Clear();
    int i = 0;
    foreach (AgenciaAduaneraNegocio f in objAgencia.listar())
    {
        dataGridView2.Rows.Add(f.id, f.nombre, f.direccion);
        dataGridView2.Rows[i].Tag = f;
        i++;
    }
}
private void BuscarAgencia(string texto)
{
    dataGridView2.Rows.Clear();
    int i = 0;
    foreach (AgenciaAduaneraNegocio f in objAgencia.listar(texto))
    {
        dataGridView2.Rows.Add(f.id, f.nombre, f.direccion);
        dataGridView2.Rows[i].Tag = f;
        i++;
    }
}
private void cargarDataEmpresa()
{

```

```

dataGridView3.Rows.Clear();
int i = 0;
foreach (empresaNegocio f in objEmpresa.listar())
{
    dataGridView3.Rows.Add(f.id, f.nombre, f.direccion);
    dataGridView3.Rows[i].Tag = f;
    i++;
}
}
private void BuscarEmpresa(string texto)
{
    dataGridView3.Rows.Clear();
    int i = 0;
    foreach (empresaNegocio f in objEmpresa.listar(texto))
    {
        dataGridView3.Rows.Add(f.id, f.nombre, f.direccion);
        dataGridView3.Rows[i].Tag = f;
        i++;
    }
}
void cargarcombo()
{
    comboBarco.Items.Clear();
    foreach (var item in objmatitimo.listar())
    {

        comboBarco.Items.Add(item.barco);
    }
    comboBarco.DisplayMember = "NOMBREBARCO";
}
void cargarcombo2()

```

```

{
    comboPlanilla.Items.Clear();
    foreach (var item in objAdministradot.listarC())
    {
        comboPlanilla.Items.Add(item.planilla);
    }
    comboPlanilla.DisplayMember = "ID_PLANILLA";
}
#endregion

private void txtterrestre_KeyUp(object sender, KeyEventArgs e)
{
    BuscarTerrestre(txtterrestre.Text);
}

private void txtagenciaAdu_KeyUp(object sender, KeyEventArgs e)
{
    BuscarAgencia(txtagenciaAdu.Text);
}

private void txtEmpresa_KeyUp(object sender, KeyEventArgs e)
{
    BuscarEmpresa(txtEmpresa.Text);
}

private void button1_Click(object sender, EventArgs e)
{
    importacionNegocio objIm = new importacionNegocio();
    if (CheckSi.Checked == true)
    {
        objIm.fecha = Convert.ToDateTime(Fecharegistro.Text);
        objIm.idTerrestre = Convert.ToInt32(lbidTerrestre.Text);
    }
}

```

```

objIm.idEmpresa = Convert.ToInt32(lbIdEmpresa.Text);
objIm.idUsuario = Convert.ToInt32(lbIDUSUSARIO.Text);
objIm.AdminPlanila = comboPlanilla.Text;
objIm.AgenciaAdu = Convert.ToInt32(lbidAgencia.Text);
objIm.barco = comboBarco.Text;
objIm.agregarB();
cargarcombo();
cargarcombo2();
this.Close();
frmPago ventana = new frmPago();
ventana.ShowDialog();

}
else
{
objIm.fecha = Convert.ToDateTime(Fecharegistro.Text);
objIm.idTerrestre = Convert.ToInt32(lbidTerrestre.Text);
objIm.idEmpresa = Convert.ToInt32(lbIdEmpresa.Text);
objIm.idUsuario = Convert.ToInt32(lbIDUSUSARIO.Text);
objIm.AdminPlanila = comboPlanilla.Text;
objIm.AgenciaAdu = Convert.ToInt32(lbidAgencia.Text);
objIm.agregar();
cargarcombo();
cargarcombo2();
this.Close();
frmPago ventana = new frmPago();
ventana.ShowDialog();

}
}

```

```
private void CheckSi_CheckedChanged_1(object sender, EventArgs e)
```

```

{

    if (CheckSi.Checked == true)
    {
        CheckNo.Checked = false;
        comboBarco.Visible = true;
    }
}

```

```

private void CheckNo_CheckedChanged_1(object sender, EventArgs e)
{
    if (CheckNo.Checked == true)
    {
        CheckSi.Checked = false;
        comboBarco.Visible = false;
    }
}

```

```

private void dataGridView1_CellClick_1(object sender, DataGridViewCellEventArgs
e)
{
    objTerrestre =
(TerresteNegocio)dataGridView1.Rows[dataGridView1.CurrentRow.Index].Tag;
    lblidTerrestre.Text = objTerrestre.id.ToString();
    lblnombreTerrestre.Text = objTerrestre.nombre.ToString();
    lbldireccionTerrestre.Text = objTerrestre.direccion.ToString();
}

```

```

private void dataGridView2_CellClick_1(object sender, DataGridViewCellEventArgs
e)
{

```

```

        objAgencia =
(AgenciaAduaneraNegocio)dataGridView2.Rows[dataGridView2.CurrentRow.Index].Tag;
        lbIdAgencia.Text = objAgencia.id.ToString();
        lbNombreAgencia.Text = objAgencia.nombre.ToString();
        lbDireccionAgencia.Text = objAgencia.direccion.ToString();
    }

private void dataGridView3_CellClick_1(object sender, DataGridViewCellEventArgs
e)
{
    objEmpresa =
(empresaNegocio)dataGridView3.Rows[dataGridView3.CurrentRow.Index].Tag;
    lbIdEmpresa.Text = objEmpresa.id.ToString();
    lbNombreEmpresa.Text = objEmpresa.nombre.ToString();
    lbDireccionEmpresa.Text = objEmpresa.direccion.ToString();
}
}

```

4.6.6 Pantalla de Transporte Marítimo

```

public class MaritimoDatos
{
    conexion con = new conexion();
    public DataTable existe(string texto)
    {
        string sql = "select * from TRANSPORTE_MARITIMO where
NOMBREBARCO=" + texto + """;
        return con.solicitar(sql).Tables[0];
    }
    public void agregar(string texto)
    {

```

```

        string sql = "insert into TRANSPORTE_MARITIMO (NOMBREBARCO)
values('" + texto + "')";
        con.ejecutar(sql);
    }
    public void modificar(string barco, int id)
    {
        string sql = "update TRANSPORTE_MARITIMO set NOMBREBARCO =" +
barco + " where ID_MARITIMO =" + id + """;
        con.ejecutar(sql);
    }
    public DataTable buscar(string text)
    {
        string sql = "select * from TRANSPORTE_MARITIMO where NOMBREBARCO
like '" + text + "%' ";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrar()
    {
        string sql = "select * from TRANSPORTE_MARITIMO";
        return con.solicitar(sql).Tables[0];
    }
}

```

4.6.6.1 Pantalla de Transporte Marítimo

```

public class MaritimoNegocios
{
    public int id { get; set; }
    public string barco { get; set; }
    MaritimoDatos obj = new MaritimoDatos();
    public void insertar()
    {

```

```

var existe = obj.existe(this.barco).Rows;
if (existe.Count > 0)
{
    MessageBox.Show("El Nombre De Barco Ingresado Ya Esta Registrado");
}
else
{
    obj.agregar(this.barco);
    MessageBox.Show("Ingresado Correctamente");
}
}
public void modificar()
{
    var existe = obj.existe(this.barco).Rows;
    if (existe.Count > 0)
    {
        MessageBox.Show("El Nombre De Barco Ingresado Ya Esta Registrado");
    }
    else
    {
        obj.modificar(this.barco, this.id);
        MessageBox.Show("Modificado Correctamente Correctamente");
    }
}
public List<MaritimoNegocios> listar()
{
    List<MaritimoNegocios> lista = new List<MaritimoNegocios>();
    foreach (DataRow f in obj.MOstrar().Rows)
    {
        MaritimoNegocios objt = new MaritimoNegocios();
        objt.id = Convert.ToInt32(f["ID_MARITIMO"]);
    }
}

```



```

        objt.barco = Convert.ToString(f["NOMBREBARCO"]);
        lista.Add(objt);
    }
    return lista;
}
public List<MaritimoNegocios> listar(string texto)
{
    List<MaritimoNegocios> lista = new List<MaritimoNegocios>();
    foreach (DataRow f in obj.buscar(texto).Rows)
    {
        MaritimoNegocios objt = new MaritimoNegocios();
        objt.id = Convert.ToInt32(f["ID_MARITIMO"]);
        objt.barco = Convert.ToString(f["NOMBREBARCO"]);
        lista.Add(objt);
    }
    return lista;
}
}

```

4.6.6.2 Pantalla de Transporte Marítimo

```

public partial class frmTransporteMaritimo : Form
{
    MaritimoNegocios obj = new MaritimoNegocios();
    public frmTransporteMaritimo()
    {
        InitializeComponent();
        dataGridView1.Columns.Add("ID", "ID");
        dataGridView1.Columns.Add("Nombre Barco", "Nombre Barco");
        actualizar();
    }
    private void actualizar()

```

```

{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (MaritimoNegocios f in obj.listar())
    {
        dataGridView1.Rows.Add(f.id, f.barco);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
private void actu(string texto)
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (MaritimoNegocios f in obj.listar(texto))
    {
        dataGridView1.Rows.Add(f.id, f.barco);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
private void txtbuscar_KeyUp(object sender, KeyEventArgs e)
{
    actu(txtbuscar.Text);
}
private void button1_Click(object sender, EventArgs e)
{
    obj.barco = txtbarco.Text;
    obj.insertar();
    actualizar();
}

```

```
}
```

4.6.7 Pantalla de Orden de Compra

```
public class OrdenDatos
{
    conexion con = new conexion();
    public DataTable existe(string nombre)
    {
        string sql = "select * from ORDEN_COMPRA where NFACTURA = " + nombre +
        """";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string text)
    {
        string sql = "select * from ORDEN_COMPRA where NFACTURA like " + text +
        "%' ";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrar()
    {
        string sql = "select * from ORDEN_COMPRA";
        return con.solicitar(sql).Tables[0];
    }
    public void insertar(DateTime fecha, decimal monto, string nfactura, string idc, string
detalle)
    {
        int idprov = 0;
        string idcuenta = "select ID_PROVEEDOR as ULTIMO from PROVEEDOR
where NOMBRE=" + idc + """";
        var ulti = con.solicitar(idcuenta).Tables[0];
        foreach (DataRow item in ulti.Rows)
```

```

    {
        idprov = Convert.ToInt32(item["ULTIMO"]);
        break;
    }

    string sql = "insert into ORDEN_COMPRA
(FECHA,MONTO,NFACTURA,ID_PROVEEDOR,DESCRIPCION) values('' +
fecha.ToString("yyyy-MM-dd") + "',' + monto + "',' + nfactura + "',' + idprov +
','+detalle+'");
    con.ejecutar(sql);
}

public void modificar(int id, DateTime fecha, decimal monto, string nfactura, string
idc, string detalle)
{
    string sql = ("update ORDEN_COMPRA set FECHA=" + fecha.ToString("yyyy-
MM-dd") + "','MONTO=" + monto + "',' NFACTURA=" + nfactura + "','
ID_PROVEEDOR=" + idc + "','+detalle+' WHERE ID_COMPRA = ' + id + '");
    con.ejecutar(sql);
}
}

```

4.6.7.1 Pantalla de Orden de Compra

```

public class OrdenNegocio
{
    public int id { get; set; }
    public DateTime fecha { get; set; }
    public decimal monto { get; set; }
    public string nfactura { get; set; }
    public string idc { get; set; }
    public string detalle { get; set; }
    OrdenDatos obj = new OrdenDatos();
    public void agregar()

```

```

{

obj.insertar(this.fecha, this.monto, this.nfactura, this.idc, this.detalle);
MessageBox.Show("Datos Ingresados Correctamente");

}

public void modificar()
{
obj.modificar(this.id, this.fecha, this.monto, this.nfactura, this.idc, this.detalle);
MessageBox.Show("Modificado Correctamente");
}

public List<OrdenNegocio> listarNombres(string texto)
{
List<OrdenNegocio> lista = new List<OrdenNegocio>();
foreach (DataRow f in obj.buscar(texto).Rows)
{
OrdenNegocio objt = new OrdenNegocio();
objt.id = Convert.ToInt32(f["ID_COMPRA"]);
objt.fecha = Convert.ToDateTime( f["FECHA"].ToString());
objt.monto = Convert.ToDecimal( f["MONTO"].ToString());
objt.nfactura = f["NFACTURA"].ToString();
objt.idc = f["ID_PROVEEDOR"].ToString();
objt.detalle = f["DESCRIPCION"].ToString();
lista.Add(objt);
}
return lista;
}

public List<OrdenNegocio> listar()
{
List<OrdenNegocio> lista = new List<OrdenNegocio>();
foreach (DataRow f in obj.MOstrar().Rows)

```

```

{
    OrdenNegocio objt = new OrdenNegocio();
    objt.id = Convert.ToInt32(f["ID_COMPRA"]);
    objt.fecha = Convert.ToDateTime(f["FECHA"].ToString());
    objt.monto = Convert.ToDecimal(f["MONTO"].ToString());
    objt.nfactura = f["NFACTURA"].ToString();
    objt.idc = f["ID_PROVEEDOR"].ToString();
    objt.detalle = f["DESCRIPCION"].ToString();
    lista.Add(objt);
}
return lista;
}
}

```

4.6.7.2 Pantalla de Orden de Compra

```
public partial class frmOrden : Form
```

```

{
    public frmOrden()
    {
        InitializeComponent();
        dataGridView1.Columns.Add("ID", "ID");
        dataGridView1.Columns.Add("FECHA", "FECHA");
        dataGridView1.Columns.Add("MONTO", "MONTO");
        dataGridView1.Columns.Add("NFACTURA", "NFACTURA");
        dataGridView1.Columns.Add("DESCRIPCION", "DESCRIPCION");
        dataGridView1.Columns.Add("ID_PROVEEDOR", "ID_PROVEEDOR");
        dataGridView1.Columns[1].Width = 100;
        actualizar();
        cargar();
    }
}

```

```

OrdenNegocio obj = new OrdenNegocio();
#region Mostrar En DataGrid
void cargar()
{
    SqlConnection con = new SqlConnection("Data Source=.;Initial
Catalog=ADUANA;Integrated Security=True");
    SqlCommand comando = new SqlCommand("select NOMBRE from
PROVEEDOR", con);
    con.Open();
    try
    {
        SqlDataReader leer = comando.ExecuteReader();
        while (leer.Read())
        {
            cproveedor.Items.Add(leer["NOMBRE"].ToString());
            cproveedor.DisplayMember = "NOMBRE";
        }
    }
    finally
    { }
}

void actualizar()
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (OrdenNegocio f in obj.listar())
    {
        dataGridView1.Rows.Add(f.id, f.fecha, f.monto, f.nfactura, f.idc, f.detalle);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}

```

```

    }
}
void datos(string texto)
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (OrdenNegocio f in obj.listarNombres(texto))
    {
        dataGridView1.Rows.Add(f.id, f.fecha, f.monto, f.nfactura, f.idc, f.detalle);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
    datos(txtfactura.Text);
}
#endregion

private void button1_Click(object sender, EventArgs e)
{
    obj.fecha = Convert.ToDateTime(dtfecha.Text);
    obj.monto = Convert.ToDecimal(txtmonto.Text);
    obj.nfactura = txtfactura.Text;
    obj.detalle = txtdescripcion.Text;
    obj.idc = cproveedor.Text;
    obj.agregar();
    actualizar();
}
}

```


4.6.8 Pantalla de Proveedor

```
public class Proveedor
{
    conexion con = new conexion();
    public DataTable existe(string nombre)
    {
        string sql = "select * from PROVEEDOR where NOMBRE = '" + nombre + "'";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable buscar(string text)
    {
        string sql = "select * from PROVEEDOR where NOMBRE like '" + text + "%' ";
        return con.solicitar(sql).Tables[0];
    }
    public DataTable MOstrar()
    {
        string sql = "select * from PROVEEDOR";
        return con.solicitar(sql).Tables[0];
    }
    public void insertar(string nombre, string direccion, int telefono, int idc, string pais)
    {
        int Cuenta1 = 0;
        string idcuenta = "select ID_CUENTA as ULTIMO from CUENTA where
CUENTA='"+idc+"'";
        var ulti = con.solicitar(idcuenta).Tables[0];
        foreach (DataRow item in ulti.Rows)
        {
            Cuenta1 = Convert.ToInt32(item["ULTIMO"]);
            break;
        }
    }
}
```

```

        string sql = "insert into PROVEEDOR
(NOMBRE,DIRECCION,PAIS,TELEFONO,ID_CUENTA) values('" + nombre + "','" +
direccion + "','" + pais + "','" + telefono + "','" + Cuenta1 + "')";
        con.ejecutar(sql);
    }
    public void modificar(int id, string nombre, string direccion, int telefono, int idc, string
pais)
    {
        string sql = ("update PROVEEDOR set NOMBRE='" + nombre +
'",DIRECCION='" + direccion + "', PAIS='" + pais + "', TELEFONO='" + telefono + "',
ID_CUENTA='" + idc + "' WHERE ID_PROVEEDOR =" + id + "'");
        con.ejecutar(sql);
    }
}

```

4.6.8.1 Pantalla de Proveedor

```

public class ProveedorNegocio
{
    public int id { get; set; }
    public string nombre { get; set; }
    public string direccion { get; set; }
    public int telefono { get; set; }
    public int idc { get; set; }
    public string pais { get; set; }
    Proveedor obj = new Proveedor();
    public void agregar()
    {

        obj.insertar(this.nombre, this.direccion, this.telefono, this.idc, this.pais);
        MessageBox.Show("Datos Ingresados Correctamente");
    }
}

```

```

}
public void modificar()
{
    obj.modificar(this.id, this.nombre, this.direccion, this.telefono, this.idc, this.pais);
    MessageBox.Show("Modificado Correctamente");
}
public List<ProveedorNegocio> listarNombres(string texto)
{
    List<ProveedorNegocio> lista = new List<ProveedorNegocio>();
    foreach (DataRow f in obj.buscar(texto).Rows)
    {
        ProveedorNegocio objt = new ProveedorNegocio();
        objt.id = Convert.ToInt32(f["ID_PROVEEDOR"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
        objt.pais = f["PAIS"].ToString();
        objt.telefono = Convert.ToInt32(f["TELEFONO"].ToString());
        objt.idc = Convert.ToInt32(f["ID_CUENTA"].ToString());
        lista.Add(objt);
    }
    return lista;
}
public List<ProveedorNegocio> listar()
{
    List<ProveedorNegocio> lista = new List<ProveedorNegocio>();
    foreach (DataRow f in obj.MOstrar().Rows)
    {
        ProveedorNegocio objt = new ProveedorNegocio();
        objt.id = Convert.ToInt32(f["ID_PROVEEDOR"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
    }
}

```

```

    objt.pais = f["PAIS"].ToString();
    objt.telefono = Convert.ToInt32(f["TELEFONO"].ToString());
    objt.idc = Convert.ToInt32(f["ID_CUENTA"].ToString());
    lista.Add(objt);
}
return lista;
}
}

```

4.6.8.2 Pantalla de Proveedor

```

public partial class frmProveedor : Form
{
    public frmProveedor()
    {
        InitializeComponent();
        dataGridView1.Columns.Add("Nº", "Nº");
        dataGridView1.Columns.Add("NOMBRE", "NOMBRE");
        dataGridView1.Columns.Add("DIRECCION", "DIRECCION");
        dataGridView1.Columns.Add("TELEFONO", "TELEFONO");
        dataGridView1.Columns.Add("NIT", "NIT");
        dataGridView1.Columns.Add("CORREO", "CORREO");
        dataGridView1.Columns[1].Width = 50;
        actualizar();
        cargar();
    }
    ProveedorNegocio obj = new ProveedorNegocio();
    #region Mostrar En DataGridView
    void cargar()
    {

```

```

SqlConnection con = new SqlConnection("Data Source=.;Initial
Catalog=ADUANA;Integrated Security=True");
SqlCommand comando = new SqlCommand("select cuenta from CUENTA", con);
con.Open();

try
{
    SqlDataReader leer = comando.ExecuteReader();
    while (leer.Read())
    {

        ccuenta.Items.Add(leer["CUENTA"].ToString());
        ccuenta.DisplayMember = "CUENTA";
    }
}
finally
{ }
}
void actualizar()
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (ProveedorNegocio f in obj.listar())
    {
        dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.idc, f.pais);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}
void datos(string texto)
{

```

```

dataGridView1.Rows.Clear();
int i = 0;
foreach (ProveedorNegocio f in obj.listarNombres(texto))
{
    dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.idc, f.pais);
    dataGridView1.Rows[i].Tag = f;
    i++;
}
}
private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
    datos(txtnombre.Text);
}
#endregion
private void frmtransporteTerrestre_Load_1(object sender, EventArgs e)
{
}

private void button1_Click(object sender, EventArgs e)
{
    obj.nombre = txtnombre.Text;
    obj.direccion = txtdireccion.Text;
    obj.telefono = Convert.ToInt32(txttelefono.Text);
    obj.idc = Convert.ToInt32(ccuenta.Text);
    obj.pais = txtpais.Text;
    obj.agregar();
    actualizar();
}
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs

```

e)

```

    {
        frmCuenta objp = new frmCuenta();
        objp.ShowDialog();
    }
}

```

4.6.9 Pantalla de Proveedor

```
public class TerrestreDatos
```

```

    {
        conexion con = new conexion();
        public DataTable existe(string nombre)
        {
            string sql = "select * from TRANSPORTE_TERRESTRE where NOMBRE = '" +
nombre + "'";
            return con.solicitar(sql).Tables[0];
        }
        public DataTable buscar(string text)
        {
            string sql = "select * from TRANSPORTE_TERRESTRE where NOMBRE like '" +
text + "%' ";
            return con.solicitar(sql).Tables[0];
        }
        public DataTable MOstrar()
        {
            string sql = "select * from TRANSPORTE_TERRESTRE";
            return con.solicitar(sql).Tables[0];
        }
        public void insertar(string nombre, string direccion, int telefono, int nit, string correo)
        {

```

```

        string sql = "insert into TRANSPORTE_TERRESTRE
(NOMBRE,DIRECCION,TELEFONO,NIT,CORREO) values('" + nombre + "','" +
direccion + "','" + telefono + "','" + nit + "','" + correo + "')";
        con.ejecutar(sql);
    }
    public void modificar(int id, string nombre, string direccion, int telefono, int nit, string
correo)
    {
        string sql = ("update TRANSPORTE_TERRESTRE set NOMBRE='" + nombre +
"',DIRECCION='" + direccion + "', TELEFONO='" + telefono + "', NIT='" + nit + "',
CORREO='" + correo + "' WHERE ID_TERRESTRE =" + id + "''");
        con.ejecutar(sql);
    }
}

```

4.6.9.1 Pantalla de Proveedor

```

public class TerresteNegocio
{
    public int id { get; set; }
    public string nombre { get; set; }
    public string direccion { get; set; }
    public int telefono { get; set; }
    public int nit { get; set; }
    public string correo { get; set; }
    TerrestreDatos obj = new TerrestreDatos();
    public void agregar()
    {

        obj.insertar(this.nombre, this.direccion, this.telefono, this.nit, this.correo);
        MessageBox.Show("Datos Ingresados Correctamente");
    }
}

```



```

}
public void modificar()
{
    obj.modificar(this.id, this.nombre, this.direccion, this.telefono, this.nit, this.correo);
    MessageBox.Show("Modificado Correctamente");
}
public List<TerresteNegocio> listarNombres(string texto)
{
    List<TerresteNegocio> lista = new List<TerresteNegocio>();
    foreach (DataRow f in obj.buscar(texto).Rows)
    {
        TerresteNegocio objt = new TerresteNegocio();
        objt.id = Convert.ToInt32(f["ID_TERRESTRE"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
        objt.telefono = Convert.ToInt32(f["TELEFONO"].ToString());
        objt.nit = Convert.ToInt32(f["NIT"].ToString());
        objt.correo = f["CORREO"].ToString();
        lista.Add(objt);
    }
    return lista;
}
public List<TerresteNegocio> listar()
{
    List<TerresteNegocio> lista = new List<TerresteNegocio>();
    foreach (DataRow f in obj.MOstrar().Rows)
    {
        TerresteNegocio objt = new TerresteNegocio();
        objt.id = Convert.ToInt32(f["ID_TERRESTRE"]);
        objt.nombre = f["NOMBRE"].ToString();
        objt.direccion = f["DIRECCION"].ToString();
    }
}

```

```

    objt.telefono = Convert.ToInt32(f["TELEFONO"].ToString());
    objt.nit = Convert.ToInt32(f["NIT"].ToString());
    objt.correo = f["CORREO"].ToString();
    lista.Add(objt);
}
return lista;
}
}

```

4.6.9.2 Pantalla de Proveedor

```
public partial class frmProveedor : Form
```

```

{
    public frmProveedor()
    {
        InitializeComponent();
        dataGridView1.Columns.Add("N°", "N°");
        dataGridView1.Columns.Add("NOMBRE", "NOMBRE");
        dataGridView1.Columns.Add("DIRECCION", "DIRECCION");
        dataGridView1.Columns.Add("TELEFONO", "TELEFONO");
        dataGridView1.Columns.Add("NIT", "NIT");
        dataGridView1.Columns.Add("CORREO", "CORREO");
        dataGridView1.Columns[1].Width = 50;
        actualizar();
        cargar();
    }
    ProveedorNegocio obj = new ProveedorNegocio();
    #region Mostrar En DataGrid
    void cargar()
    {
        SqlConnection con = new SqlConnection("Data Source=.;Initial
Catalog=ADUANA;Integrated Security=True");
        SqlCommand comando = new SqlCommand("select cuenta from CUENTA", con);

```

```

con.Open();

try
{
    SqlDataReader leer = comando.ExecuteReader();
    while (leer.Read())
    {
        ccuenta.Items.Add(leer["CUENTA"].ToString());
        ccuenta.DisplayMember = "CUENTA";
    }
}
finally
{ }

}

void actualizar()
{
    dataGridView1.Rows.Clear();
    int i = 0;
    foreach (ProveedorNegocio f in obj.listar())
    {
        dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.idc, f.pais);
        dataGridView1.Rows[i].Tag = f;
        i++;
    }
}

void datos(string texto)
{
    dataGridView1.Rows.Clear();
    int i = 0;

```

```

foreach (ProveedorNegocio f in obj.listarNombres(texto))
{
    dataGridView1.Rows.Add(f.id, f.nombre, f.direccion, f.telefono, f.idc, f.pais);
    dataGridView1.Rows[i].Tag = f;
    i++;
}
}
private void textBox1_KeyUp(object sender, KeyEventArgs e)
{
    datos(txtnombre.Text);
}
#endregion
private void button1_Click(object sender, EventArgs e)
{
    obj.nombre = txtnombre.Text;
    obj.direccion = txtdireccion.Text;
    obj.telefono = Convert.ToInt32(txttelefono.Text);
    obj.idc = Convert.ToInt32(ccuenta.Text);
    obj.pais = txtpais.Text;
    obj.agregar();
    actualizar();
}

private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e)
{
    frmCuenta objp = new frmCuenta();
    objp.ShowDialog();
}
}

```

4.6.10 Pantalla de Usuario

```

public class usuarioDatos
{
    conexion conex = new conexion();
    public DataTable autentificar(string usuario, string clave)
    {
        string sql = "select USUARIO.ID_USUARIO, USUARIO.USUARIO,
USUARIO.CONSTRACENA, TIPO_USUARIO.NOMBRE from USUARIO,
TIPO_USUARIO where USUARIO.ID_USUARIO = TIPO_USUARIO.ID_USUARIO
and USUARIO.USUARIO='" + usuario + "' and USUARIO.CONSTRACENA='" + clave +
'''';
        return conex.solicitar(sql).Tables[0];
    }
    public void agregarUsuario(string usuario, string clave, string tipo)
    {
        try
        {
            int ultimoID = 0;
            string AgregarUsuario = "insert into USUARIO values('" + usuario + "', '" + clave
+ "')";
            conex.ejecutar(AgregarUsuario);
            string ultimoRegistro = "select MAX(ID_USUARIO) as 'ID' from USUARIO";
            var s = conex.solicitar(ultimoRegistro).Tables[0];
            foreach (DataRow item in s.Rows)
            {
                ultimoID = Convert.ToInt32(item["ID"]);
                break;
            }
            string AgregarTipo = "insert into TIPO_USUARIO values('" + tipo + "', '" +
ultimoID + "')";
            conex.ejecutar(AgregarTipo);
            MessageBox.Show("Usuario Agregado");
        }
    }
}

```

```

    }
    catch
    {
        MessageBox.Show("Error Al Registrar");
    }
}
}
}

```

4.6.10.1 Pantalla de Usuario

```
public class UsuariosNegocio
```

```

{
    public int id { get; set; }
    public string nombre { get; set; }
    public string password { get; set; }
    public string Repassword { get; set; }
    public string tipo { get; set; }
    usuarioDatos datos = new usuarioDatos();
    public List<UsuariosNegocio> listaDeusuarios()
    {
        List<UsuariosNegocio> datosDeUsuario = new List<UsuariosNegocio>();
        UsuariosNegocio obj = new UsuariosNegocio();
        foreach (DataRow f in datos.autenticar(this.nombre, this.password).Rows)
        {
            obj.id = Convert.ToInt32(f["ID_USUARIO"]);
            obj.nombre = f["USUARIO"].ToString();
            obj.password = f["CONSTRACENA"].ToString();
            obj.tipo = f["NOMBRE"].ToString();
            datosDeUsuario.Add(obj);
        }
        return datosDeUsuario;
    }
}

```

```

public void agregarUsuario()
{
    try
    {
        if (this.password == this.Repassword)
        {
            datos.agregarUsuario(this.nombre, this.password, this.tipo);
        }
        else
        {
            MessageBox.Show("Las Contraseñas No Son Iguales");
        }
    }
    catch
    {
        MessageBox.Show("Error");
    }
}
}

```

4.6.10.2 Pantalla de Usuario

```

public partial class VistaUsuario : Form
{
    public VistaUsuario()
    {
        InitializeComponent();
        LbIdUsuario.Visible = false;
    }

    private void toolStripButton2_Click_1(object sender, EventArgs e)

```

```

{
    frmEmpresa obj = new frmEmpresa();
    obj.Show();
}

private void toolStripButton3_Click_1(object sender, EventArgs e)
{
    frmtransporteTerrestre obj = new frmtransporteTerrestre();
    obj.ShowDialog();
}

private void toolStripButton4_Click_1(object sender, EventArgs e)
{
    frmAdministrarPuertos onj = new frmAdministrarPuertos();
    onj.ShowDialog();
}

private void toolStripButton5_Click_1(object sender, EventArgs e)
{
    frmTransporteMaritimo obj = new frmTransporteMaritimo();
    obj.ShowDialog();
}

private void toolStripButton6_Click_1(object sender, EventArgs e)
{
    frmImportacion impor = new frmImportacion();
    impor.lbIDUSUARIO.Text = LbIdUsuario.Text;
    impor.Show();
}

private void toolStripButton8_Click_1(object sender, EventArgs e)

```



```

{
    frmAgenciaAduanera obj = new frmAgenciaAduanera();
    obj.ShowDialog();
}

private void toolStripButton7_Click(object sender, EventArgs e)
{
    Close();
}

private void toolStripButton9_Click(object sender, EventArgs e)
{
    frmProveedor objp = new frmProveedor();
    objp.ShowDialog();
}

private void toolStripButton10_Click(object sender, EventArgs e)
{
    frmOrden obji = new frmOrden();
    obji.ShowDialog();
}

```

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Se realizó satisfactoriamente el análisis y diseño de un software para la Empresa “Edifica S.R.L” y para con esto proponer y sistematizar los programas necesarios. El análisis y el diseño del Sistema de Información contempla el control de los documentos que son participes en la importación de una mercadería.
- Para realizar el modelo conceptual y análisis. Luego de haber completado la documentación correspondiente a cada caso de uso hallado en los requerimientos utilizando el método de causa efecto, se procedió a identificar en los escenarios, las

primeras clases para el Diagrama de Clases, el cual funcionó como punto de partida para la siguiente etapa, esto debido a que en el análisis, la construcción de los diagramas de secuencia requieren tener esas clases previamente definidas, para emplearlas y mostrar la interacción que existe entre ellas en cada función del sistema. De esta forma se pudo cumplir los objetivos planteados utilizando para el análisis y diseño de este Sistema de Información la metodología RUP y además con una herramienta de modelado como es UML.

- Modelo de diseño: Finalmente, para este momento se comienza a tener una vista general del sistema, para ver la funcionalidad y definición del mismo. Con el desarrollo del modelo de análisis y diseño se creó niveles para el acceso a los usuarios.
- Con el desarrollo del modelo de análisis y diseño será capaz de prestar información oportuna sobre el seguimiento de los documentos que se generan en una importación.
- Con el desarrollo del modelo de análisis y diseño se presta información oportuna las importaciones realizadas, en una determinada gestión.
- Con el desarrollo del modelo de análisis y diseño se lleva un mejor control de los pagos y solicitudes en cada una de las entidades que intervienen para realizar una importación.

5.2. RECOMENDACIONES

Las recomendaciones que podemos expresar después de la realización de este trabajo son las siguientes:

- ✓ Es recomendable implantar un sistema automatizado de gestión. Tomando en cuenta que dentro de nuestra sociedad y más específicamente en “Edifica S.R.L”, actualmente no existe un sistema confiable controle los documentos y además para realizar un seguimiento sobre una determinada importación. Esto mejorará el trabajo de los empleados encargados de realizar una importación.
- ✓ Frente al excesivo tiempo que conlleva realizar la importación de productos, se recomienda la utilización de las tecnologías de sistemas de información, así como la aplicación de técnicas de [motivación](#) al trabajador.
- ✓ Se recomienda al personal encargado de gestionar la importación a crear un manual de procedimientos para el correcto manejo de los procesos de importación de productos.
- ✓ Se recomienda al administrador del sistema tomar en cuenta los backups de la base de datos dependiendo de las políticas de la empresa pero se debe tener por lo menos uno cada mes.
- ✓ Se recomienda al administrador del sistema tomar en cuenta los roles del personal para asignar las claves de usuario, capacitando al personal para evitar facilitar la contraseña a personas ajenas.

6. GLOSARIO DE TÉRMINOS

Métodos: que se refiere al **medio utilizado para llegar a un fin**.

Herramientas: Es un objeto elaborado a fin de facilitar la realización de una tarea.

Procedimientos: Es la acción de proceder o el **método** de ejecutar algunas cosas. Se trata de una **serie común de pasos definidos**, que permiten realizar un trabajo de forma correcta.

RUP: Rational Unified Process.

UML: Lenguaje de Modelado Unificado.

Iterativo: Término que indica una acción repetitiva.

Funcionalidad: Conjunto de características que hacen que algo sea práctico y utilitario.

Portabilidad: Capacidad de los productos de software para ser fácilmente transportados de un ordenador a otro: la portabilidad de este juego se anula al instalarlo. Propiedad de un programa o aplicación que le permite funcionar bajo distintos sistemas.

Estabilidad: Permanencia o duración en el tiempo.

Interfaz: Zona de comunicación o acción de un sistema sobre otro.

Automatizar: Aplicar procedimientos automáticos a un aparato, proceso o sistema.

Roles: Papel que desempeña una persona o grupo en cualquier actividad.

Amigable: Dícese de la interface de usuario basada en elementos y metáforas gráficas que facilitan la interacción con la computadora y con las aplicaciones que corren en éste.

Ciente: Cualquier elemento de un sistema de información que requiere un servicio mediante el envío de solicitudes al servidor. Cuando dos programas se comunican en una red el cliente es el que inicia la comunicación, mientras que el programa que espera ser contactado es el servidor.

Dato: Unidad mínima de información, sin sentido en sí misma, pero que adquiere significado en conjunción con otras precedentes de la aplicación que las creó.

Hardware: Conjunto de componentes materiales de un sistema informático. Cada una de las partes físicas que forman una computadora, incluidos sus periféricos.

Interface o Interfaz: Conexión e interacción entre hardware, software y el usuario.

Factura Comercial: Emitida por el exportador. Describe las mercancías que constituye la base de la transacción. Indica el precio contenido entre ambas partes y el valor total, las condiciones y la moneda de la transacción.

MIC/DTA: Es el Manifiesto Internacional de Carga/ Documento de Transito Aduanero, se lo utiliza para transporte carretero, desde el lugar donde son cargadas a bordo de un vehículo,

hasta el lugar en donde se descargan, mismo que debe contener todos los datos de importador, fechas, aduana, país de partida, país de destino, transporte, moneda, peso descripción y otros.

Conocimiento de Empaque: También conocido con B/L (Bill of Lading), empleada con mucha frecuencia en los documentos de transporte marítimo. Prueba al recibido de mercancías embarcadas o por embarcar, que establece el estado de las mismas entregas y recibidas a bordo.

Lista de Empaque: Acompaña en todo momento a la factura comercial, esta lista proporciona datos sobre la forma de embalaje de las mercancías, contenido, peso y dimensiones, esencial para las autoridades aduaneras al realizar su inspección y para el cliente para identificar el contenido de la expedición.

Verificadoras: Son empresas autorizadas que hacen la verificación de la mercadería. Los documentos que pide una verificadora para realizar la apertura del servicio de verificación de importación son:

- Parte de recepción
- Manifiesto de carga
- Factura comercial
- Seguro
- Flete.

Certificado de Inspección: Es emitido únicamente por las verificadoras. Una vez aperturado el SVI dan la autorización para que puedan entrar a almacenes emitiendo el Aviso de Conformidad, pudiendo la mercadería ser verificada en origen o en el lugar de destino, sin embargo cuando ingresa al recinto aduanero, a partir de cierto valor, el operador del recinto aduanero exige que la mercadería sea verificada en origen. La emisión de este certificado tiene un valor del 2% del valor de la mercadería.

Formulario 135: Es un formulario gratuito que lo emite la agencia despachante, es una declaración donde se detallan todos los documentos que acompañan la póliza.

Póliza de Importación (Formulario 133): Es la declaración de mercancías de importación. La emiten las agencias aduaneras, haciendo un cobro entre el 3 y el 5%, dependiendo de la mercadería. Además el costo del formulario es de Bs. 50. El Formulario debe contener la siguiente información:

- Nombre del importador

- Nombre de la Agencia Despachante de Aduana.
- El detalle de la Importación (Descripción arancelaria, sub partida arancelaria, valor CIF en frontera)

Impuesto al Valor Agregado (I.V.A): El IVA es un impuesto creado en todo el territorio nacional que se aplicará, en este caso, sobre las importaciones definitivas, y que debe ser pagado por el importador en el momento del despacho aduanero.

Son sujetos pasivos del impuesto quienes realicen a nombre propio importaciones definitivas.

El hecho imponible se perfeccionara en el momento del despacho aduanero, en el caso de importaciones definitivas, e inclusive para los despachos de emergencia.

El IVA a pagar es de 14.94% y se liquidará sobre la base imponible resultante de la suma de los siguientes conceptos: Valor CIF + GAC + otros recargos aduaneros + otras erogaciones necesarias para efectuar el despacho aduanero.

Se consideran “otros cargos aduaneros” a los cargos no arancelarios determinados en las disposiciones generales y multas que se apliquen dentro del Régimen Aduanero de Importación.

Se consideran “otras erogaciones necesarias para efectuar el despacho aduanero” a la comisión de la agencia de aduana, la comisión por servicios de la empresa verificadora y las tarifas cobradas por el recinto aduanero.

Gravamen Arancelario Consolidado (GAC): Para la aplicación de la normativa aduanera y arancelaria de importaciones, la denominación de Gravamen Arancelario Consolidado se sustituye por la de Gravamen Arancelario Consolidado en los reglamentos, procedimientos e instructivos vigentes.

El Gravamen Arancelario Consolidado (GAC) es el arancel que se debe pagar para poder internar y nacionalizar una mercadería, en la alícuota que corresponda a la sub partida arancelaria, aplicándose sobre los valores CIF – Frontera, cuando el transporte sea terrestre y CIF – Aduana cuando sea aéreo, en este caso se consignará el veinticinco por ciento (25%) del flete aéreo.

Los Tributos de importación se aplicarán sobre el valor de transacción, consignado en la factura de reexpedición, incluyendo los costos de flete y seguro de la mercancía desde el lugar de embarque hasta la zona franca de destino.

El GAC a pagar es el establecido en el Arancel Aduanero de Importaciones vigente (el último es el de la Gestión 2010).