Universidad Autónoma de Santo Domingo

Primada de América | Fundada el 28 de octubre del 1538

Facultad de Ciencias

**Asignatura**

Ingeniería de Software II (INF-5230)

**Sección**

Z03

**Profesor**

José Manuel Amado Peralta

**Trabajo**

Proyecto Final

**Estudiante**

Lisbeth Inmaculada Díaz Reinoso - 100340759

Magdelin Encarnación Suazo - 100355496

Linny Rosmery Abreu Vargas - 100159510

23 de Octubre de 2020

Índice

[Introducción 4](#_Toc39942869)

[Planificación y Gerencia de Proyecto 5](#_Toc39942870)

[Ficha General del Proyecto 5](#_Toc39942871)

[Equipo de Trabajo: 5](#_Toc39942872)

[o Coordinador del Proyecto 5](#_Toc39942873)

[o Miembros del Proyecto 5](#_Toc39942874)

[o Organigrama del Proyecto 5](#_Toc39942875)

[o Descripción de las Funciones y responsabilidades 6](#_Toc39942876)

[Descripción General del Proyecto 8](#_Toc39942877)

[o Enfoque del Proyecto 8](#_Toc39942878)

[o Descripción del Proyecto 8](#_Toc39942879)

[o Objetivo General 8](#_Toc39942880)

[o Objetivo Específicos 9](#_Toc39942881)

[o Necesidad del Proyecto 9](#_Toc39942882)

[o Alcance del Proyecto 9](#_Toc39942883)

[o Descripción de los entregables del proyecto 9](#_Toc39942884)

[o Duración del proyecto 10](#_Toc39942885)

[Equipo de Trabajo 10](#_Toc39942886)

[o Coordinador del Proyecto 11](#_Toc39942887)

[o Miembros del Proyecto 11](#_Toc39942888)

[o Organigrama del proyecto 11](#_Toc39942889)

[o Descripción de las Funciones y Responsabilidades 11](#_Toc39942890)

[Descripción de los aspectos técnicos 11](#_Toc39942891)

[o Requerimiento de Hardware 11](#_Toc39942892)

[o Requerimiento de Software 11](#_Toc39942893)

[Matriz de información: actividad, antecedente y tiempo estándar 12](#_Toc39942894)

[Crear cronograma del proyecto. 12](#_Toc39942895)

[Gestión de Requisitos de Software 12](#_Toc39942896)

[Recolectar información (indicar las técnicas usadas para recolectar información) y crear documento sobre la información recopilada. 12](#_Toc39942897)

[Analizar la información y crear documento sobre la información analizada. Para describir vía documentación el proceso completo del negocio tienen que usar Diagrama de Flujo de Datos (DFD) y narrativas de procesos, indicando los departamentos y los roles que intervienen en cada proceso. 12](#_Toc39942898)

[Crear el Documento Visión del Producto. 13](#_Toc39942899)

[o Introducción 13](#_Toc39942900)

[o Objetivo: 13](#_Toc39942901)

[o Alcance: 13](#_Toc39942902)

[o Visión general: 14](#_Toc39942903)

[Crear los casos de uso (crear un caso de uso por cada programa de mediana o alta complejidad). 14](#_Toc39942904)

[Crear Especificaciones de Programas u Opciones Suplementarias para los programas más sencillos. 36](#_Toc39942905)

[Indicar las conclusiones y recomendaciones para el nuevo sistema sobre la base del análisis realizado. 36](#_Toc39942906)

[Diseño del Procesos 37](#_Toc39942907)

[Crear Realización de Caso de Uso 37](#_Toc39942908)

[Crear Diagrama de Secuencia a la Realización de Caso de Uso 37](#_Toc39942909)

[Crear Diagrama de Clases a la Realización de Caso de Uso 56](#_Toc39942910)

[Diseño de la base de datos 56](#_Toc39942911)

[Crear el modelo de datos (Diagrama de Datos) 56](#_Toc39942912)

[Crear el esquema de la base de datos (en un Script) 59](#_Toc39942913)

[Diseñar el prototipo de la aplicación completa, este tiene que contener: 69](#_Toc39942914)

[El menú principal y los diferentes módulos que tiene la aplicación 69](#_Toc39942915)

[Todas las entradas del sistema (pantallas que llama el sistema en los diferentes programas). 69](#_Toc39942916)

[Todas las salidas del sistema, representar la salidas usando pantallas. 69](#_Toc39942917)

[Iniciar la codificación de los Programas 89](#_Toc39942918)

[Conclusiones y recomendaciones finales sobre la propuesta de diseño 106](#_Toc39942919)

[Bibliografía 107](#_Toc39942920)

# Introducción

Este Plan de Desarrollo del Software es una versión preliminar preparada para ser incluida en la propuesta elaborada como respuesta al proyecto de prácticas de la asignatura de Ingeniería de Software II de la Escuela de Informática de la Facultad de Ciencias de la Universidad Autónoma de Santo Domingo. Este documento provee una visión global del enfoque de desarrollo propuesto.

El proyecto ha sido ofertado por Magdelin Encarnación, Lisbeth Díaz y Linny Abreu basado en una metodología de Rational Unified Process en la que únicamente se procederá a cumplir con las tres primeras fases que marca la metodología, constando únicamente en la tercera fase de dos iteraciones. Es importante destacar esto puesto que utilizaremos la terminología RUP en este documento. Se incluirá el detalle para las fases de Inicio y Elaboración y adicionalmente se esbozarán las fases posteriores de Construcción y Transición para dar una visión global de todo proceso.

El enfoque desarrollo propuesto constituye una configuración del proceso RUP de acuerdo a las características del proyecto, seleccionando los roles de los participantes, las actividades a realizar y los artefactos (entregables) que serán generados. Este documento es a su vez uno de los artefactos de RUP.

# Planificación y Gerencia de Proyecto

## Ficha General del Proyecto

|  |  |
| --- | --- |
| Nombre del proyecto: FunParty | |
| Lugar de desarrollo | **C/San Antonio #99, Capotillo** |
| Descripción | **El proyecto permitirá la gestión de**  **el inventario y la facturación de la empresa FunParty** |
| Fecha de inicio y duración | **Inicia el 30/8/2019 hasta el 1/12/2020** |
| Número y tipo de beneficiarios | **Directa: 10 Empleados**  **Indirecta: 1,000 Clientes** |

## Equipo de Trabajo:

### Coordinador del Proyecto

Lisbeth Díaz es la coordinadora, se encarga de la supervisión que todos los plazos se cumplan a tiempo y que el trabajo se realice de manera óptima.

### Miembros del Proyecto

Linny Abreu: Jefe de proyecto

Magdelin Encarnación: Analista de sistema

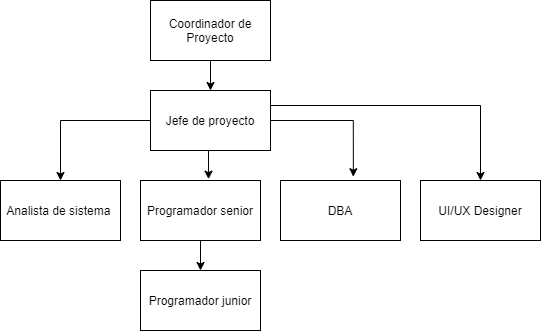
Luis Cortez: Programador senior

Benito Almonte: DBA

Camilo Batista: Programador Junior

Marilenny Soriano: UI/UX Designer

### Organigrama del Proyecto



### Descripción de las Funciones y responsabilidades

|  |  |
| --- | --- |
| Puesto | Responsabilidad |
| Coordinador de proyecto | El Coordinador del proyecto es responsable de ejecutar, supervisar, coordinar las actividades del proyecto, preparar los informes técnicos y revisar los informes financieros del proyecto. |
| Jefe de Proyecto | El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto. |
| Analista de Sistemas | Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos. |
| Programador senior | Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario |
| Programador junior | Ejecutar las tareas de desarrollo de software que se le asignen considerando los objetivos y lineamientos del proyecto. |
| DBA | Administración de la estructura de los datos, implementar restricciones, definir esquema de la DB, diseña la DB del proyecto tanto de manera lógica como física, valida la DB y realiza las respectivas pruebas de inyección SQL y colabora con la integración de la DB a la aplicación. |
| UI/UX Designer | Realiza flujos de trabajo dentro de una aplicación a nivel de mockups, para determinar los diseños que se deben realizar y como se comportara la aplicación, se encarga de realizar el diseño gráfico de las pantallas que componen la aplicación |

## 

## Descripción General del Proyecto

### Enfoque del Proyecto

Este proyecto está enfocado en analizar y diseñar un sistema de información para agilizar y automatizar los procesos de la empresa solicitante, ya que el registro manual de los artículos interrumpe el buen funcionamiento de esta.

### Descripción del Proyecto

El proyecto está diseñado para ayudar a tiendas de ventas y alquileres de artículos para fiestas, organizar y agilizar las ventas, con un inventario para mostrar las ganancias y pérdidas del día, la semana, meses y año.

### Objetivo General

El Desarrollo de un software que cubra los requerimientos de administrar el flujo de los artículos de la empresa de manera más eficaz y eficientes. Se corregirá la pérdida de datos creando una base de datos, se creará un sistema para que la empresa llegue a más personas, tener control de los productos que maneja la empresa, los clientes a los cuales se factura y las ventas realizadas.

### Objetivo Específicos

* Almacenar los datos de los artículos.
* Almacenar los datos de los empleados y los clientes para tener un mejor control.
* Almacenar Pagos para poder generar el inventario.
* Disminuir el tiempo de espera.

### Necesidad del Proyecto

El problema surge de la empresa Party Queens – L&S donde se ve la necesidad de llevar un control de todos los artículos que venden y alquilan, y el registro de los datos de sus empleados y clientes, ya que cada artículo requiere una información detallada, y para así poder llevar un mejor control de las ventas y alquileres. También se requiere de un inventario de las mercancías, la cual se realizan de forma manual, donde no se lleva un control adecuado.

Otros problemas que se enfrentan son: Reportes inadecuados de ventas y alquileres, Productos no disponibles, Agilidad para la atención al cliente, Errores en los reportes de inventario.

En vista de estas necesidades de la empresa que no posee un sistema capaz de desempeñar las tareas expuestas anteriormente, se procederá a diseñar e implementar el software FunParty que cumple con dichos requisitos.

### Alcance del Proyecto

El sistema FunParty será capaz de resolver los inconvenientes que se puede presentar al momento de vender y alquilar algún artículos dando a conocer si está disponible la cantidad necesaria de los artículos que el cliente necesite, también será capaz de agregar, modificar y descontinuar artículos y facturas y llevar el control total de las ventas y alquileres por fecha, especialmente, por día, semanas, meses y años.

Este sistema podrá también tener un catálogo de todos los artículos que están en la tienda y llevar el control de los datos de los empleados y clientes.

### Descripción de los entregables del proyecto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificación Entregable | Descripción Entregable | Fecha de entrega | Lugar de entrega | Condiciones satisfacción |
| E01 | Solicitud de Sistema | 28/01/2021 | Oficina Principal | El tiempo de entrega |
| E02 | Definición del proyecto | 14/02/2021 | Oficina Principal | Facilidad de entendimiento |
| E03 | Diagramas de actividad | 14/03/2021 | Oficina Principal | Claro y preciso |
| E04 | Tabla de eventos | 31/03/2021 | Oficina Principal | Claro y preciso |
| E05 | Diagramas de casos de usos | 30/04/2021 | Oficina | Facilidad de entrenamiento |
| E06 | Especificaciones de casos de uso | 15/05/2021 | Oficina  Principal | Facilidad de entendimiento |
| E07 | Diagrama UML de clases | 31/05/2021 | Oficina Principal | Claro y preciso |
| E08 | Diagramas de secuencia | 30/06/2021 | Oficina Principal | Facilidad de entendimiento |
| E09 | Diseño de la base de datos | 31/07/2021 | Oficina Principal | Calidad |
| E10 | Tarjetas CRC | 15/08/2021 | Oficina Principal | Claro y preciso |
| E11 | Diseño de GUI | 15/09/2021 | Oficina Principal | Calidad |
| E12 | Diagramas de secuencia detallado de tres capas y diagrama de paquete | 15/10/2021 | Oficina Principal | Facilidad de entendimiento |
| E13 | SinglentonDA Layer | 31/10/2021 | Oficina Principal | Calidad |
| E14 | Diseño de clases detallado | 15/11/2021 | Oficina Principal | Calidad |
| E15 | Desarrollo del código | 28/02/2022 | Oficinal Principal | Calidad |

### Duración del proyecto

El proyecto FunParty tuvo una duración aproximadamente de un año y medio.

## Descripción de los aspectos técnicos

### Requerimiento de Hardware

* Procesador Intel Core i3
* Memoria RAM 4 GB
* 400 GB de Disco Duro
* Mouse
* Teclado
* Impresora
* Sistema Operativo Microsoft Windows 10 Home o Profesional
* SQL Server 2008 R2

### Requerimiento de Software

**Los requerimientos funcionales del usuario**

* Crear, actualizar y descontinuar datos de los artículos.
* Crear, actualizar y descontinuar facturas.
* Crear el registro de los clientes, solicitando mínimo el nombre si es venta y nombre, dirección, teléfono si es alquiler.
* Poder visualizar información de la base de datos.
* Las facturas de ventas y alquileres tendrán identificadores únicos.
* Realizar informes del día, semanales, mensuales y anuales de las ganancias y pérdidas.
* Tener un login en el cual solo pueda acceder los usuarios autorizados.
* Crear y actualizar los datos de los empleados.

**Los requerimientos no-funcionales del usuario**

* Responder al usuario en menos de 5 segundos las funcionalidades del sistema.
* Operar adecuadamente con al menos 3 usuarios con sesiones.
* Permitir modificar los datos antes de 30 minutos después de crearlas.
* Aplicar la seguridad de los datos.
* Instrucciones de uso para permitir el aprendizaje y el manejo al usuario con un tiempo menor de 5 horas.
* Proporcionar mensajes de error.
* Proporcionar mensajes de advertencias al modificar y descontinuar algún dato.
* Poseer interfaces graficas bien formadas.
* Tener el módulo de ayuda.

## Matriz de información: actividad, antecedente y tiempo estándar

## Crear cronograma del proyecto.

# Gestión de Requisitos de Software

Recolectar información (indicar las técnicas usadas para recolectar información) y crear documento sobre la información recopilada.

Analizar la información y crear documento sobre la información analizada. Para describir vía documentación el proceso completo del negocio tienen que usar Diagrama de Flujo de Datos (DFD) y narrativas de procesos, indicando los departamentos y los roles que intervienen en cada proceso.

|  |  |  |
| --- | --- | --- |
| NOMBRE | Personal de Inventario | |
| CASO(S) DE USO AL (A LOS) QUE PERTENECE | | **CU-100, CU-110, CU-120, CU-190** |
| DESCRIPCION | | |
| Representa a un empleado que será el encargado de vigilar y dirigir cualquier actividad dentro del almacén. Es el responsable de modificar, ingresar y/o descontinuar alguna entrada de la mercancía en el inventario del sistema y realizar infirme de la mercancía ingresada. Por norma, debe iniciar sesión con su cuenta de empleado que le otorga dichos accesos para llevar a cabo cada una de sus funciones dentro del sistema. | | |

|  |  |  |
| --- | --- | --- |
| NOMBRE | Cajera | |
| CASO(S) DE USO AL (A LOS) QUE PERTENECE | | **CU-130, CU-160, CU-180, CU-190** |
| DESCRIPCION | | |
| Representa a un empleado que se encarga de la venta de las mercancías. Por lo general, tiene acceso a crear una factura, tener indicador único las facturas e ingresar los clientes. Este también tiene la responsabilidad de entregar dicha factura al cliente, junto con la mercancía vendida. Por norma, debe iniciar sesión para realizar sus funciones dentro del sistema. | | |

|  |  |  |
| --- | --- | --- |
| NOMBRE | Supervisor de Venta | |
| CASO(S) DE USO AL (A LOS) QUE PERTENECE | | **CU-140, CU-150, CU-170** |
| DESCRIPCION | | |
| Representa a un empleado que será el responsable de vigilar y dirigir cualquier actividad en el área de ventas. Tiene el poder de modificar y/o descontinuar una factura ya creada, además de ser el encargado de solicitar el informe de ventas al sistema, el cual queda bajo su supervisión y revisión. Por norma, debe iniciar sesión para realizar sus funciones dentro del sistema. | | |

|  |  |  |
| --- | --- | --- |
| NOMBRE | Gerente | |
| CASO(S) DE USO AL (A LOS) QUE PERTENECE | | **CU-170, CU-210, CU-220** |
| DESCRIPCION | | |
| Representa a un empleado que será el responsable de vigilar y dirigir cualquier actividad en la empresa. Es el responsable de registrar los empleados y de visualizar todos los datos del sistema. Por norma, debe iniciar sesión para realizar sus funciones dentro del sistema. | | |

## Crear el Documento Visión del Producto.

### Introducción

En esta visión del proyecto presentaremos el objetivo, el alcance entre otras referencias concerniente al proyecto que estamos realizando.

### Objetivo:

Presentar la visión del proyecto que estamos desarrollando para tener una idea de hacia dónde nos estamos dirigiendo.

### Alcance:

El proyecto que estamos realizando alcanzara a resolver las necesidades más profundas de cada uno de nuestros clientes.

### Visión general:

Este documento contiene la visión general del proyecto a realizar para la empresa de FunParty, a la cual se le está haciendo un sistema de información llamado FunParty.

Este documento se organiza de tal manera que los lectores puedan entender el software que se está realizando, el cual servirá para aligerar la carga que se tiene hoy en día en la empresa y para que no se estén perdiendo los datos por la informalidad del negocio.

## Crear los casos de uso (crear un caso de uso por cada programa de mediana o alta complejidad).

**CU-100**

|  |  |
| --- | --- |
| ID | CU-100 |
| Nombre | Ingresar Artículos |
| Versión | V0 |
| Actores | Personal de Inventario |
| Propósito | Ingresar los artículos al sistema |
| Descripción | El personal de inventario podrá acceder al sistema mediante inicio de sesión para poder registrar los nuevos artículos. |
| Nivel | Primario |
| Precondiciones | Tener los artículos revisados y verificados |
| Poscondiciones | Tener inventario con los artículos registrados |
| Desencadenador | El personal de inventario ingresa la mercancía al sistema |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el personal de inventario puede ingresar mercancías al sistema |
| Prioridad en la programación | Primero |
| Frecuencia | Cada vez que se necesite ingresar un articulo |
| Niveles superiores | Iniciar Sesión |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Personal de Inventario recibe y verifica la mercancía |  |  |
| 2 | El Personal de Inventario selecciona ingresar artículos |  |  |
| 3 | El personal de Inventario inicia el registro de los artículos | 4 | El sistema requiere una confirmación |
| 5 | El Personal de Inventario confirma el registro del nuevo articulo | 6 | El sistema almacena los datos |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | El Personal de Inventario no registra completo los datos del articulo | 2 | El sistema no valida los datos, ni permite continuar y muestra mensaje de falta información |
|  | 3 | Regresar al paso 3 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | El Personal de Inventario cancela la ejecución del caso de uso | 2 | El sistema no almacena el articulo y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-110**

|  |  |
| --- | --- |
| ID | CU-110 |
| Nombre | Modificar Articulo |
| Versión | V0 |
| Actores | Personal de Inventario |
| Propósito | Cambiar o actualizar algún artículo del sistema |
| Descripción | El personal de inventario podrá acceder al sistema mediante inicio de sesión para poder actualizar o cambiar algún articulo |
| Nivel | Primario |
| Precondiciones | Tener artículos registrados |
| Poscondiciones | Modificar artículos |
| Desencadenador | Personal de inventario modifica los artículos del sistema |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Personal de Inventario puede modificar los artículos. |
| Prioridad en la programación | Segundo |
| Frecuencia | Cada vez que se necesite alterar algún detalle de los artículos. |
| Niveles superiores | Ingresar Articulo |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Personal de inventario navega los artículos |  |  |
| 2 | El Personal de Inventario selecciona un articulo |  |  |
| 3 | El Personal de Inventario inicia la modificación del artículo. | 4 | El sistema requiere una confirmación. |
| 5 | El personal de Inventario confirma la modificación del artículo. | 6 | El sistema modifica el artículo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Personal de Inventario filtra los artículos. |  |  |
|  | 2 | Regresar al paso 2 del escenario principal. |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Personal de Inventario cancela la ejecución del caso de uso | 2 | El sistema no modifica el artículo y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-120**

|  |  |
| --- | --- |
| ID | CU-120 |
| Nombre | Descontinuar Articulo |
| Versión | V0 |
| Actores | Personal de Inventario |
| Propósito | Descontinuar artículos del Sistema |
| Descripción | El Personal de Inventario podrá acceder al sistema mediante Inicio de Sesión para descontinuar algún artículo ya registrado en el inventario del sistema. |
| Nivel | Primario |
| Precondiciones | Tener la mercancía ya registrada |
| Poscondiciones | Descontinuar la mercancía alterada |
| Desencadenador | Personal de Inventario descontinua los artículos en el sistema |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Personal de Inventario puede descontinuar los artículos del sistema |
| Prioridad en la programación | Segundo |
| Frecuencia | Cada vez que se necesite descontinuar algún articulo |
| Niveles superiores | Ingresar Artículo |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Personal de inventario navega los artículos |  |  |
| 2 | El Personal de Inventario selecciona un articulo |  |  |
| 3 | El Personal de Inventario inicia la descontinuación del artículo. | 4 | El sistema requiere una confirmación. |
| 5 | El personal de Inventario confirma la descontinuación del artículo. | 6 | El sistema descontinua el artículo. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Personal de Inventario filtra los artículos |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Personal de Inventario cancela la ejecución del caso de uso | 2 | El sistema no descontinua el articulo y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-130**

|  |  |
| --- | --- |
| ID | CU-130 |
| Nombre | Crear Factura |
| Versión | V0 |
| Actores | Cajera |
| Propósito | Crear nueva factura de venta |
| Descripción | La cajera podrá acceder al sistema mediante inicio de sesión para la realización de nuevas facturas con los artículos que serán vendidos |
| Nivel | Primario |
| Precondiciones | Tener la mercancía a vender ya registrada |
| Poscondiciones | Tener constancias para fines de informes y entregar la factura a los clientes |
| Desencadenador | La cajera crea facturas de ventas |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo la cajera crea las facturas |
| Prioridad en la programación | Primero |
| Frecuencia | Cada vez que se necesite crear una factura |
| Niveles superiores | Iniciar Sesión |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | La Cajera inicia sesión en el sistema | 2 | El sistema solicita la acción deseada |
| 3 | La Cajera recibe los artículos que serán vendidos |  |  |
| 4 | La Cajera ingresa los datos y crea la factura | 5 | El sistema almacena la factura |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | La Cajera filtra los artículos |  |  |
|  | 2 | Regresar al paso 3 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 4.a | |  |  |  |
|  | 1 | La Cajera cancela la ejecución del caso de uso | 2 | El sistema no crea, ni almacena la factura y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-140**

|  |  |
| --- | --- |
| ID | CU-140 |
| Nombre | Modificar Factura |
| Versión | V0 |
| Actores | Supervisor de Ventas |
| Propósito | Modificar las facturas |
| Descripción | El Supervisor de Ventas podrá iniciar sesión en el sistema para poder cambiar o actualizar algún detalle en la factura ya registrada. |
| Nivel | Primero |
| Precondiciones | Tener la factura ya registrada |
| Poscondiciones | Modificar la factura alterada |
| Desencadenador | El Supervisor de ventar modifica las facturas |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Supervisor de Ventas puede modificar la factura. |
| Prioridad en la programación | Segunda |
| Frecuencia | Cada vez que se necesite alterar algún detalle de la factura. |
| Niveles superiores | Crear Factura |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Supervisor de Ventas navega las facturas |  |  |
| 2 | El Supervisor de Ventas selecciona una factura |  |  |
| 3 | El Supervisor de Ventas inicia la modificación de la factura. | 4 | El sistema requiere una confirmación. |
| 5 | El Supervisor de Ventas confirma la modificación de la factura. | 6 | El sistema modifica la factura. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Supervisor de Ventas filtra los artículos |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Supervisor de Ventas cancela la ejecución del caso de uso | 2 | El sistema no modifica la factura y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-150**

|  |  |
| --- | --- |
| ID | CU-150 |
| Nombre | Descontinuar Facturas |
| Versión | V0 |
| Actores | Supervisor de Ventas |
| Propósito | Descontinuar o anular las facturas del sistema |
| Descripción | El Supervisor de Ventas podrá acceder al sistema mediante inicio de sesión para descontinuar una factura ya antes registrada en el sistema. |
| Nivel | Primero |
| Precondiciones | Tener la factura ya registrada |
| Poscondiciones | Descontinuar la factura del sistema |
| Desencadenador | El Supervisor de Ventas descontinua factura en el sistema |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Supervisor de Ventas puede descontinuar la factura. |
| Prioridad en la programación | Segundo |
| Frecuencia | Cada vez que se necesite descontinuar alguna factura |
| Niveles superiores | Crear Factura |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Supervisor de Ventas navega las facturas |  |  |
| 2 | El Supervisor de Ventas selecciona una factura |  |  |
| 3 | El Supervisor de Ventas inicia la descontinuación de la factura. | 4 | El sistema requiere una confirmación. |
| 5 | El Supervisor de Ventas confirma la descontinuación de la factura. | 6 | El sistema descontinua la factura. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Supervisor de Ventas filtra los artículos |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Supervisor de Ventas cancela la ejecución del caso de uso | 2 | El sistema no descontinua la factura y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-160**

|  |  |
| --- | --- |
| ID | CU-160 |
| Nombre | Registrar Cliente |
| Versión | V0 |
| Actores | Cajera |
| Propósito | Registrar nuevos clientes |
| Descripción | La Cajera debe iniciar sesión para poder registrar los datos necesarios del cliente. |
| Nivel | Primario |
| Precondiciones | La Cajera debe de tener los datos del cliente |
| Poscondiciones | Se deben guardar los datos registrados |
| Desencadenador | La Cajera debe de crear nuevo cliente |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo la cajera puede registrar los clientes en el sistema. |
| Prioridad en la programación | Segunda |
| Frecuencia | Cada vez que haya un nuevo cliente |
| Niveles superiores | Iniciar Sesión |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | La Cajera navega los Clientes |  |  |
| 2 | La Cajera selecciona un nuevo cliente | 3 | El sistema genera un formulario |
| 4 | La Cajera ingresa los datos del cliente |  |  |
| 5 | La Cajera guarda los datos del cliente. | 6 | El sistema requiere una confirmación. |
| 7 | La Cajera confirma el nuevo cliente | 8 | El sistema guarda los datos |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 2.a | |  |  |  |
|  | 1 | Si la cajera no completa el formulario | 2 | Sistema muestra mensaje que incompleto |
|  | 3 | Regresar al paso 4 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 4.a | |  |  |  |
|  | 1 | La Cajera cancela la ejecución del caso de uso | 2 | El sistema no guarda la información del cliente y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-170**

|  |  |
| --- | --- |
| ID | CU-170 |
| Nombre | Visualizar Datos |
| Versión | V0 |
| Actores | Supervisor de ventas, Gerente |
| Propósito | Ver los datos registrados en el sistema |
| Descripción | El Supervisor de Ventas y el Gerente deben iniciar sesión para poder ver los datos que están registrados en el sistema. |
| Nivel | Primario |
| Precondiciones | El Supervisor de Ventas y Gerente deben haber iniciado sesión |
| Poscondiciones | Ver los datos ya antes registrados |
| Desencadenador | El Supervisor de Ventas y al Gerente visualizara los datos del sistema |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Supervisor de Ventas y el Gerente pueden visualizar los datos del sistema |
| Prioridad en la programación | Segunda |
| Frecuencia | Cada vez que se necesite ver los datos registrados |
| Niveles superiores | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Supervisor de Ventas o Gerentes navegan los datos |  |  |
| 2 | El Supervisor de Ventas o Gerente selecciona lo que desea visualizar | 3 | El sistema requiere una confirmación |
| 4 | El Supervisor de Ventas o Gerente confirman visualizar | 5 | El sistema muestra los datos |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Gerente o el Supervisor de Ventas cancela la ejecución del caso de uso | 2 | El sistema no muestra los datos y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-180**

|  |  |
| --- | --- |
| ID | CU-180 |
| Nombre | Tener Identificador |
| Versión | V0 |
| Actores | Cajera |
| Propósito | Tener ID las facturas |
| Descripción | La cajera debe verificar que el identificador de las facturas este correcto |
| Nivel | <<include>> |
| Precondiciones | Tener la mercancía a vender ya registrada |
| Poscondiciones | Crear factura |
| Desencadenador | La cajera verifica el identificador de las facturas |
| Caso de uso <<include>> | N/A |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo la cajera puede verificar el identificador |
| Prioridad en la programación | Primero |
| Frecuencia | Cada vez que se necesite crear una factura |
| Niveles superiores | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | La Cajera entra a crear factura | 2 | El sistema le genera un identificador |
| 3 | La Cajera verifica que el identificador este correcto |  |  |
| 4 | La Cajera continua con la creación de la factura |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | Si la factura no tiene el identificador correcto |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 4.a | |  |  |  |
|  | 1 | La Cajera cancela la ejecución del caso de uso | 2 | El sistema no continua con la creación de la factura y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-190**

|  |  |
| --- | --- |
| ID | CU-190 |
| Nombre | Realizar informe |
| Versión | V0 |
| Actores | Personal de inventario, Cajera |
| Propósito | Realizar informe de las ventas y los artículos |
| Descripción | El Personal de Inventario y la Cajera podrán acceder al sistema mediante inicio de sesión para realizar el informe ya sea, diario, semanal o mensual. |
| Nivel | Primario |
| Precondiciones | Disponer de los artículos registrados y de las ventas realizadas |
| Poscondiciones | Controlar la entrada de las mercancías, las ventas realizadas, las ganancias y las perdidas. |
| Desencadenador | El Personal de Inventario realiza el informe delos artículos ingresados y la cajera realiza el de las ventas. |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Personal de inventario y la Cajera pueden realizar el informe |
| Prioridad en la programación | Primero |
| Frecuencia | Cada vez que se necesite un reporte de las ventas y los artículos ingresados |
| Niveles superiores | Crear Factura, Ingresar Artículos |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Personal de inventario o la Cajera genera informe de ventas | 2 | Sistema requiere periodicidad |
| 3 | El Personal de Inventario o Cajera ingresa el periodo | 4 | El sistema solicita confirmación de realizar informe |
| 5 | El Personal de Inventario o cajera confirma realizar informe. | 6 | El Sistema realiza el informe |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | El Personal de inventario o Cajera no ingresan correctamente el periodo |  |  |
|  | 2 | Regresar al paso 3 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | La Cajera o El Personal de Inventario cancela la ejecución del caso de uso | 2 | El sistema no realiza el informe y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-200**

|  |  |
| --- | --- |
| ID | CU-200 |
| Nombre | Iniciar Sesión |
| Versión | V0 |
| Actores | Personal de Inventario, Cajera, Personal de Ventas, Gerente |
| Propósito | Identificarse antes de acceder al sistema |
| Descripción | Solicitar inicio de sesión antes de poder acceder al sistema, para poder identificar que tarea le toca a cada actor. |
| Nivel | <<include>> |
| Precondiciones | Tener el sistema abierto. |
| Poscondiciones | Permitir el acceso deseado al actor que inicie sesión. |
| Desencadenador | El sistema solicita inicio de sesión |
| Caso de uso <<include>> | N/A |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo los que tengan la cuenta, pueden iniciar sesión. |
| Prioridad en la programación | Primero |
| Frecuencia | Siempre |
| Niveles superiores | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El usuario abre el sistema | 2 | El sistema requiere inicio de sesión |
| 3 | El usuario ingresa sus credenciales | 4 | El sistema valida los datos |
|  |  | 5 | El sistema permite el acceso |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | El usuario ingresa sus credenciales incorrecta | 2 | El sistema no valida los datos, ni permite continuar y muestra mensaje de usuario o contraseña incorrecta |
|  | 3 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 3.a | |  |  |  |
|  | 1 | El usuario olvida sus credenciales | 2 | El sistema no valida los datos, ni permite continuar y regresa a la pantalla anterior para recuperar sus credenciales de acceso. |
|  | 3 | Regresar al paso 2 del escenario principal |  |  |

**CU-210**

|  |  |
| --- | --- |
| ID | CU-210 |
| Nombre | Registrar Empleados |
| Versión | V0 |
| Actores | Gerente |
| Propósito | Registrar los datos de los empleados |
| Descripción | El Gerente debe iniciar sesión para poder registrar los datos del empleado y le otorgara su usuario y contraseña para poder acceder al sistema. |
| Nivel | Primario |
| Precondiciones | El Gerente debe de tener los datos del empleado |
| Poscondiciones | Se deben guardar los datos registrados |
| Desencadenador | El Gerente debe de crear nuevo empleado |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo el Gerente puede registrar los empleados en el sistema. |
| Prioridad en la programación | Segunda |
| Frecuencia | Cada vez que haya un nuevo empleado |
| Niveles superiores | Iniciar Sesión |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | El Gerente selecciona crear empleado | 2 | El Sistema proporciona un formulario |
| 2 | El Gerente ingresa los datos del Empleado |  |  |
| 3 | El Gerente guarda los datos | 4 | El sistema requiere una confirmación. |
| 5 | El Gerente confirma guardar | 6 | El sistema guarda los datos |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Gerente no ingresan correctamente los datos |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | El Gerente cancela la ejecución del caso de uso | 2 | El sistema registra el empleado y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

**CU-230**

|  |  |
| --- | --- |
| ID | CU-230 |
| Nombre | Imprimir Factura |
| Versión | V0 |
| Actores | Cajera, Sistema |
| Propósito | Imprimir factura para el cliente |
| Descripción | La cajera podrá acceder al sistema mediante inicio de sesión para poder imprimir la factura ya creada. |
| Nivel | Primero |
| Precondiciones | Tener la factura creada |
| Poscondiciones | Tener constancias para fines de informes y entregar la factura al cliente. |
| Desencadenador | La cajera manda a imprimir la factura |
| Caso de uso <<include>> | Iniciar Sesión |
| Caso de uso <<extend>> | N/A |
| Condiciones especiales | Solo la Cajera puede imprimir la factura del sistema |
| Prioridad en la programación | Primero |
| Frecuencia | Cada vez que se necesite imprimir una factura |
| Niveles superiores | Crear Factura |

|  |  |  |  |
| --- | --- | --- | --- |
| FLUJO DE EVENTOS | | | |
| FLUJO PRINCIPAL DE EVENTOS | | | |
| ACCION DEL ACTOR | | **RESPUESTA DEL SISTEMA** | |
| # |  | **#** |  |
| 1 | La cajera abre la factura |  |  |
| 2 | La cajera inicia la impresión de la factura | 3 | El sistema solicita confirmación de impresión |
| 4 | La cajera confirma la impresión | 5 | El sistema imprime la factura |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) ALTERNATIVO(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 1.a | |  |  |  |
|  | 1 | El Supervisor de Ventas filtra los artículos |  |  |
|  | 2 | Regresar al paso 2 del escenario principal |  |  |

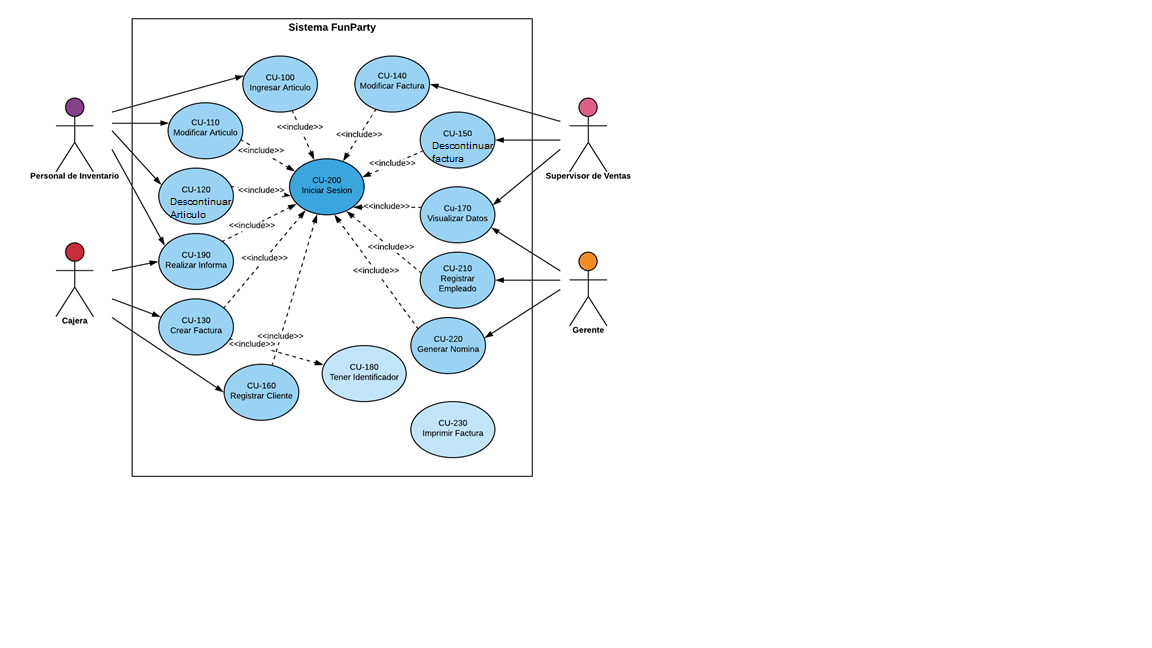
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FLUJO(S) EXCEPCIONAL(S) DE EVENTOS | | | | |
| ACCION DEL ACTOR | | | **RESPUESTA DEL SISTEMA** | |
| # | |  | **#** |  |
| 5.a | |  |  |  |
|  | 1 | La Cajera cancela la ejecución del caso de uso | 2 | El sistema no imprime la factura y la ejecución del caso de uso termina regresando a la pantalla anterior |
|  | 3 | Regresa al siguiente paso del escenario principal |  |  |

## Crear Especificaciones de Programas u Opciones Suplementarias para los programas más sencillos.

## Indicar las conclusiones y recomendaciones para el nuevo sistema sobre la base del análisis realizado.

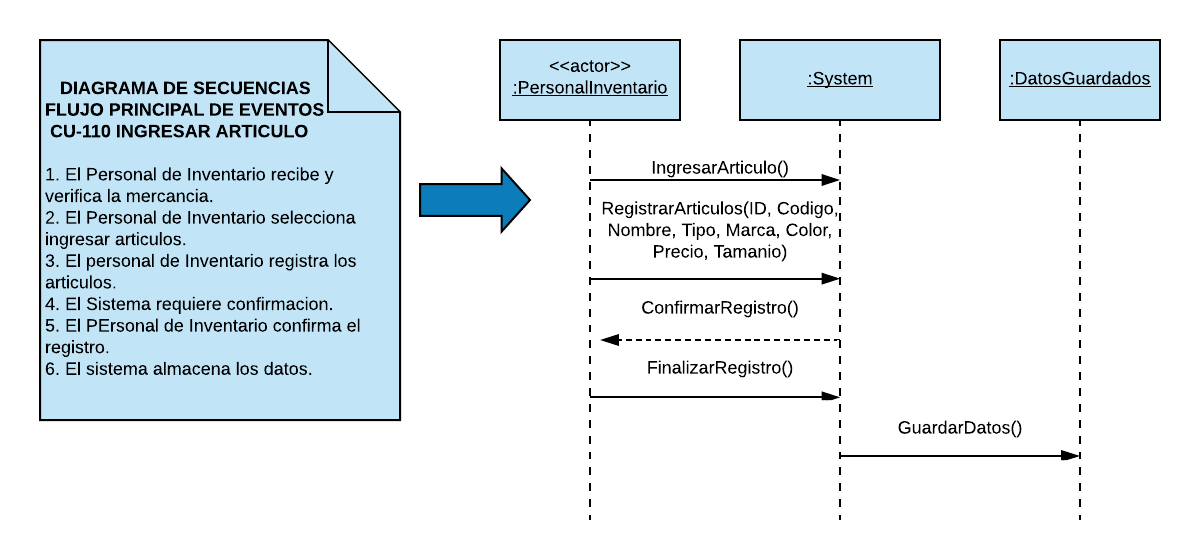
# Diseño del Procesos

## Crear Realización de Caso de Uso

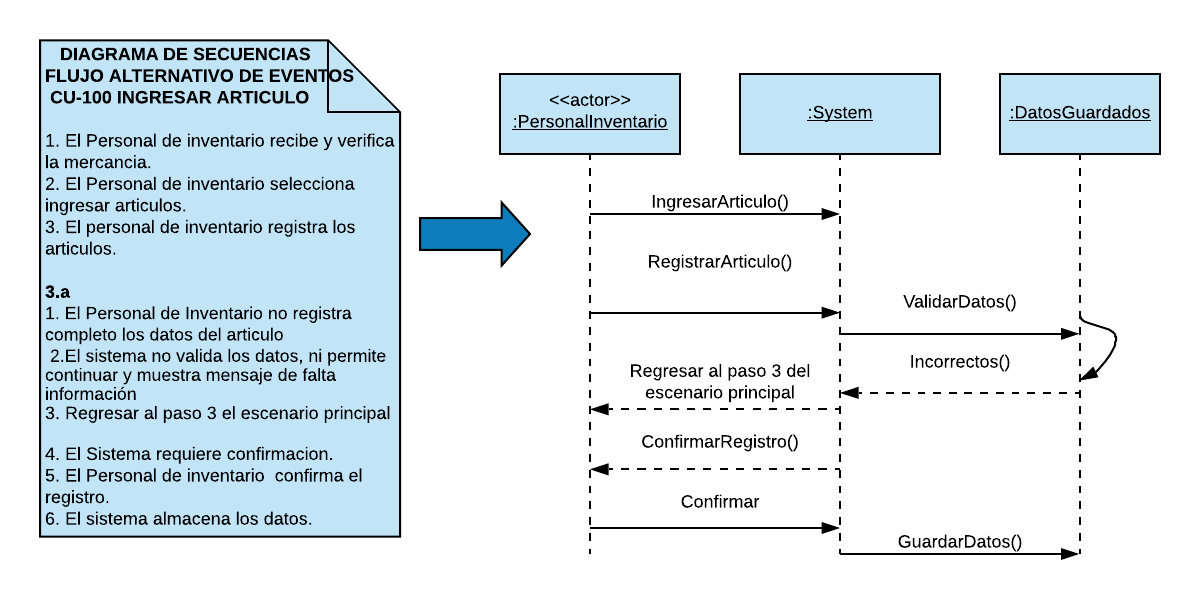
****

## Crear Diagrama de Secuencia a la Realización de Caso de Uso

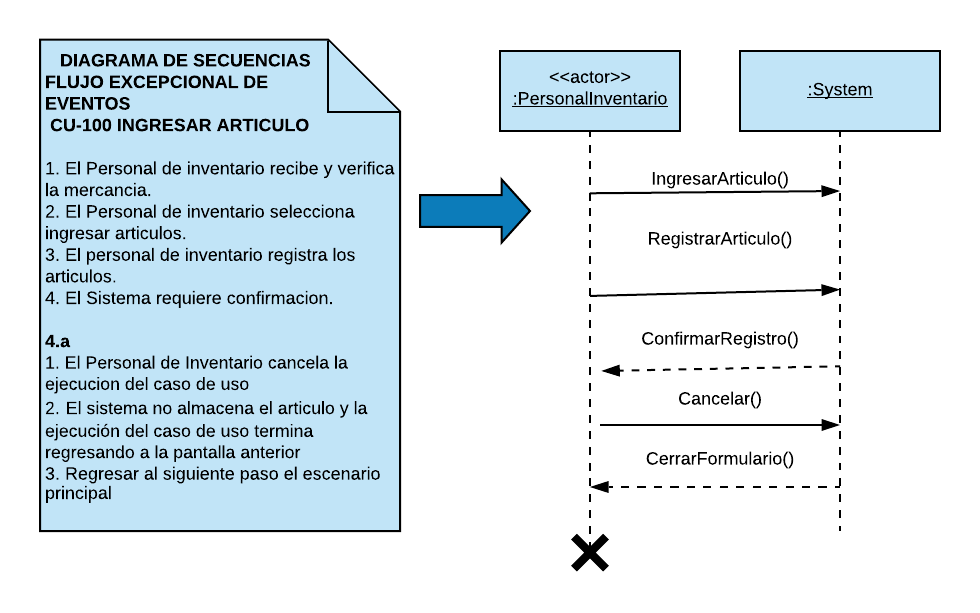
* 1. **CU-100 Ingresar Artículos**
     1. **Flujo Principal de Eventos**

****

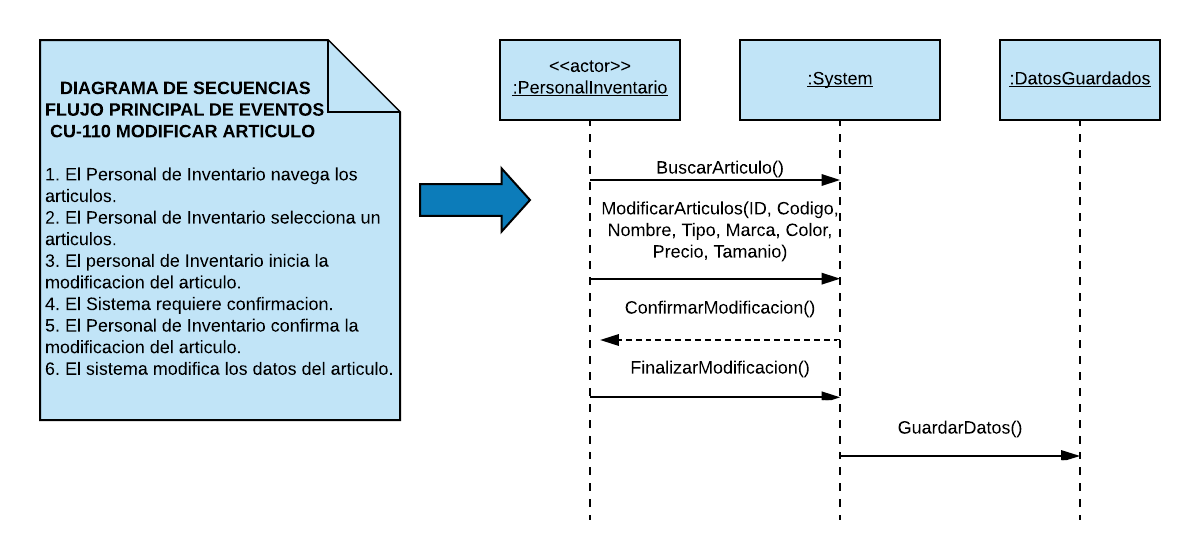
* + 1. **Flujo Alternativo de Eventos**

****

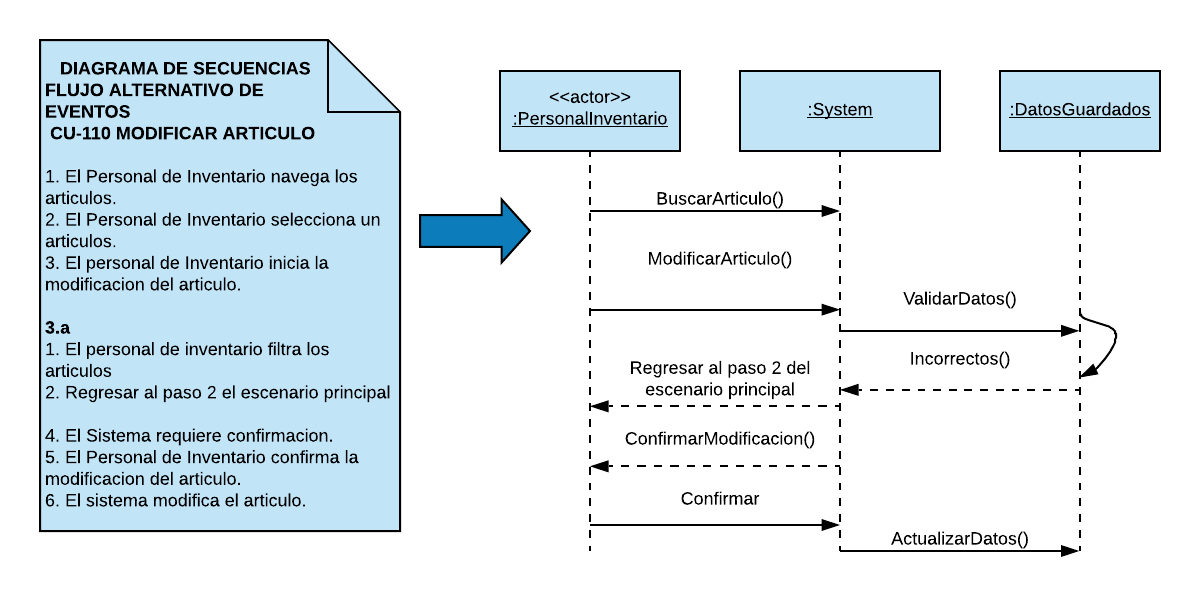
* + 1. **Flujo Excepcional de Eventos**

****

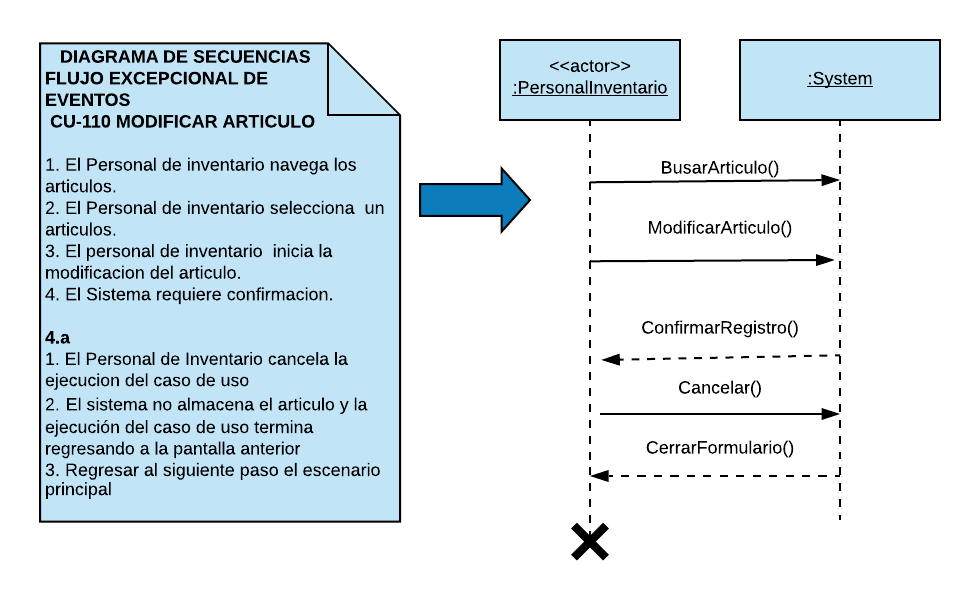
* 1. **CU-110 Modificar Artículos**
     1. **Flujo Principal de Eventos**

****

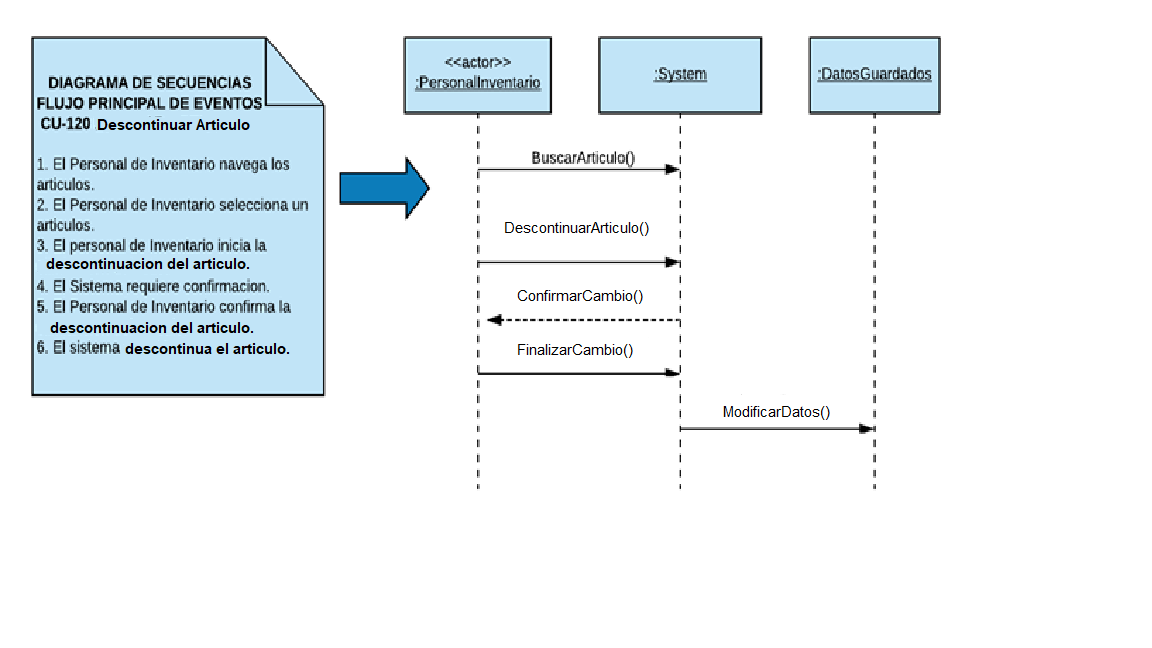
* + 1. **Flujo Alternativo de Eventos**

****

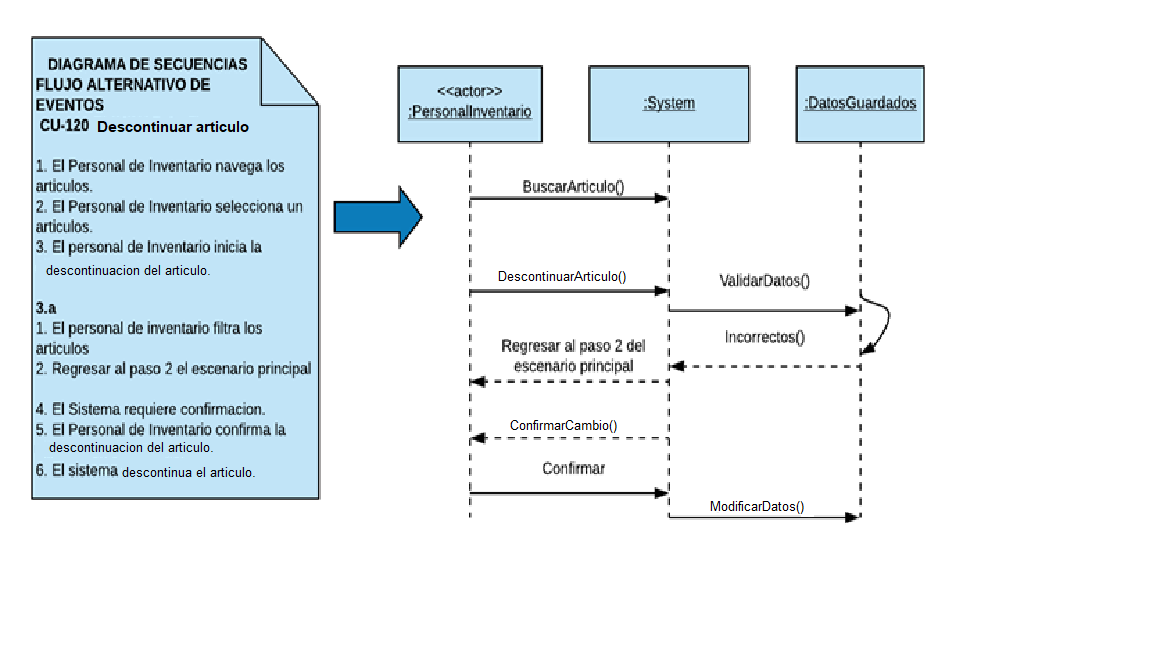
* + 1. **Flujo Excepcional de Eventos**

****

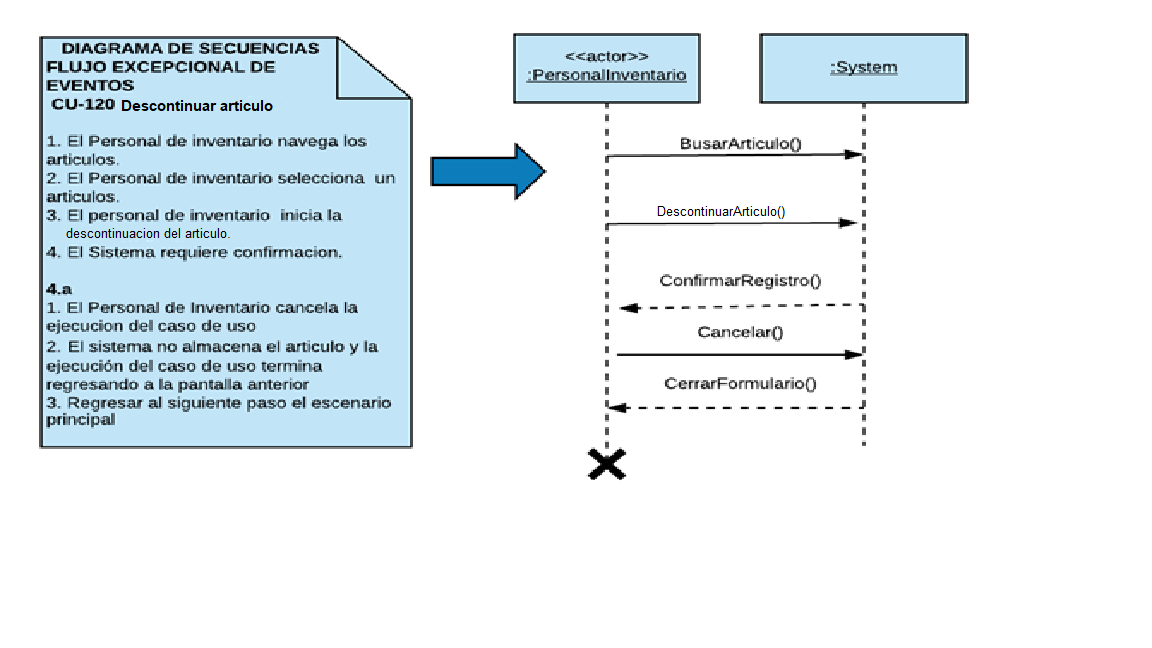
* 1. **CU-120 Descontinuar Artículos**
     1. **Flujo Principal de Eventos**

****

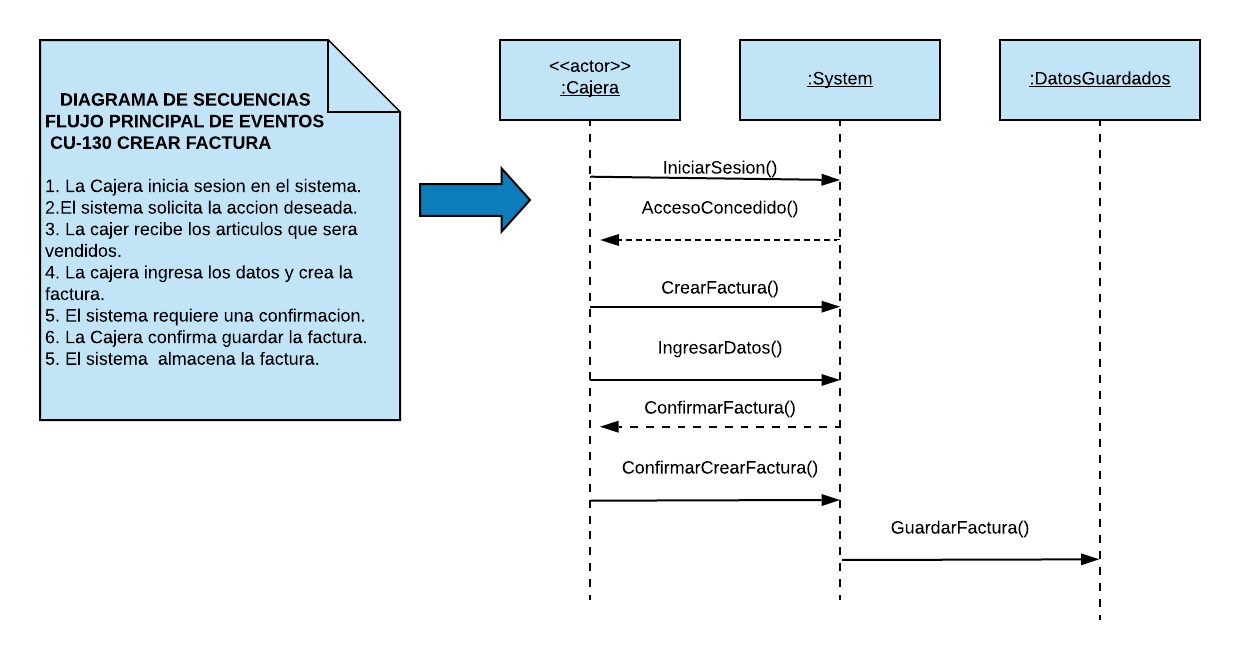
* + 1. **Flujo Alternativo de Eventos**

****

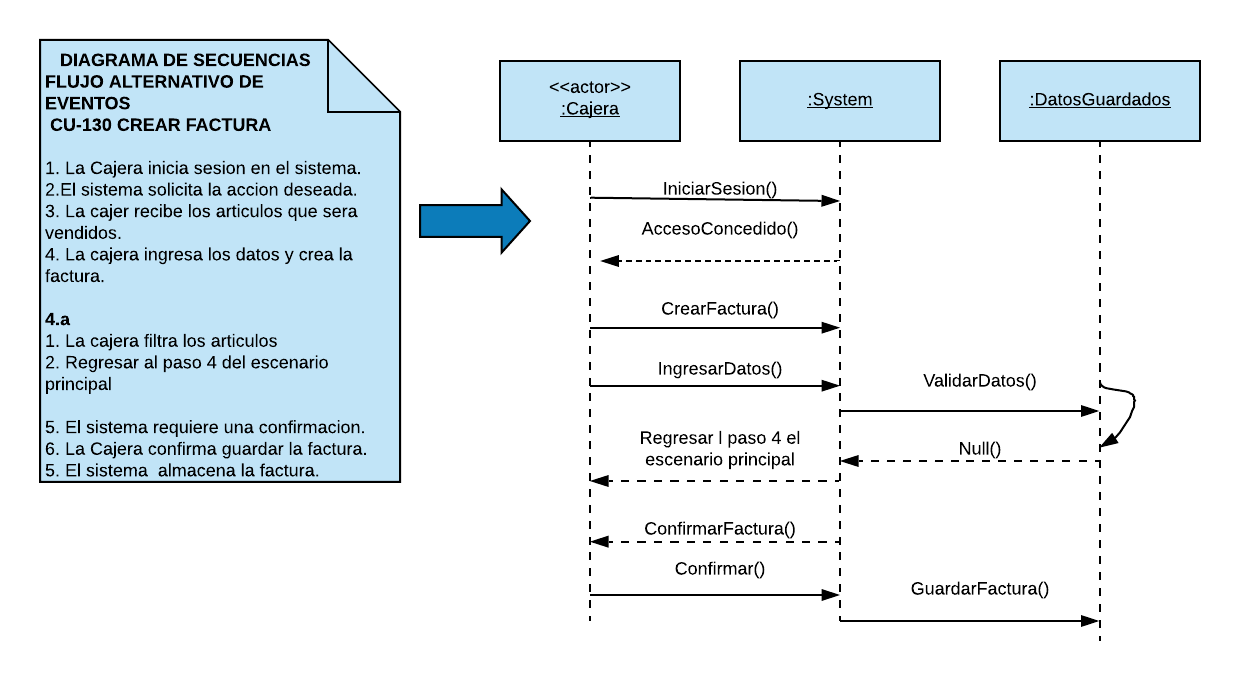
* + 1. **Flujo Excepcional de Eventos**

****

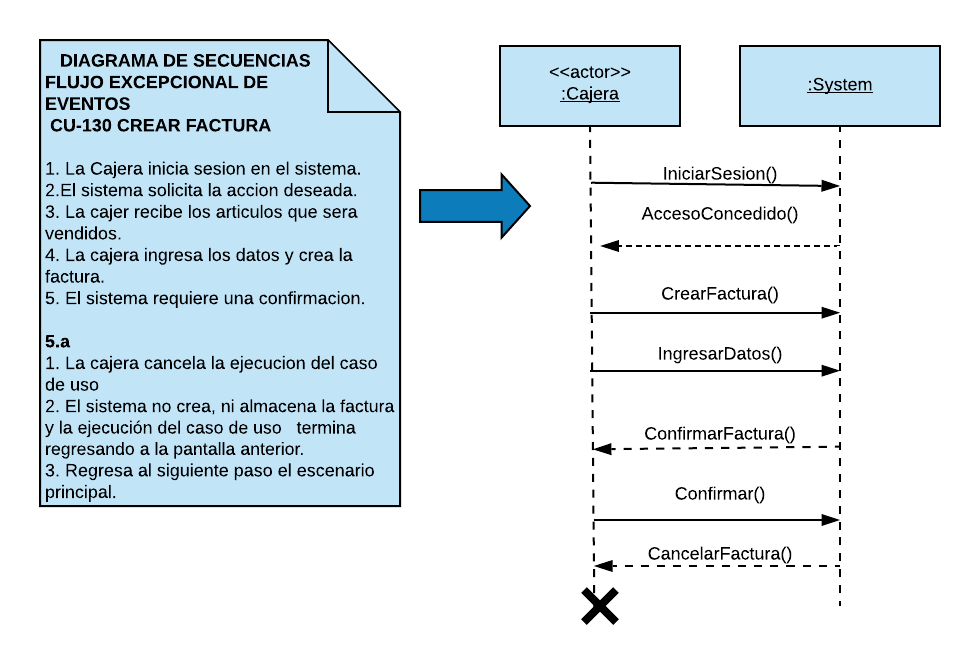
* 1. **CU-130 Crear Factura**
     1. **Flujo Principal de Eventos**

****

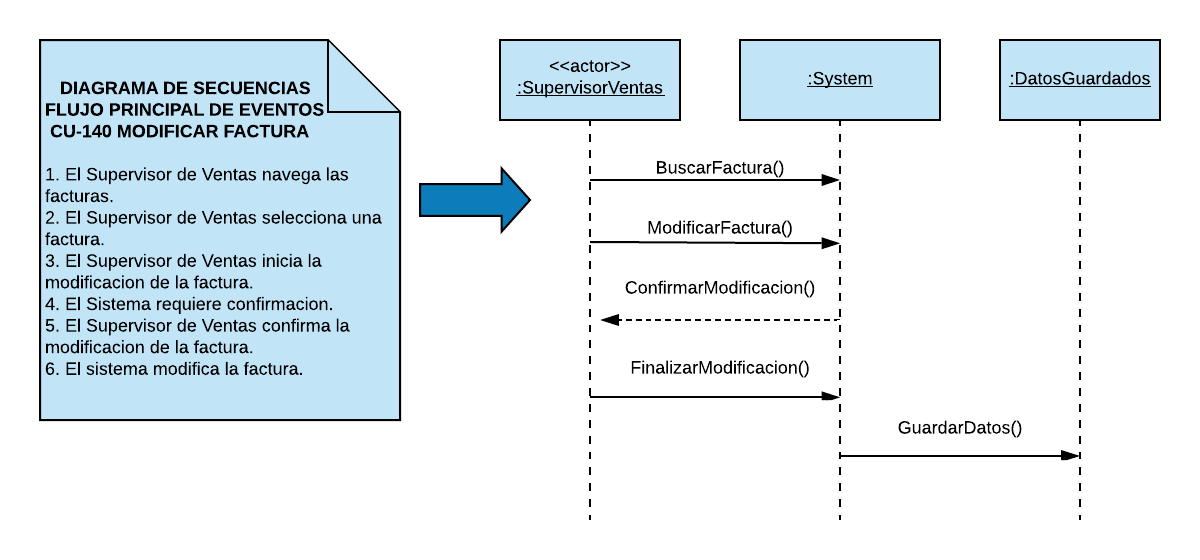
* + 1. **Flujo Alternativo de Eventos**

****

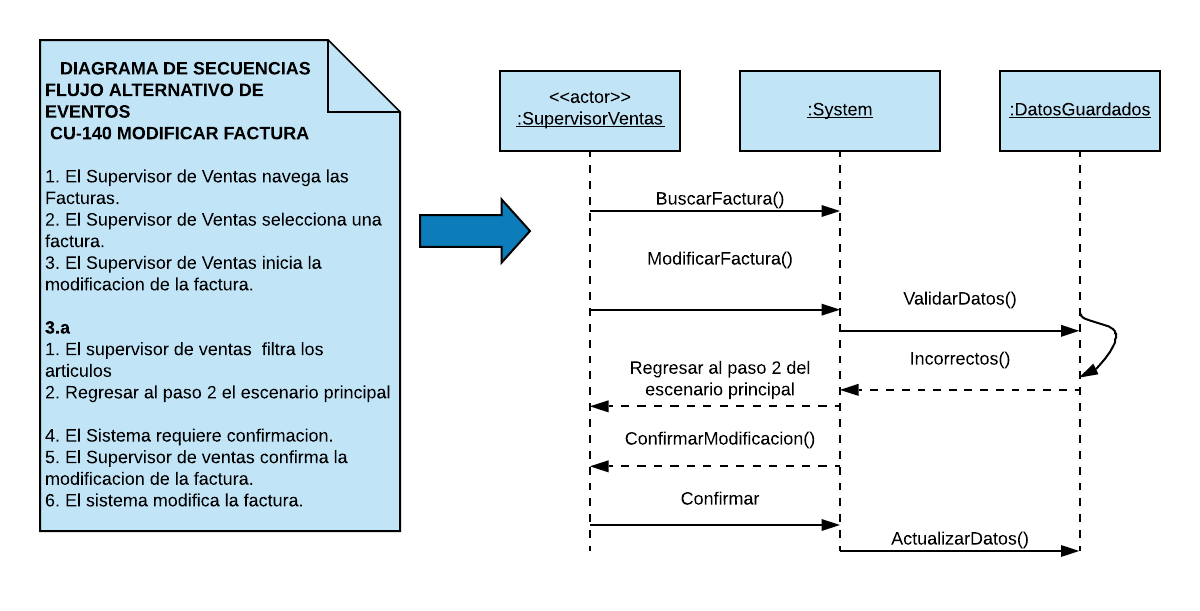
* + 1. **Flujo Excepcional de Eventos**

****

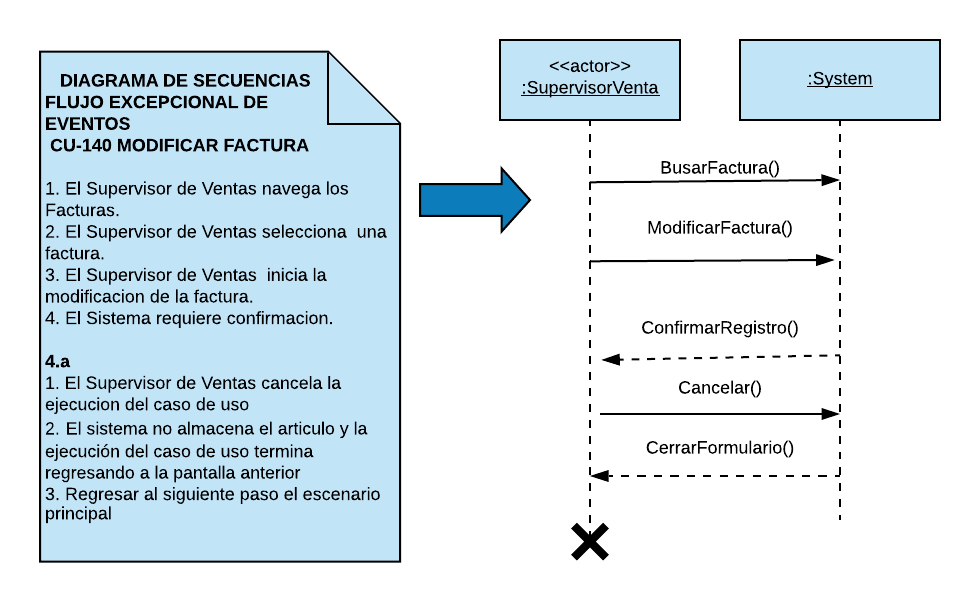
* 1. **CU-140 Modificar Factura**
     1. **Flujo Principal de Eventos**

****

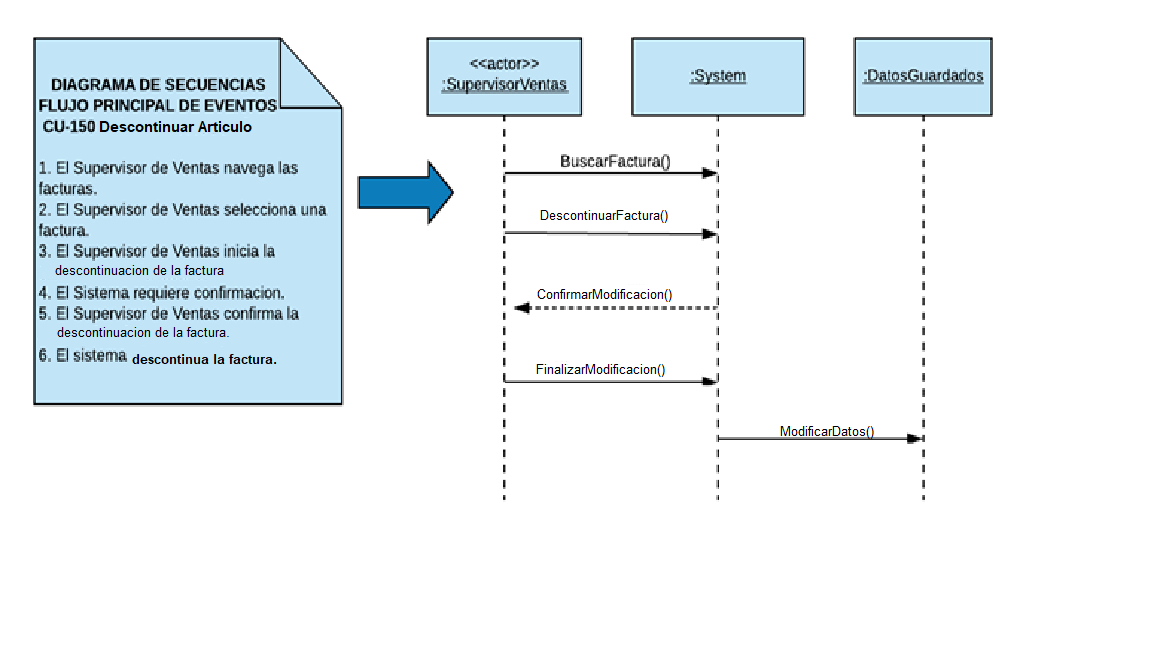
* + 1. **Flujo Alternativo de Eventos**

****

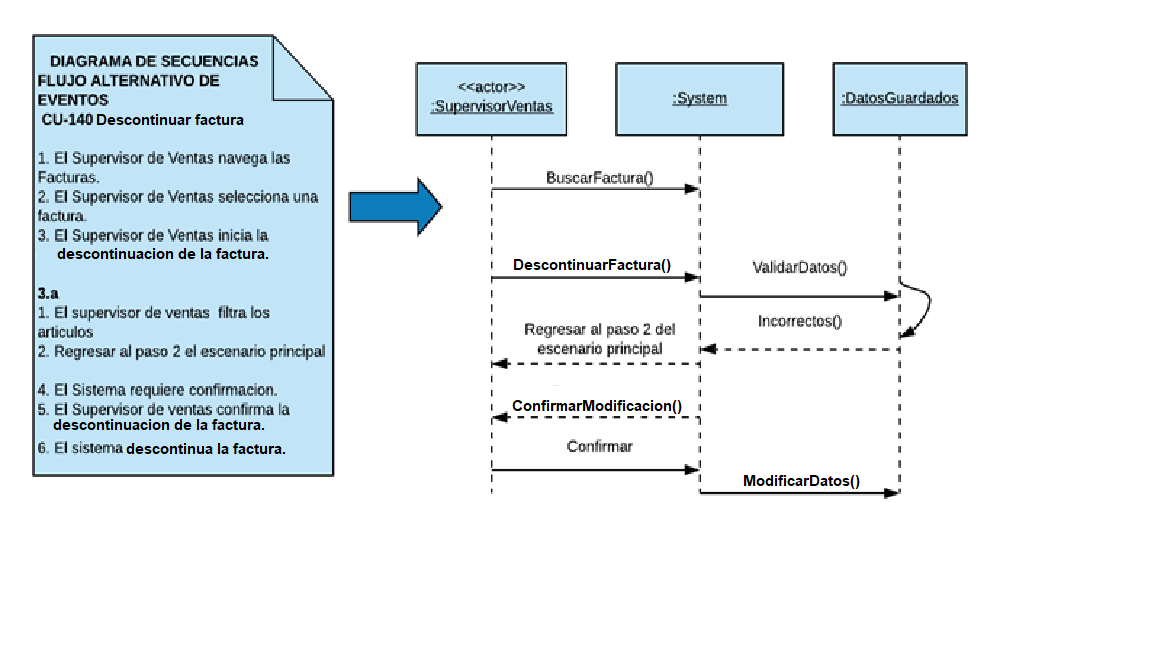
* + 1. **Flujo Excepcional de Eventos**

****

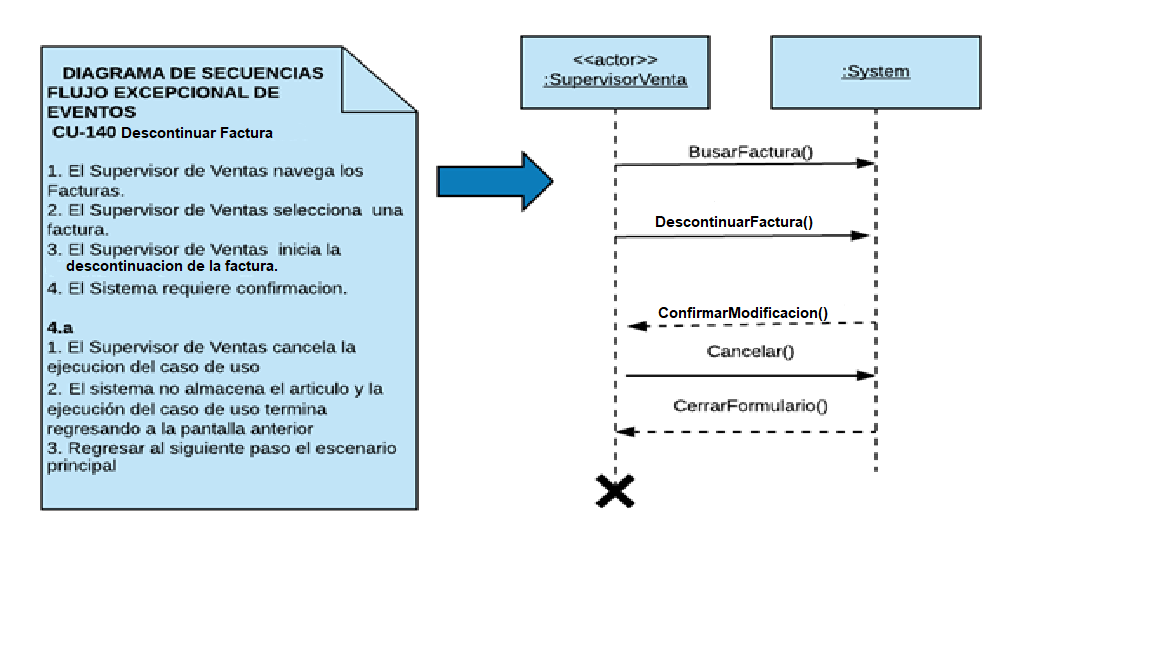
* 1. **CU-150 Descontinuar Factura**
     1. **Flujo Principal de Eventos**

****

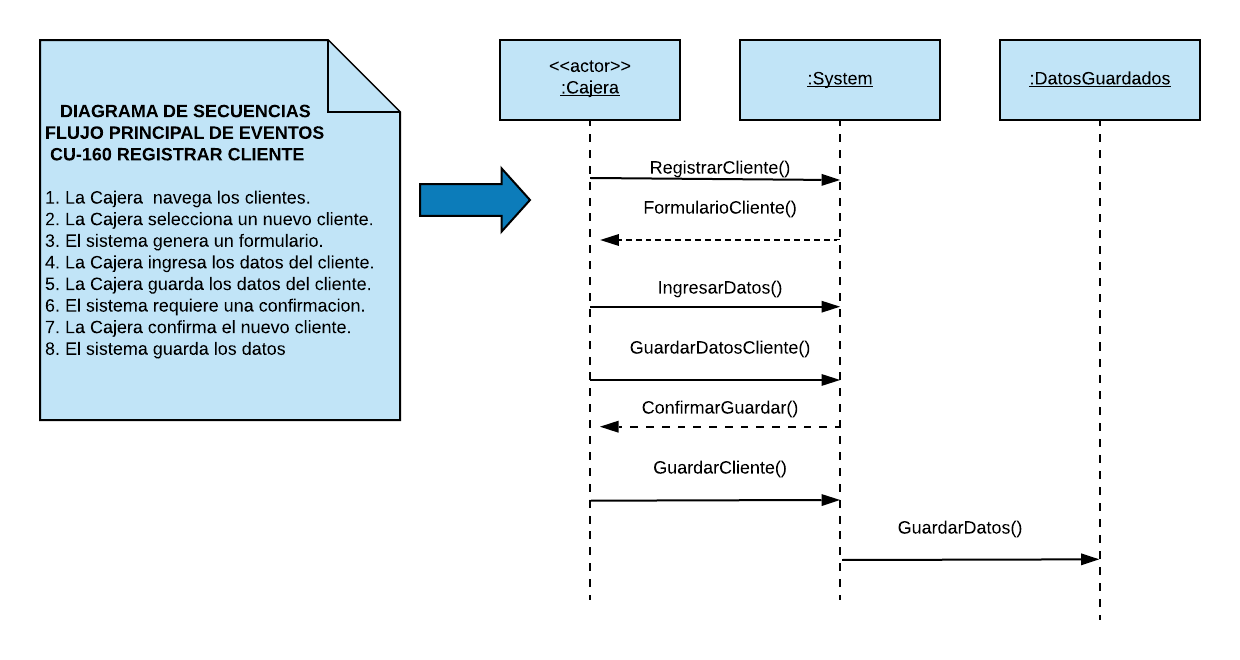
* + 1. **Flujo Alternativo de Eventos**

****

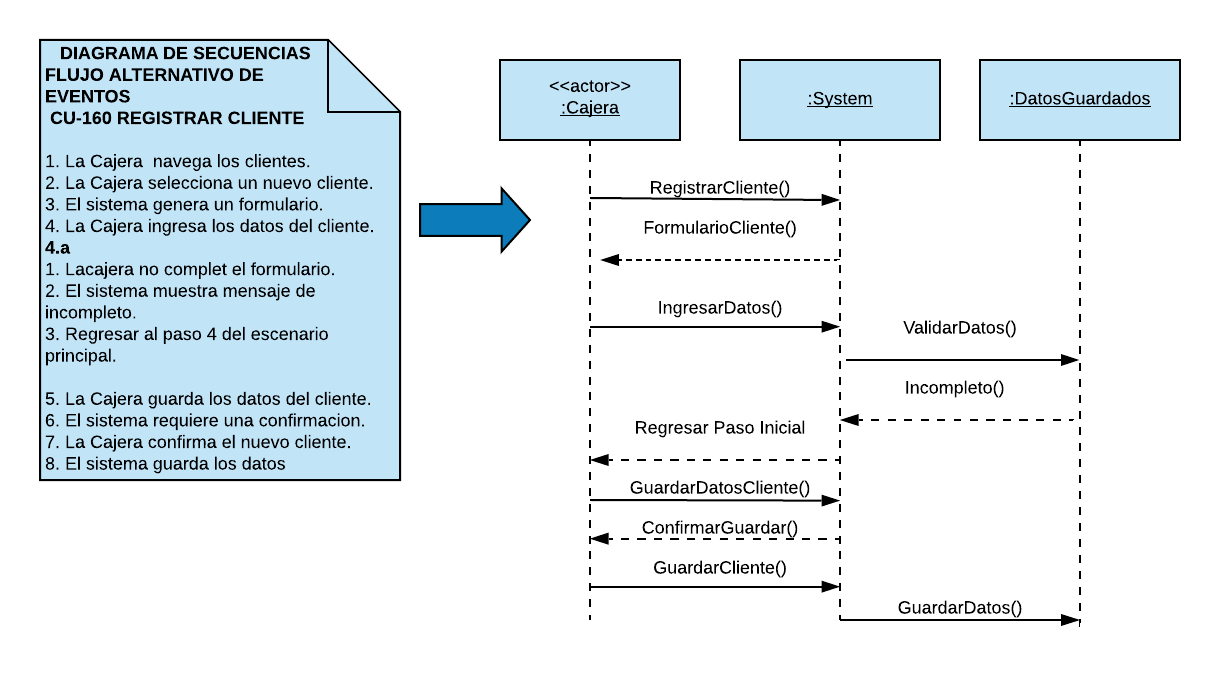
* + 1. **Flujo Excepcional de Eventos**

****

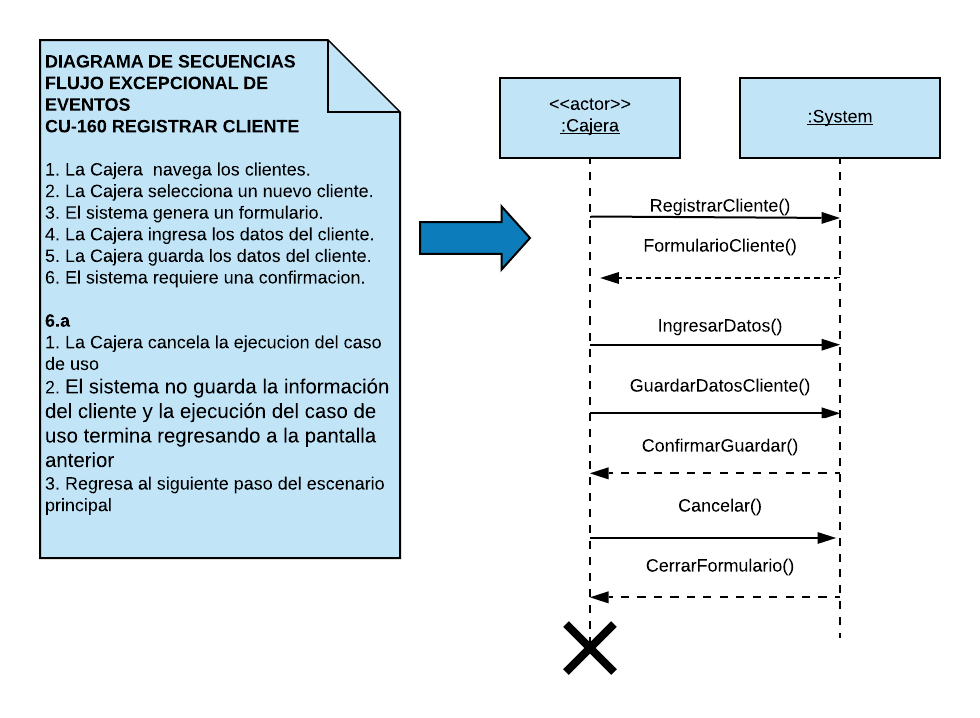
* 1. **CU-160 Registrar Cliente** 
     1. **Flujo Principal de Eventos**

****

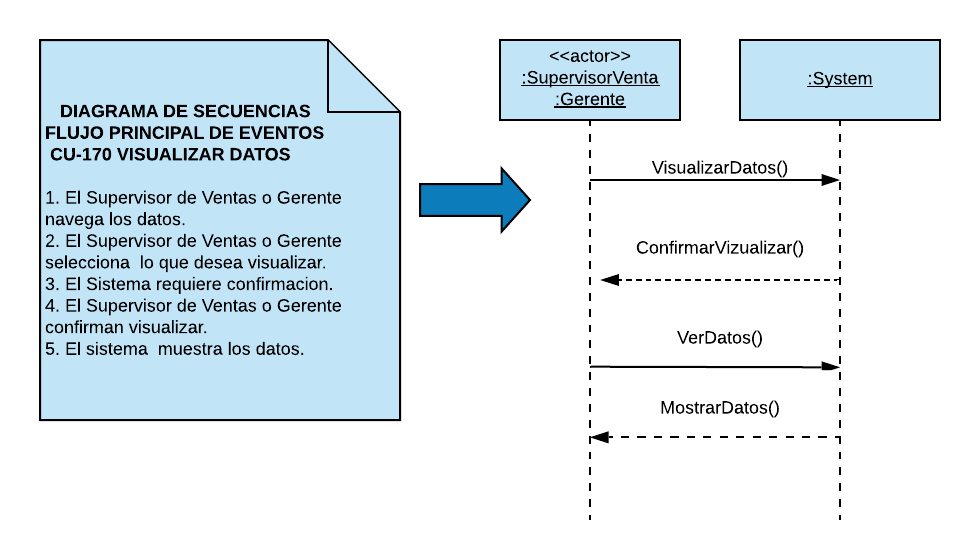
* + 1. **Flujo Alternativo de Eventos**

****

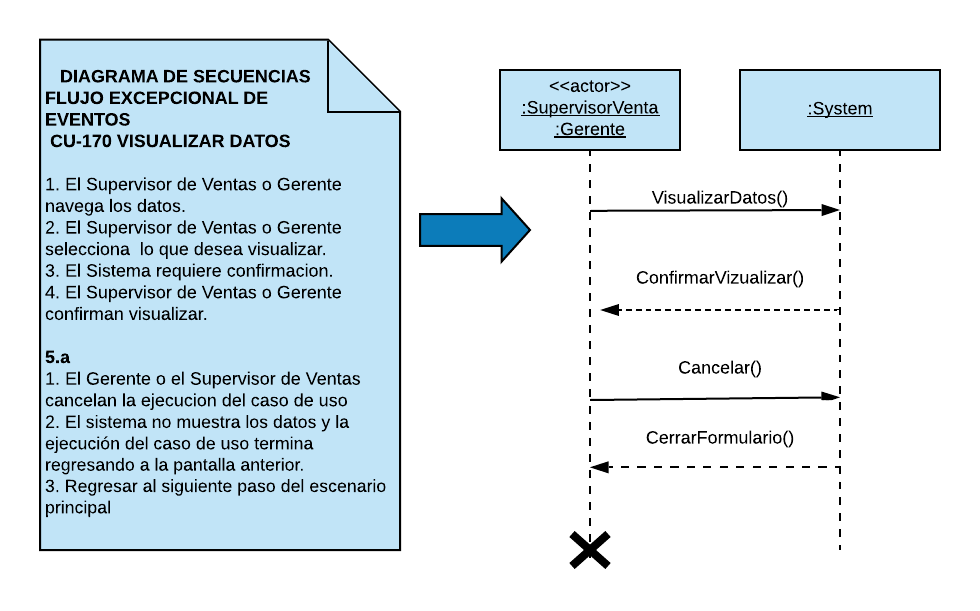
* + 1. **Flujo Excepcional de Eventos**

****

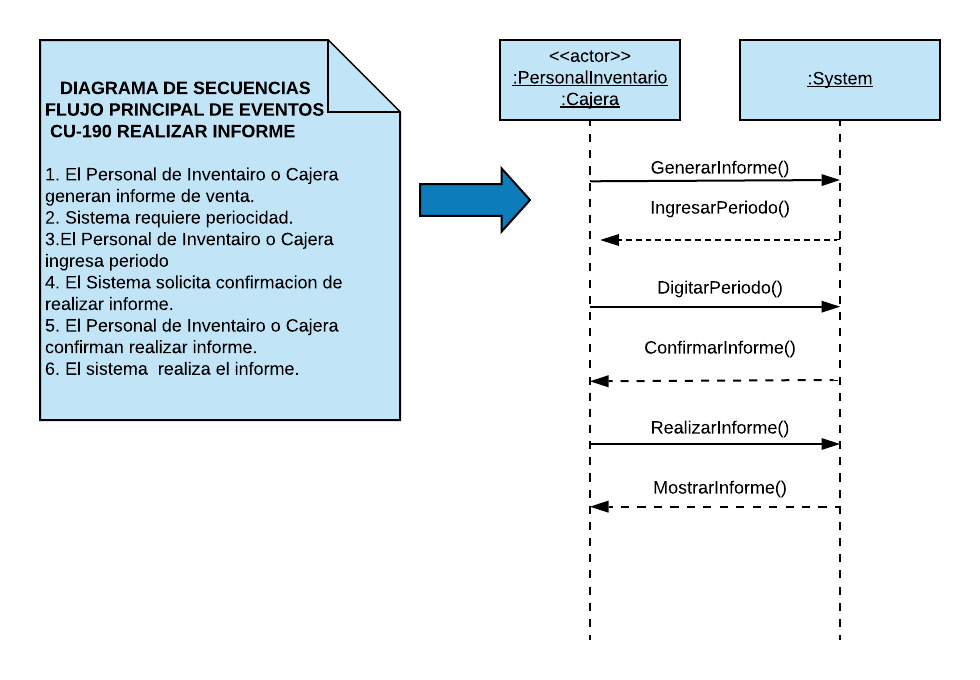
* 1. **CU-170 Visualizar Datos**
     1. **Flujo Principal de Eventos**

****

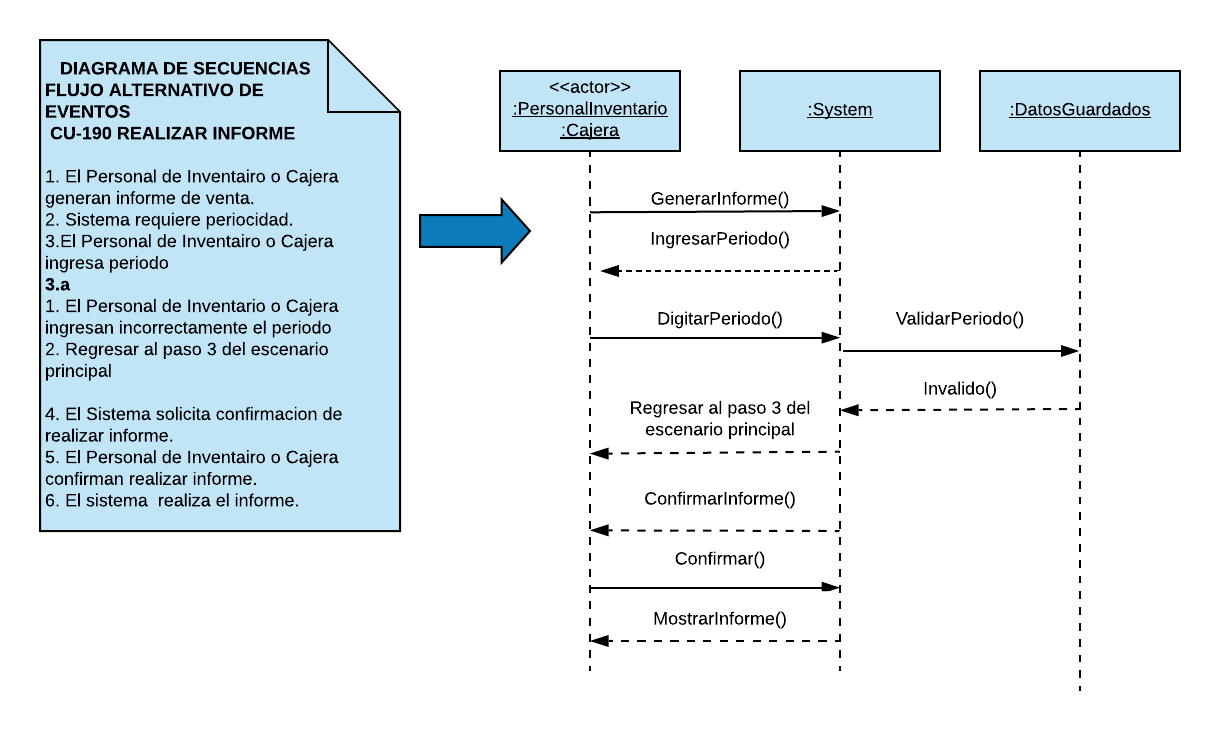
* + 1. **Flujo Excepcional de Eventos**

****

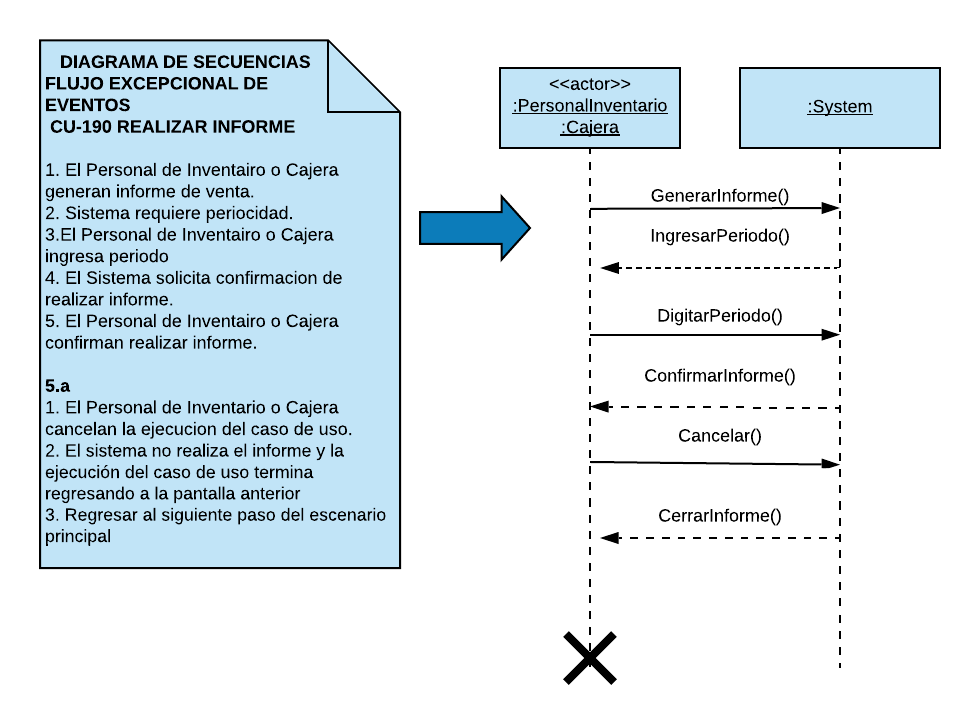
* 1. **CU-190 Realizar Informe**
     1. **Flujo Principal de Eventos**

****

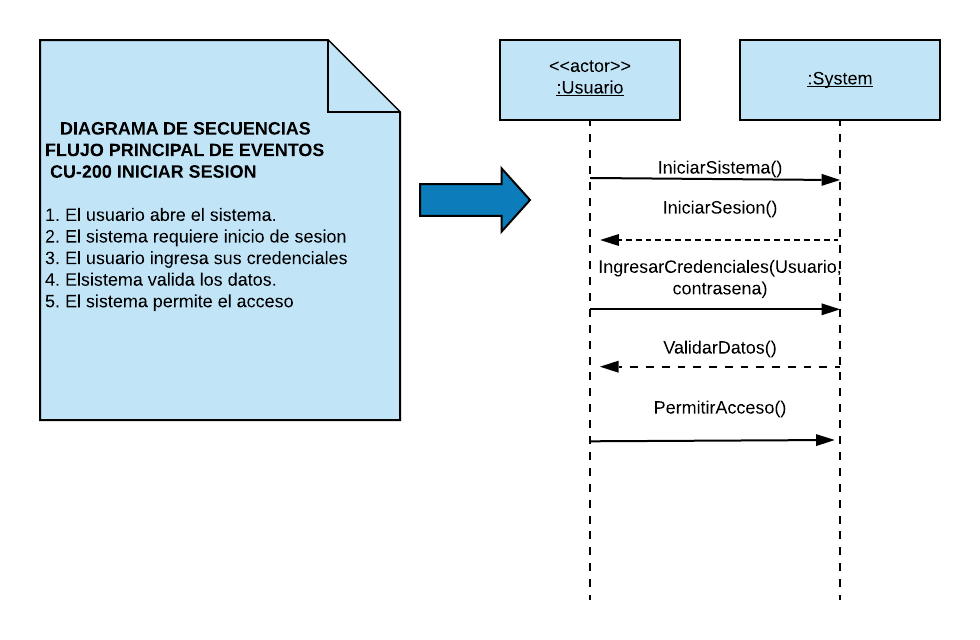
* + 1. **Flujo Alternativo de Eventos**

****

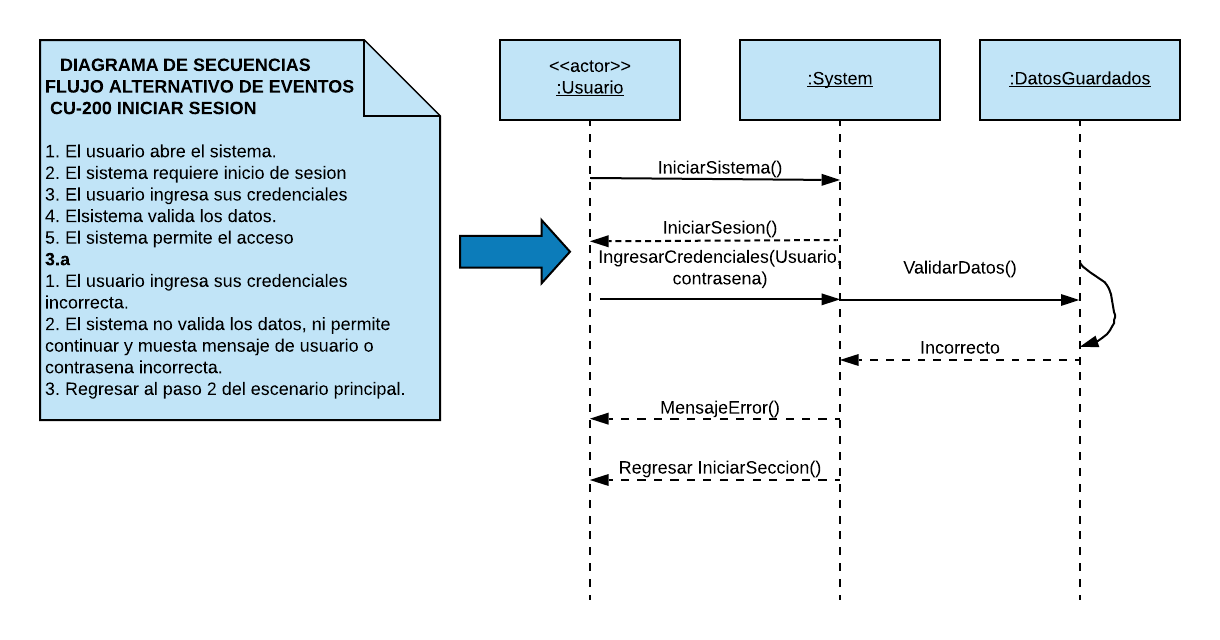
* + 1. **Flujo Excepcional de Eventos**

****

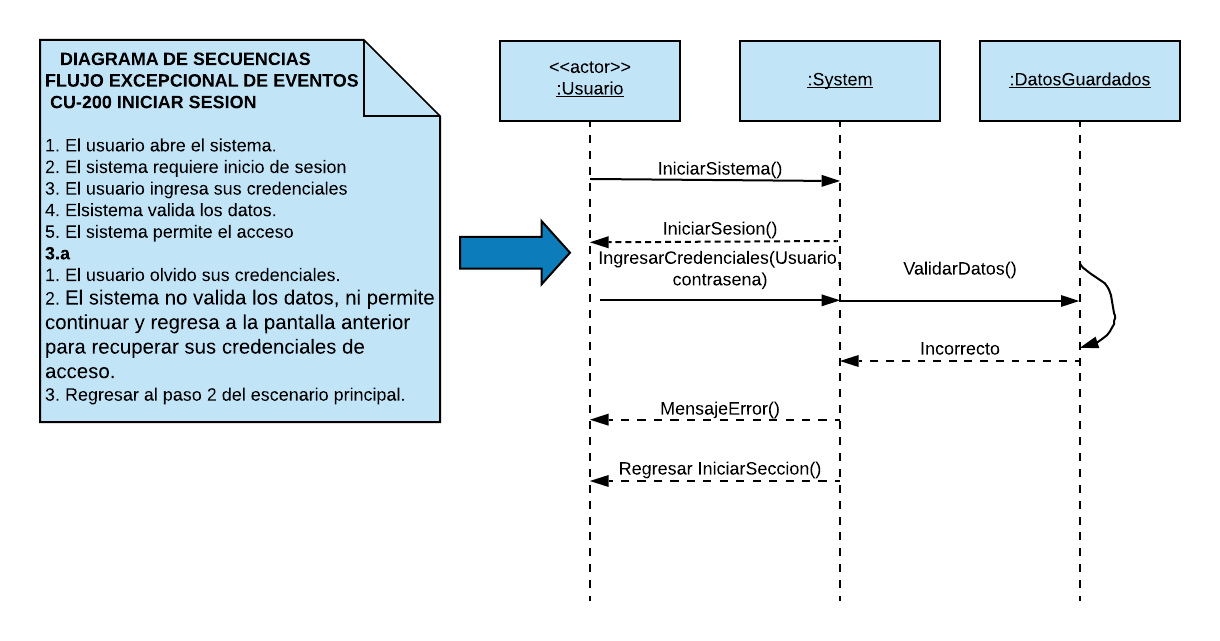
* 1. **CU-200 Iniciar Sesión**
     1. **Flujo Principal de Eventos**

****

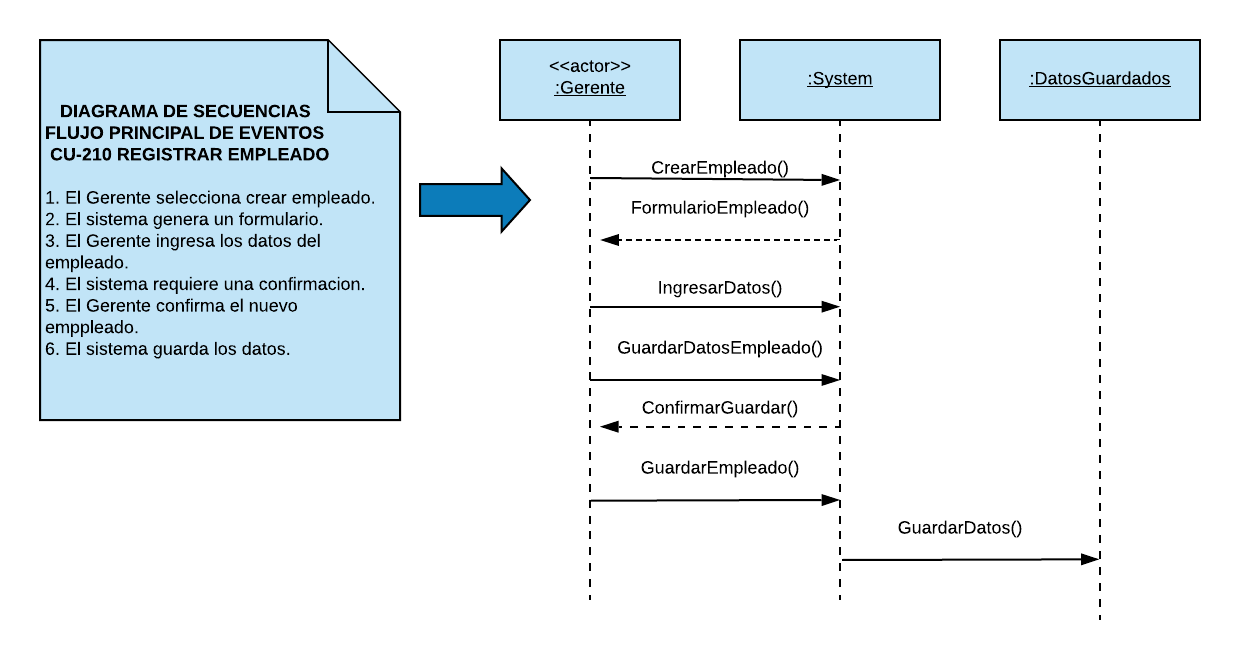
* + 1. **Flujo Alternativo de Eventos**

****

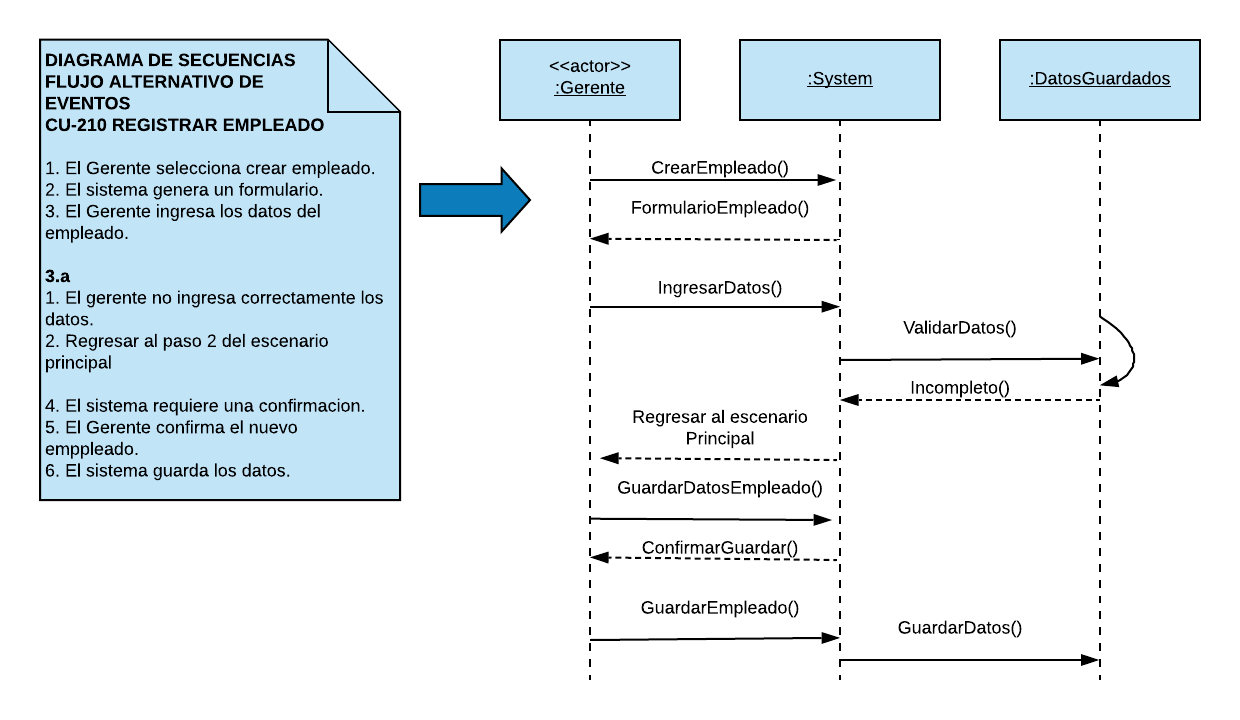
* + 1. **Flujo Excepcional de Eventos**

****

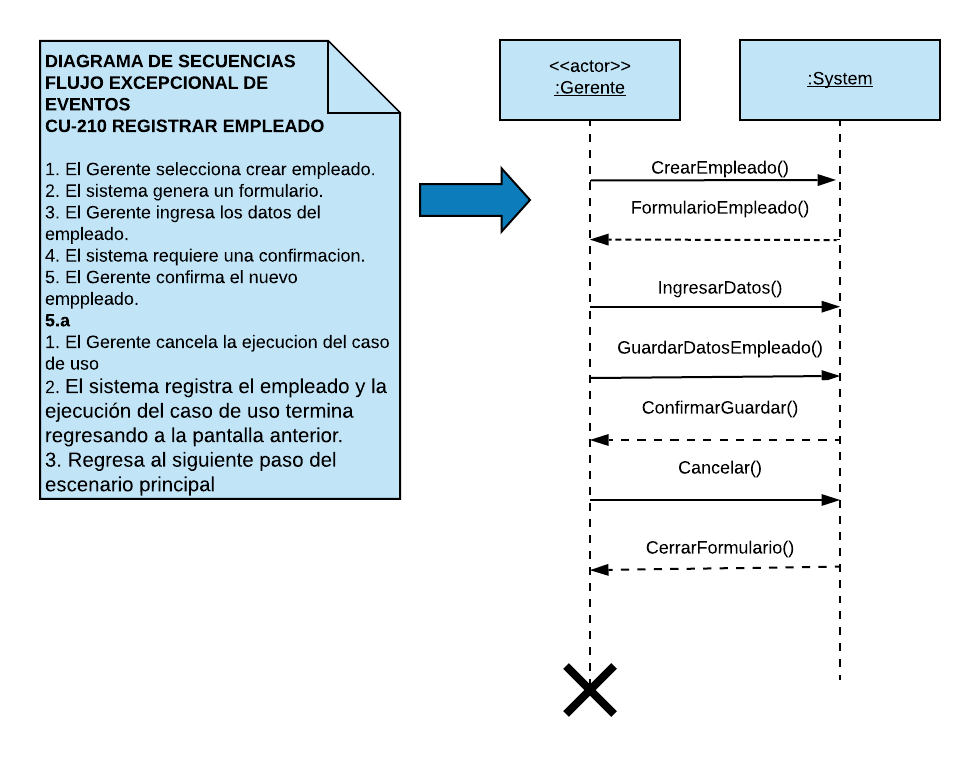
* 1. **CU-210 Registrar Empleados**
     1. **Flujo Principal de Eventos**

****

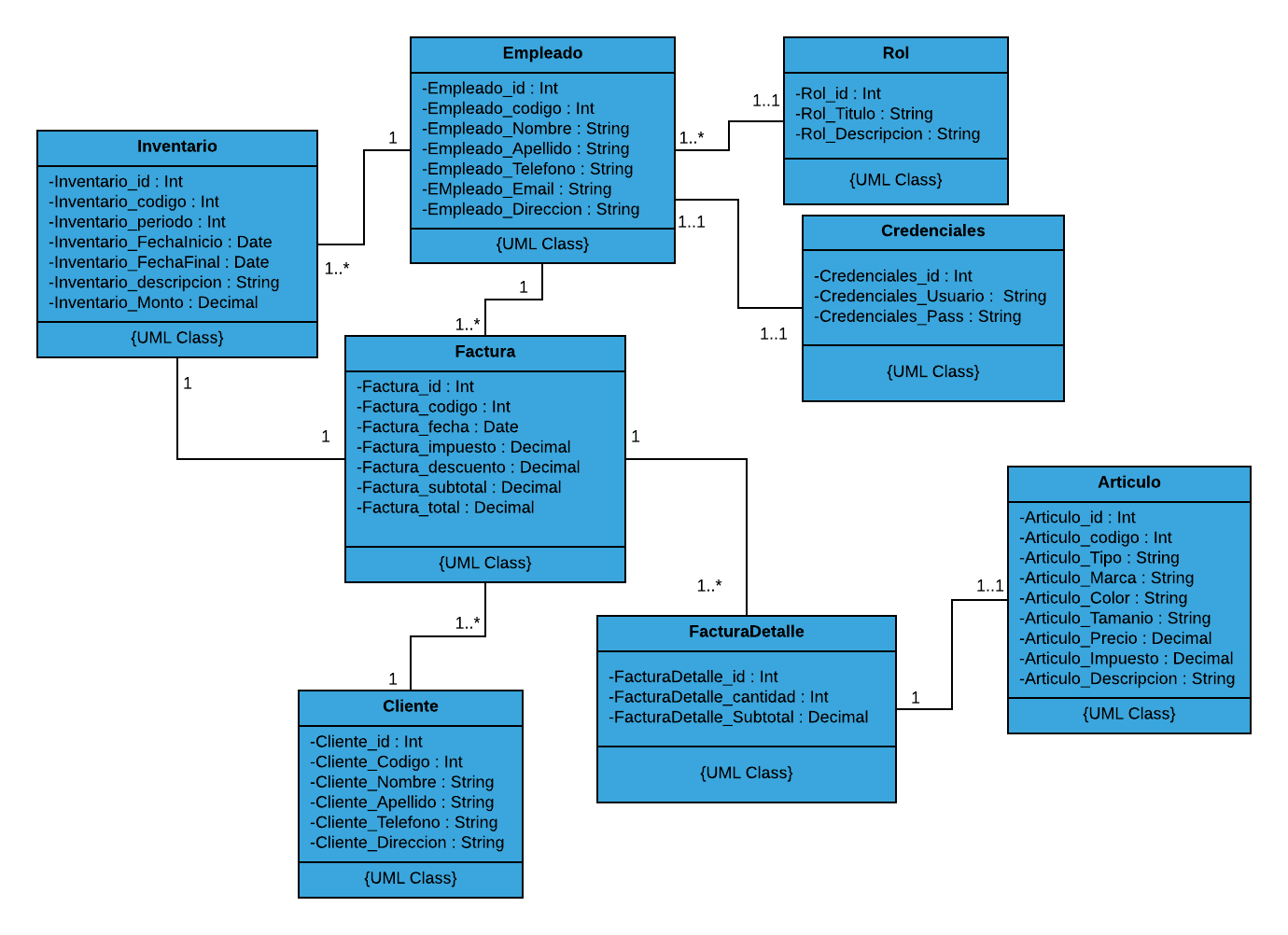
* + 1. **Flujo Alternativo de Eventos**

****

* + 1. **Flujo Excepcional de Eventos**

****

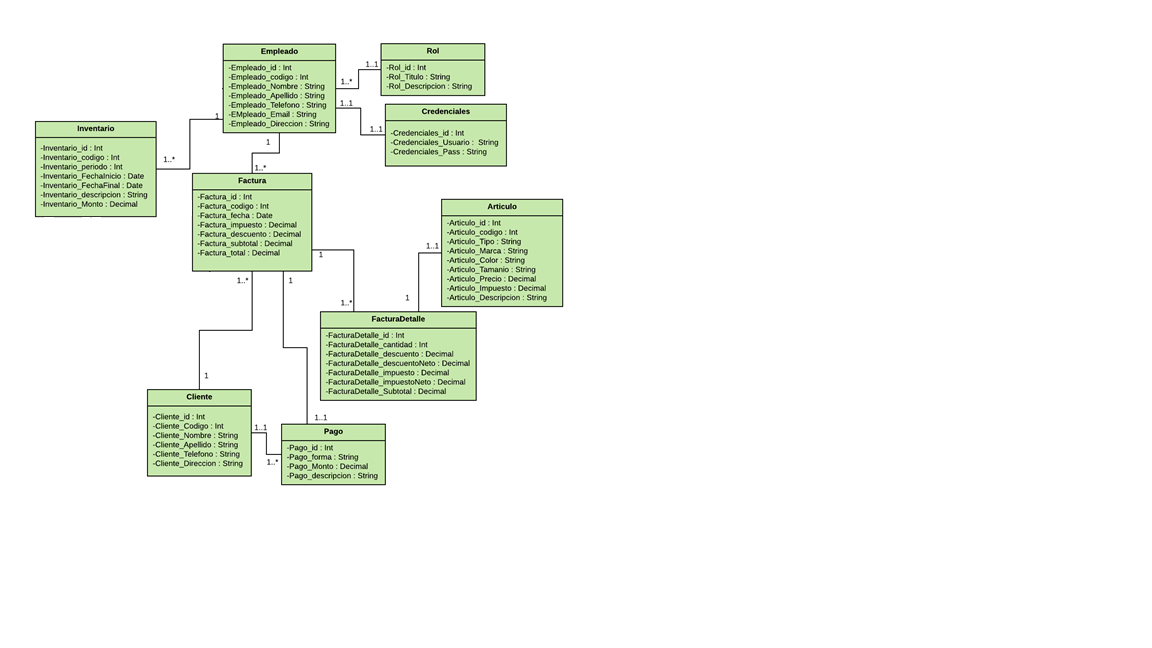
## Crear Diagrama de Clases a la Realización de Caso de Uso



# Diseño de la base de datos

## Crear el modelo de datos (Diagrama de Datos)

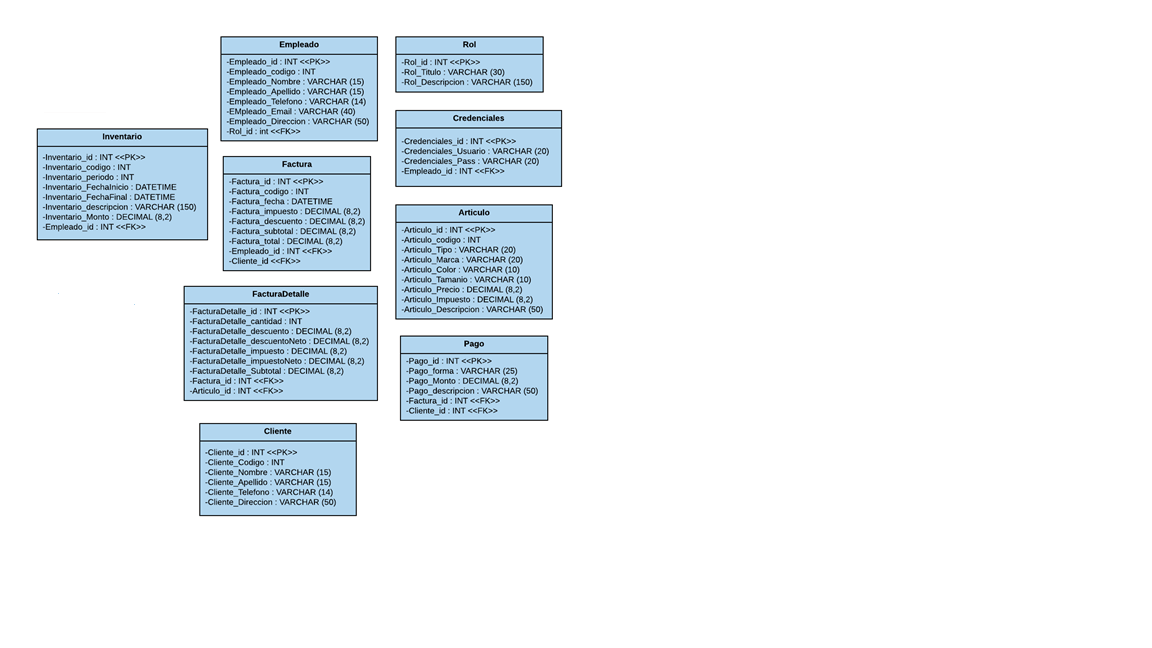
* 1. **Diagrama de Clases UML del Modelo Conceptual del Dominio**



* 1. **Diagrama de Clases UML del Modelo Lógico Puente**



* 1. **Modelos Relacionales de las Clases en el Modelo Lógico Puente**



## Crear el esquema de la base de datos (en un Script)

* 1. -- SQL SERVER

CREATE DATABASE FunPartyDB

* 1. **Escribir los scripts para:**
     1. **Crear las tablas**

-- CREACION DE LAS TABLAS

CREATE TABLE EMPLEADO(

EMPLEADO\_ID INT NOT NULL PRIMARY KEY IDENTITY(1,1),

EMPLEADO\_CODIGO INT NOT NULL,

ROL\_ID INT,

EMPLEADO\_NOMBRE VARCHAR (15) NOT NULL,

EMPLEADO\_APELLIDO VARCHAR (15) NOT NULL,

EMPLEADO\_TELEFONO VARCHAR (14) NOT NULL,

EMPLEADO\_EMAIL VARCHAR (40) NOT NULL,

EMPLEADO\_DIRECCION VARCHAR (50) NOT NULL,

FOREIGN KEY (ROL\_ID) REFERENCES ROL(ROL\_ID)

)

GO

CREATE TABLE ROL(

ROL\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

ROL\_TITULO VARCHAR (30) NOT NULL,

ROL\_DESCRIPCION VARCHAR (150) NOT NULL,

)

GO

CREATE TABLE CREDENCIALES (

CREDENCIALES\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1)

EMPLEADO\_ID INT NOT NULL,

CREDENCIALES\_USUARIO VARCHAR (20) NOT NULL,

CREDENCIALES\_PASS VARCHAR (20) NOT NULL,

FOREIGN KEY (EMPLEADO\_ID) REFERENCES EMPLEADO(EMPLEADO\_ID)

)

GO

CREATE TABLE CLIENTE (

CLIENTE\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

CLIENTE\_CODIGO INT NOT NULL,

CLIENTE\_NOMBRE VARCHAR (15) NOT NULL,

CLIENTE\_APELLIDO VARCHAR (15) NOT NULL,

CLIENTE\_TELEFONO VARCHAR (14) NOT NULL,

CLIENTE\_DIRECCION VARCHAR (50) NOT NULL,

)

GO

CREATE TABLE INVENTARIO (

EMPLEADO\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

INVENTARIO\_ID INT NOT NULL,

INVENTARIO\_CODIGO INT NOT NULL,

INVENTARIO\_PERIODO INT NOT NULL,

INVENTARIO\_FECHAINICIAL DATETIME,

INVENTARIO\_FECHAFINAL DATETIME,

INVENTARIO\_DESCRIPCION VARCHAR (150) NOT NULL,

INVENTARIO\_MONTO DECIMAL (8,2) DEFAULT 0,

FOREIGN KEY (EMPLEADO\_ID) REFERENCES EMPLEADO(EMPLEADO\_ID)

)

GO

CREATE TABLE ALQUILER (

ALQUILER\_ID INT NOT NULL PRIMARY KEY IDENTITY(1,1),

INVENTARIO\_ID INT NOT NULL,

FACTURA\_ID INT NOT NULL,

FOREIGN KEY (INVENTARIO\_ID) REFERENCES INVENTARIO(INVENTARIO\_ID),

FOREIGN KEY (FACTURA\_ID) REFERENCES FACTURA(FACTURA\_ID)

)

GO

CREATE TABLE VENTA(

VENTA\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

INVENTARIO\_ID INT NOT NULL,

FACTURA\_ID INT NOT NULL,

FOREIGN KEY (INVENTARIO\_ID) REFERENCES INVENTARIO(INVENTARIO\_ID),

FOREIGN KEY (FACTURA\_ID) REFERENCES FACTURA(FACTURA\_ID)

)

GO

CREATE TABLE FACTURA(

FACTURA\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

FACTURA\_CODIGO INT NOT NULL,

EMPLEADO\_ID INT NOT NULL,

CLIENTE\_ID INT,

FACTURA\_FECHA DATETIME DEFAULT CURRENT\_DATE,

FACTURA\_IMPUESTOS DECIMAL (8,2) DEFAULT 0,

FACTURA\_DESCUENTO DECIMAL (8,2) DEFAULT 0,

FACTURA\_SUBTOTAL DECIMAL (8,2) DEFAULT 0,

FACTURA\_TOTAL DECIMAL (8,2) DEFAULT 0,

FOREIGN KEY (EMPLEADO\_ID) REFERENCES EMPLEADO(EMPLEADO\_ID),

FOREIGN KEY (CLIENTE\_ID) REFERENCES CLIENTE(CLIENTE\_ID)

)

GO

CREATE TABLE FACTURADETALLE(

FACTURADETALLE\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

FACTURA\_ID INT NOT NULL,

ARTICULO\_ID INT NOT NULL,

FACTURADETALLE\_CANTIDAD INT NOT NULL DEFAULT 1,

FACTURADETALLE\_DESCUENTO DECIMAL (8,2) DEFAULT 0,

FACTURADETALLE\_DESCUENTONETO DECIMAL (8,2) DEFAULT 0,

FACTURADETALLE\_IMPUESTO DECIMAL (8,2) DEFAULT 0,

FACTURADETALLE\_IMPUESTONETO DECIMAL (8,2) DEFAULT 0,

FACTURADETALLE\_SUBTOTAL DECIMAL (8,2) DEFAULT 0,

FOREIGN KEY (FACTURA\_ID) REFERENCES FACTURA(FACTURA\_ID),

FOREIGN KEY (ARTICULO\_ID) REFERENCES ARTICULO(ARTICULO\_ID)

)

GO

CREATE TABLE ARTICULO(

ARTICULO\_ID INT NOT NULL PRIMARY KEY IDENTITY (1,1),

ARTICULO\_CODIGO INT NOT NULL,

ARTICULO\_TIPO VARCHAR (20) NOT NULL,

ARTICULO\_MARCA VARCHAR (20) NOT NULL,

ARTICULO\_COLOR VARCHAR (10) NOT NULL,

ARTICULO\_TAMANIO VARCHAR (10) NOT NULL,

ARTICULO\_PRECIO DECIMAL (4,2) NOT NULL

ARTICULO\_IMPUESTO DECIMAL (8,2) NOT NULL,

ARTICULO\_DESCRIPCION VARCHAR (50) NOT NULL

)

GO

CREATE TABLE PAGO(

PAGO\_ID INT NOT NULL PRIMARY KEY IDENTITY(1,1),

FACTURA\_ID INT NOT NULL,

CLIENTE\_ID INT NOT NULL,

PAGO\_FORMA VARCHAR (25) NOT NULL,

PAGO\_MONTO DECIMAL (8,2) NOT NULL,

PAGO\_DESCRIPCION VARCHAR (50) NOT NULL,

FOREIGN KEY (FACTURA\_ID) REFERENCES FACTURA(FACTURA\_ID)

FOREIGN KEY (CLIENTE\_ID) REFERENCES CLIENTE(CLIENTE\_ID)

)

GO

* + 1. **Poblar las tablas con 10 tuplas**

--- TABLA EMPLEADO

INSERT INTO EMPLEADO (EMPLEADO\_ID, EMPLEADO\_CODIGO, ROL\_ID, NOMBRE, APELLIDO, TELEFONO, EMAIL, DIRECCION)

VALUES (1, 1010, 1, 'LISBETH', 'DIAZ', '849-656-1208', 'L.DIAZ@HOTMAIL.COM', 'CALLE SAN ANTONIO 99');

INSERT INTO EMPLEADO (EMPLEADO\_ID, EMPLEADO\_CODIGO, ROL\_ID, NOMBRE, APELLIDO, TELEFONO, EMAIL, DIRECCION)

VALUES (2, 2020, 2, 'SCARLET', 'DIAZ', '809-245-1416', 'S.DIAZ@HOTMAIL.COM', 'CALLE SAN ANTONIO 99');

INSERT INTO EMPLEADO (EMPLEADO\_ID, EMPLEADO\_CODIGO, ROL\_ID, NOMBRE, APELLIDO, TELEFONO, EMAIL, DIRECCION)

VALUES (3, 3030, 2, 'ESTARLYN', 'DIAZ', '809-245-1416', 'E.DIAZ@HOTMAIL.COM', 'CALLE SAN ANTONIO 99');

INSERT INTO EMPLEADO (EMPLEADO\_ID, EMPLEADO\_CODIGO, ROL\_ID, NOMBRE, APELLIDO, TELEFONO, EMAIL, DIRECCION)

VALUES (4, 4040, 2, 'MANUEL', 'DIAZ', '829-967-0803', 'M.DIAZ@HOTMAIL.COM', 'CALLE SAN ANTONIO 99');

INSERT INTO EMPLEADO (EMPLEADO\_ID, EMPLEADO\_CODIGO, ROL\_ID, NOMBRE, APELLIDO, TELEFONO, EMAIL, DIRECCION)

VALUES (5, 5050, 3, 'JUANA', 'REINOSO', '829-967-0803', 'J.REINOSO@HOTMAIL.COM', 'CALLE SAN ANTONIO 99');

--- TABLA ROL

INSERT INTO ROL (ROL\_ID, TITULO, DESCRIPCION)

VALUES (1, 'GERENTE', 'SE ENCARGA DE REGISTRAR LOS EMPLEADOS, GENERAR LA NOMINA Y VER LA INFORMACION REGISTRADA EN EL SISTEMA.' );

INSERT INTO ROL (ROL\_ID, TITULO, DESCRIPCION)

VALUES (2, 'SUPERVISOR DE VENTAS', 'SE ENCARGA DE MODIFICAR Y DESCONTINUAR LAS FACTURA Y VER LA INFORMACION DEL SISTEMA.' );

INSERT INTO ROL (ROL\_ID, TITULO, DESCRIPCION)

VALUES (3, 'PERSONAL DE INVENTARIO', 'SE ENCARGA DE INGRESAR, MODIFICAR Y DESCONTINUAR LAS FACTURAS DEL SISTEMA Y REALIZAR EL INVENTARIO.' );

INSERT INTO ROL (ROL\_ID, TITULO, DESCRIPCION)

VALUES (4, 'CAJERA' , 'SE ENCARGA DE CREAR LAS FACTURAS, REGISTRAR LOS CLIENTES Y REALIZAR EL INVENTARIO.' );

--- TABLA CREDENCIALES

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

VALUES (1, 'L.DIAZ', 'DIAZ1234' );

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

VALUES (2, 'S.DIAZ', 'DIAZ2345' );

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

VALUES (3, 'E.DIAZ', 'DIAZ3456');

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

VALUES (4, 'M.DIAZ', 'DIZ4567');

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

VALUES (5, 'J.REINOSO', 'REINOSO1234' );

INSERT INTO CREDENCIALES (EMPLEADO\_ID, USUARIO, CLAVE)

--- TABLA CLIENTE

INSERT INTO CLIENTE (CLIENTE\_ID, CLIENTE\_CODIGO, NOMBRE, APELLIDO, TELEFONO, DIRECCION)

VALUES (1, 100, 'JULIAN', 'PEREZ', '809-123-4567', 'CALLE SOL 83');

INSERT INTO CLIENTE (CLIENTE\_ID, CLIENTE\_CODIGO, NOMBRE, APELLIDO, TELEFONO, DIRECCION)

VALUES (2, 200, 'PABLO', 'PAULINO', '809-123-4567', 'CALLE SOL 84');

INSERT INTO CLIENTE (CLIENTE\_ID, CLIENTE\_CODIGO, NOMBRE, APELLIDO, TELEFONO, DIRECCION)

VALUES (3, 300, 'ALAN', 'PINEDA', '809-123-4567', 'CALLE SOL 85');

INSERT INTO CLIENTE (CLIENTE\_ID, CLIENTE\_CODIGO, NOMBRE, APELLIDO, TELEFONO, DIRECCION)

VALUES (4, 400, 'JULIAN', 'PEREZ', '809-123-4567', 'CALLE SOL 86');

INSERT INTO CLIENTE (CLIENTE\_ID, CLIENTE\_CODIGO, NOMBRE, APELLIDO, TELEFONO, DIRECCION)

VALUES (5, 500, 'ANA', 'MARTINEZ', '809-123-4567', 'CALLE SOL 87');

--- TABLA INVENTARIO

INSERT INTO INVENTARIO (EMPLEADO\_ID, INVENTARIO\_ID, INVENTARIO\_CODIGO, PERIODO, FECHAINICIAL, FECHAFINAL, DESCRIPCION, MONTO)

VALUES (1, 7, 101010, 'DIA', '27-OCT-2019', '28-OCT-2019', 'INVENTARIO DEL DIA', 1000.00);

INSERT INTO INVENTARIO (EMPLEADO\_ID, INVENTARIO\_ID, INVENTARIO\_CODIGO, PERIODO, FECHAINICIAL, FECHAFINAL, DESCRIPCION, MONTO)

VALUES (2, 8, 202020, 'SEMANA', '27-OCT-2019', '02-NOV-2019', 'INVENTARIO SEMANAL', 2000.00);

INSERT INTO INVENTARIO (EMPLEADO\_ID, INVENTARIO\_ID, INVENTARIO\_CODIGO, PERIODO, FECHAINICIAL, FECHAFINAL, DESCRIPCION, MONTO)

VALUES (3, 9, 303030, 'MES', '01-OCT-2019', '31-OCT-2019', 'INVENTARIO MENSUAL', 3000.00);

INSERT INTO INVENTARIO (EMPLEADO\_ID, INVENTARIO\_ID, INVENTARIO\_CODIGO, PERIODO, FECHAINICIAL, FECHAFINAL, DESCRIPCION, MONTO)

VALUES (4, 10, 404040, 'ANIO', '01-ENE-2019', '31-DIC-2019', 'INVENTARIO ANUAL', 4000.00);

INSERT INTO INVENTARIO (EMPLEADO\_ID, INVENTARIO\_ID, INVENTARIO\_CODIGO, PERIODO, FECHAINICIAL, FECHAFINAL, DESCRIPCION, MONTO)

VALUES (5, 7, 505050, 'DIA', '15-NOV-2019', '16-NOV-2019', 'INVENTARIO DEL DIA', 5000.00);

--- TABLA FACTURA

INSERT INTO FACTURA (FACTURA\_ID, FACTURA\_CODIGO, EMPLEADO\_ID, CLIENTE\_ID, FECHA, IMPUESTO, DESCUENTO, SUBTOTAL, TOTAL)

VALUES (1, 010101, 1, 1, 1, DEFAULT, 100.00, 0.00, 1000.00, 1100.00);

INSERT INTO FACTURA (FACTURA\_ID, FACTURA\_CODIGO, EMPLEADO\_ID, CLIENTE\_ID, FECHA, IMPUESTO, DESCUENTO, SUBTOTAL, TOTAL)

VALUES (2, 020202, 2, 2, 2, DEFAULT, 200.00, 0.00, 2000.00, 2200.00);

INSERT INTO FACTURA (FACTURA\_ID, FACTURA\_CODIGO, EMPLEADO\_ID, CLIENTE\_ID, FECHA, IMPUESTO, DESCUENTO, SUBTOTAL, TOTAL)

VALUES (3, 030303, 3, 3, 3, DEFAULT, 300.00, 0.00, 3000.00, 3300.00);

INSERT INTO FACTURA (FACTURA\_ID, FACTURA\_CODIGO, EMPLEADO\_ID, CLIENTE\_ID, FECHA, IMPUESTO, DESCUENTO, SUBTOTAL, TOTAL)

VALUES (4, 040404, 4, 4, 4, DEFAULT, 400.00, 0.00, 4000.00, 4400.00);

INSERT INTO FACTURA (FACTURA\_ID, FACTURA\_CODIGO, EMPLEADO\_ID, CLIENTE\_ID, FECHA, IMPUESTO, DESCUENTO, SUBTOTAL, TOTAL)

VALUES (5, 050505, 5, 5, 5, DEFAULT, 500.00, 0.00, 5000.00, 5500.00 );

--- TABLA FACTURADETALLE

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID, FACTURA\_ID, ARTICULO\_ID, CANTIDAD, DESCUENTO, DESCUENTONETO IMPUESTO, IMPUESTONETO, SUBTOTAL)

VALUES (1, 1, 1, 1, 0.0, 0.0, 50.00, 100.00, 1000.00, 1100.00 );

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID, FACTURA\_ID, ARTICULO\_ID, CANTIDAD, DESCUENTO, DESCUENTONETO IMPUESTO, IMPUESTONETO, SUBTOTAL)

VALUES (2, 2, 2, 2, 0.0, 0.0, 100.00, 200.00, 2000.00, 2200.00 );

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID, FACTURA\_ID, ARTICULO\_ID, CANTIDAD, DESCUENTO, DESCUENTONETO IMPUESTO, IMPUESTONETO, SUBTOTAL)

VALUES (3, 3, 3, 3, 0.0, 0.0, 100.00, 300.00, 3000.00, 3300.00 );

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID, FACTURA\_ID, ARTICULO\_ID, CANTIDAD, DESCUENTO, DESCUENTONETO IMPUESTO, IMPUESTONETO, SUBTOTAL)

VALUES (4, 4, 4, 4, 0.0, 0.0, 100.00, 400.00, 4000.00, 4400.00 );

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID, FACTURA\_ID, ARTICULO\_ID, CANTIDAD, DESCUENTO, DESCUENTONETO IMPUESTO, IMPUESTONETO, SUBTOTAL)

VALUES (5, 5, 5, 5, 0.0, 0.0, 100.00, 500.00, 5000.00, 5500.00 );

INSERT INTO FACTURADETALLE (FACTURADETALLE\_ID,

--- TABLA ARTICULO

INSERT INTO ARTICULO (ARTICULO\_ID, ARTICULO\_CODIGO, NOMBRE, TIPO, MARCA, COLOR, TAMANIO, PRECIO, DESCUENTO, DESCRIPCION)

VALUES (1, 101, 'GLOBOS', 'ALUMINIO', 'TEXTIL', 'AZUL', '4', 1000, 0.05, 'AL POR MAYOR' );

INSERT INTO ARTICULO (ARTICULO\_ID, ARTICULO\_CODIGO, NOMBRE, TIPO, MARCA, COLOR, TAMANIO, PRECIO, DESCUENTO, DESCRIPCION)

VALUES (2, 102, 'GLOBOS', 'ALUMINIO', 'TEXTIL', 'ROSA', '4', 1000, 0.05, 'AL POR MAYOR' );

INSERT INTO ARTICULO (ARTICULO\_ID, ARTICULO\_CODIGO, NOMBRE, TIPO, MARCA, COLOR, TAMANIO, PRECIO, DESCUENTO, DESCRIPCION)

VALUES (3, 103, 'VASOS', 'ALUMINIO', 'PLASTICOL', 'AZUL', '4', 1000, 0.05, 'AL POR MAYOR' );

INSERT INTO ARTICULO (ARTICULO\_ID, ARTICULO\_CODIGO, NOMBRE, TIPO, MARCA, COLOR, TAMANIO, PRECIO, DESCUENTO, DESCRIPCION)

VALUES (4, 104, 'PLATOS', 'ALUMINIO', 'PLASTICOL', 'AZUL', '4', 1000, 0.05, 'AL POR MAYOR' );

INSERT INTO ARTICULO (ARTICULO\_ID, ARTICULO\_CODIGO, NOMBRE, TIPO, MARCA, COLOR, TAMANIO, PRECIO, DESCUENTO, DESCRIPCION)

VALUES (5, 105, 'CUCHARAS', 'ALUMINIO', 'PLASTICOL', 'AZUL', '4', 1000, 0.05, 'AL POR MAYOR' );

--- TABLA PAGO

INSERT INTO PAGO (PAGO\_ID, FACTURA\_ID, FORMAPAGO, MONTO, DESCRIPCION)

VALUES (1, 'EFECTIVO', 1000.00, 'PAGO CORRECTO');

INSERT INTO PAGO (PAGO\_ID, FACTURA\_ID, FORMAPAGO, MONTO, DESCRIPCION)

VALUES (2, 'CREDITO', 2000.00, 'PAGO CORRECTO');

INSERT INTO PAGO (PAGO\_ID, FACTURA\_ID, FORMAPAGO, MONTO, DESCRIPCION)

VALUES (3, 'EFECTIVO', 3000.00, 'PAGO CORRECTO');

INSERT INTO PAGO (PAGO\_ID, FACTURA\_ID, FORMAPAGO, MONTO, DESCRIPCION)

VALUES (4, 'CREDITO', 4000.00, 'PAGO CORRECTO');

INSERT INTO PAGO (PAGO\_ID, FACTURA\_ID, FORMAPAGO, MONTO, DESCRIPCION)

VALUES (5, 'EFECTIVO', 5000.00, 'PAGO CORRECTO');

# Diseñar el prototipo de la aplicación completa, este tiene que contener:

## El menú principal y los diferentes módulos que tiene la aplicación

## Todas las entradas del sistema (pantallas que llama el sistema en los diferentes programas).

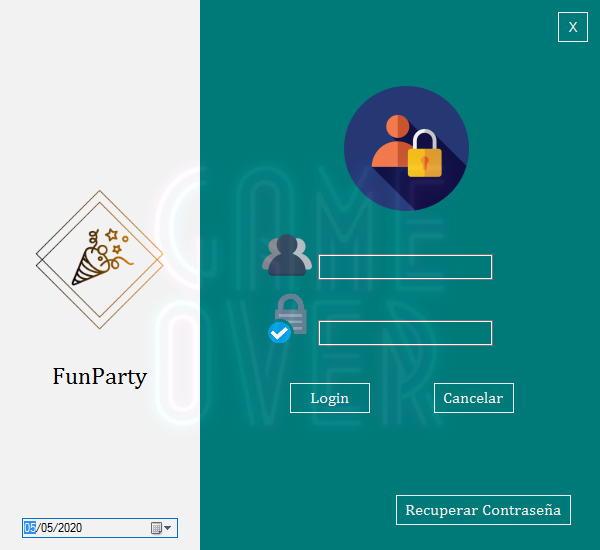
## Todas las salidas del sistema, representar la salidas usando pantallas.

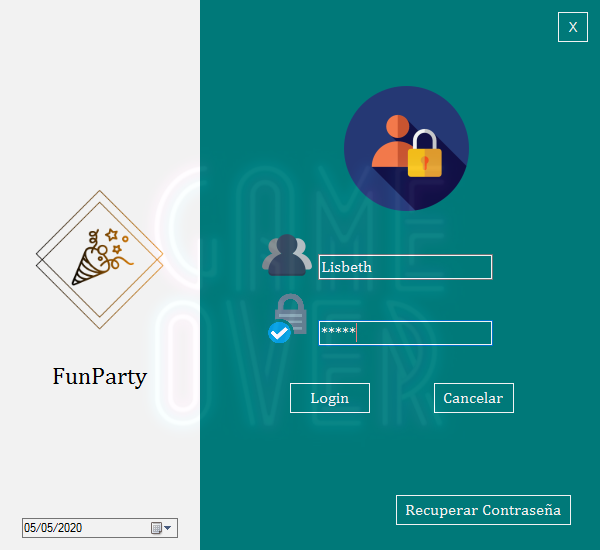


















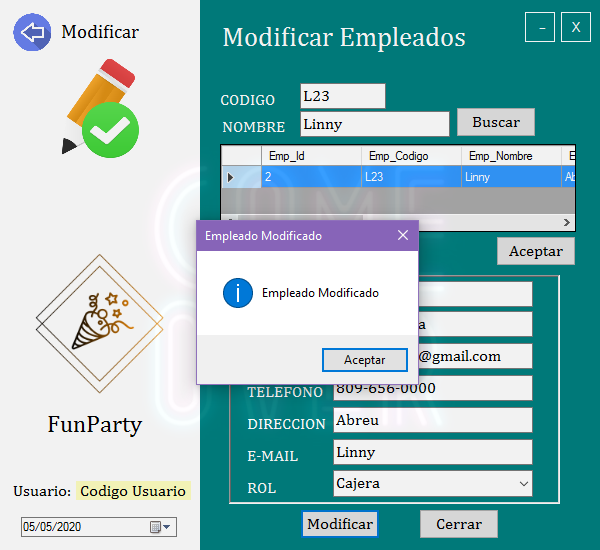


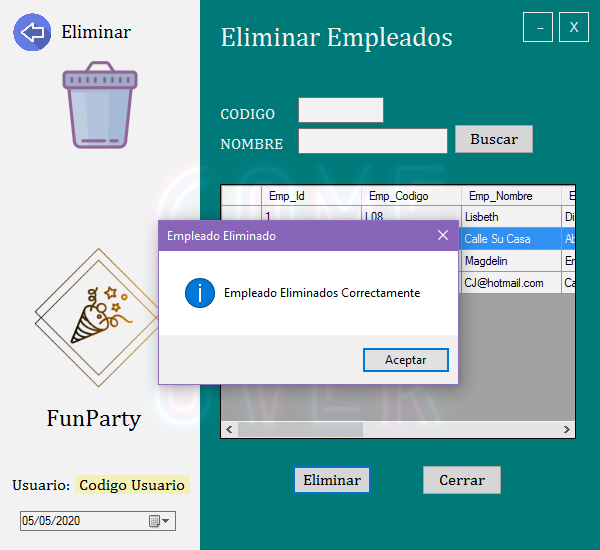
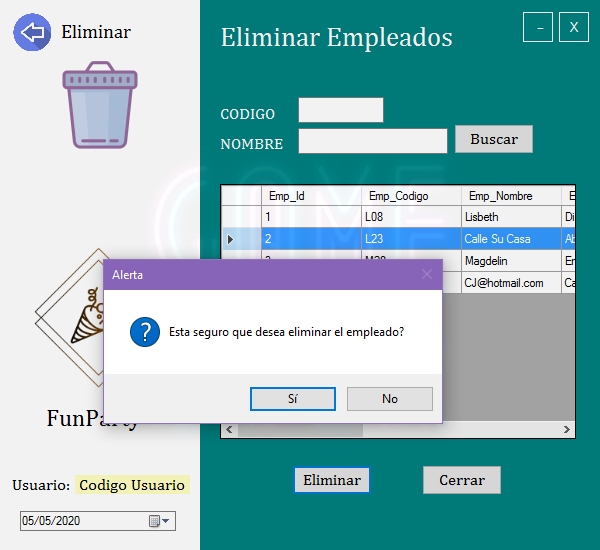




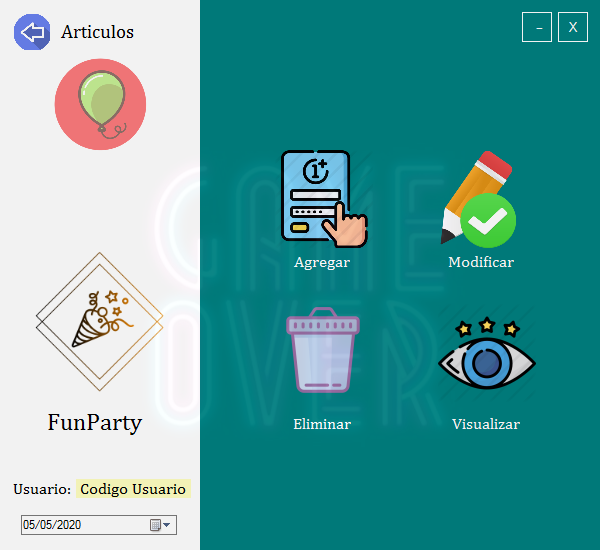
















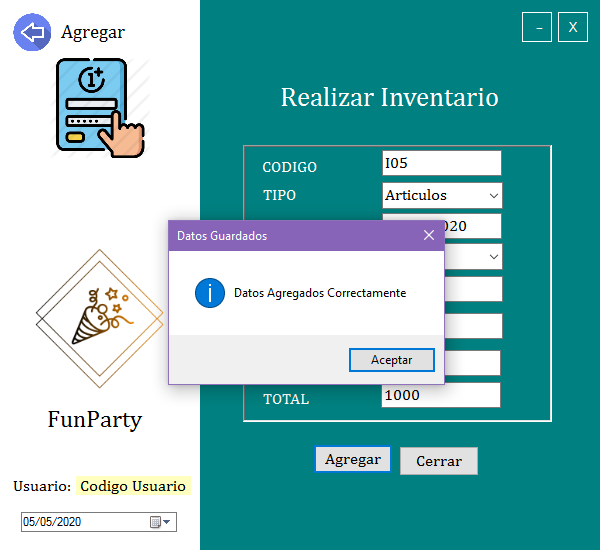












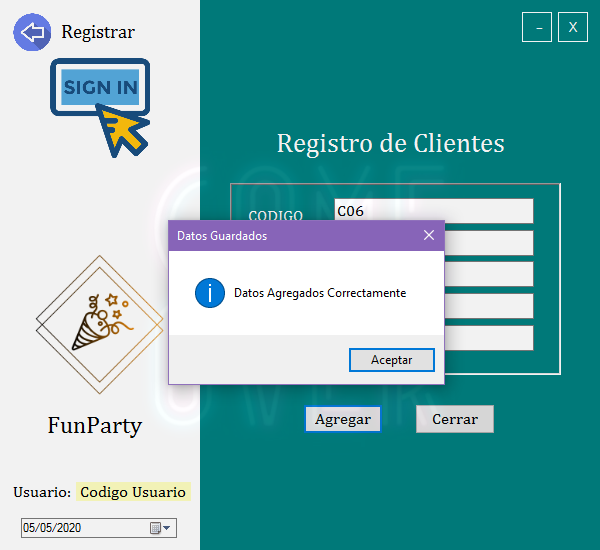






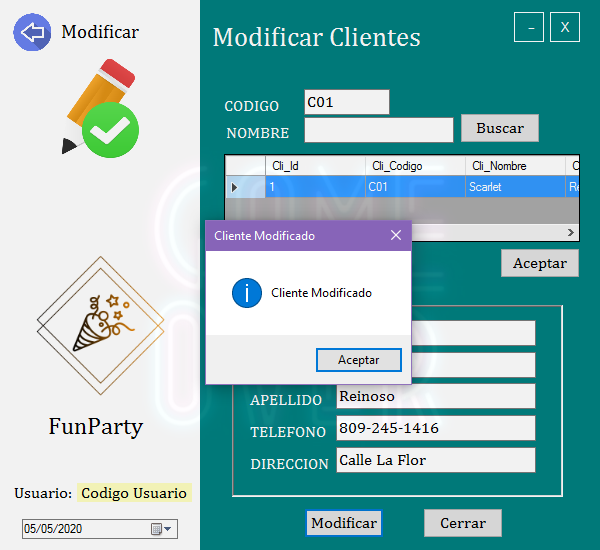






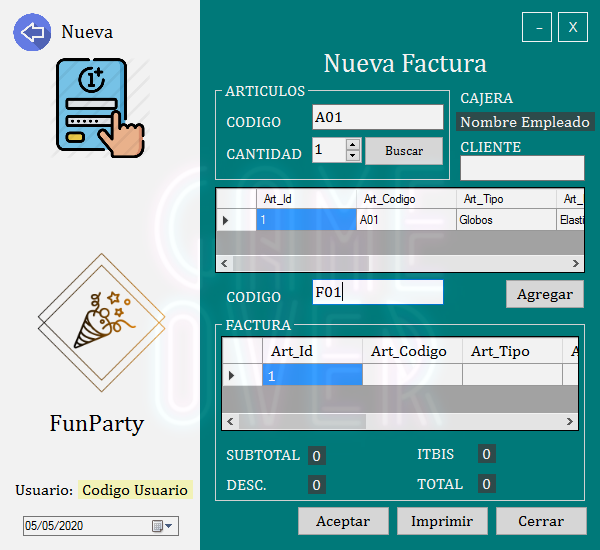


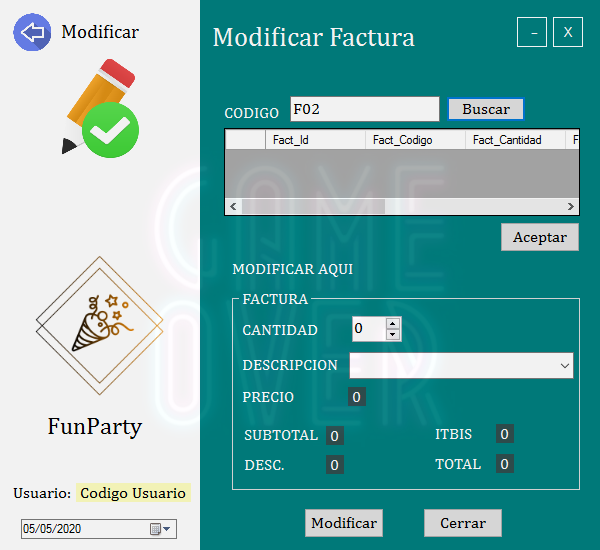












# Iniciar la codificación de los Programas

CONEXION BASE DE DATOS

namespace FunPartyApplication

{

public class BDComun

{

public static SqlConnection ObtenerConexion()

{

SqlConnection Conn = new SqlConnection("server=DESKTOP-4HVOAD4 ; database=DBFunParty ; integrated security = true");

Conn.Open();

return Conn;

}

}

}

USUARIOS

namespace FunPartyApplication

{

public class UsuarioD

{

public static int CrearCuentas(string pUsuario, string pContrasena)

{

int resultado = 0;

SqlConnection conn = BDComun.ObtenerConexion();

SqlCommand comando = new SqlCommand(string.Format("Insert into Credenciales (Usuarios, Pass) values ('{0}','{1}')", pUsuario, pContrasena), conn);

resultado = comando.ExecuteNonQuery();

conn.Close();

return resultado;

}

public static int Autentificar(String pUsuario, String pContrasena)

{

int resultado = -1;

SqlConnection conn = BDComun.ObtenerConexion();

SqlCommand comando = new SqlCommand(string.Format("Select \* from Credenciales where Usuarios = '{0}' and PwdCompare('{1}', Pass) = 1", pUsuario, pContrasena), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

resultado = 50;

}

conn.Close();

return resultado;

}

public static int Modificar(string pUsuario, string pContrasena)

{

int resultado = 0;

SqlConnection conn = BDComun.ObtenerConexion();

SqlCommand comando = new SqlCommand(string.Format("update Credenciales set Pass='{0}' where Usuarios='{1}' ", pContrasena, pUsuario), conn);

resultado = comando.ExecuteNonQuery();

conn.Close();

return resultado;

}

}

}

EMPLEADOS

namespace FunPartyApplication

{

public class Empleado

{

public Int32 Emp\_Id { get; set; }

public string Emp\_Codigo { get; set; }

public string Emp\_Nombre { get; set; }

public string Emp\_Apellido { get; set; }

public string Emp\_Telefono { get; set; }

public string Emp\_Email { get; set; }

public string Emp\_Domicilio { get; set; }

public string Emp\_Rol { get; set; }

public Empleado() { }

public Empleado(Int32 pId, String pCodigo, String pNombre, String pApellido, String pTelefono, String pEmail, String pDomicilio, String pRol )

{

this.Emp\_Id = pId;

this.Emp\_Codigo = pCodigo;

this.Emp\_Nombre = pNombre;

this.Emp\_Apellido = pApellido;

this.Emp\_Telefono = pTelefono;

this.Emp\_Email = pEmail;

this.Emp\_Domicilio = pDomicilio;

this.Emp\_Rol = pRol;

}

}

}

namespace FunPartyApplication

{

public class EmpleadoD

{

public static int Agregar(Empleado pEmpleado)

{

int retorno = 0;

using (SqlConnection Conn = BDComun.ObtenerConexion())

{

SqlCommand Comando = new SqlCommand(string.Format("Insert Into Empleado (Emp\_Codigo, Emp\_Nombre, Emp\_Apellido, Emp\_Telefono, Emp\_Email, Emp\_Domicilio, Emp\_Rol) values ('{0}','{1}','{2}','{3}','{4}', '{5}', '{6}' )",

pEmpleado.Emp\_Codigo, pEmpleado.Emp\_Nombre, pEmpleado.Emp\_Apellido, pEmpleado.Emp\_Telefono, pEmpleado.Emp\_Email, pEmpleado.Emp\_Domicilio, pEmpleado.Emp\_Rol), Conn);

retorno = Comando.ExecuteNonQuery();

Conn.Close();

}

return retorno;

}

public static List<Empleado> Buscar(String pCodigo, String pNombre)

{

List<Empleado> Lista = new List<Empleado>();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Emp\_Id, Emp\_Codigo, Emp\_Nombre, Emp\_Apellido, Emp\_Telefono, Emp\_Email, Emp\_Domicilio, Emp\_Rol from Empleado where Emp\_Codigo like '%{0}%' or Emp\_Nombre like '%{1}%'", pCodigo, pNombre), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

Empleado pEmpleado = new Empleado();

pEmpleado.Emp\_Id = reader.GetInt32(0);

pEmpleado.Emp\_Codigo = reader.GetString(1);

pEmpleado.Emp\_Nombre = reader.GetString(2);

pEmpleado.Emp\_Apellido = reader.GetString(3);

pEmpleado.Emp\_Telefono = reader.GetString(4);

pEmpleado.Emp\_Email = reader.GetString(5);

pEmpleado.Emp\_Domicilio = reader.GetString(6);

pEmpleado.Emp\_Rol = reader.GetString(7);

Lista.Add(pEmpleado);

}

conn.Close();

return Lista;

}

}

public static Empleado ObtenerEmpleado(Int32 pId)

{

Empleado pEmpleado = new Empleado();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Emp\_Id, Emp\_Codigo, Emp\_Nombre, Emp\_Apellido, Emp\_Telefono, Emp\_Email, Emp\_Domicilio, Emp\_Rol from Empleado where Emp\_Id={0}", pId), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

pEmpleado.Emp\_Id = reader.GetInt32(0);

pEmpleado.Emp\_Codigo = reader.GetString(1);

pEmpleado.Emp\_Nombre = reader.GetString(2);

pEmpleado.Emp\_Apellido = reader.GetString(3);

pEmpleado.Emp\_Telefono = reader.GetString(4);

pEmpleado.Emp\_Email = reader.GetString(5);

pEmpleado.Emp\_Domicilio = reader.GetString(6);

pEmpleado.Emp\_Rol = reader.GetString(7);

}

conn.Close();

return pEmpleado;

}

}

public static int Modificar(Empleado pEmpleado)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Update Empleado set Emp\_Codigo='{0}', Emp\_Nombre='{1}', Emp\_Apellido='{2}', Emp\_Telefono='{3}', Emp\_Email='{4}', Emp\_Domicilio='{5}', Emp\_Rol='{6}' where Emp\_Id={7}",

pEmpleado.Emp\_Codigo, pEmpleado.Emp\_Nombre, pEmpleado.Emp\_Apellido, pEmpleado.Emp\_Telefono, pEmpleado.Emp\_Email, pEmpleado.Emp\_Domicilio, pEmpleado.Emp\_Rol, pEmpleado.Emp\_Id), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

public static int Eliminar (Int32 pId)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Delete From Empleado where Emp\_Id={0}", pId), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

}

}

ARTICULOS

namespace FunPartyApplication

{

public class Articulo

{

public Int32 Art\_Id { get; set; }

public string Art\_Codigo { get; set; }

public string Art\_Tipo { get; set; }

public string Art\_Marca { get; set; }

public string Art\_Color { get; set; }

public string Art\_Tamano { get; set; }

public Int32 Art\_Precio { get; set; }

public string Art\_Descripcion { get; set; }

public Articulo() { }

public Articulo(Int32 pId, String pCodigo, String pTipo, String pMarca, String pColor, String pTamano, Int32 pPrecio, String pDescripcion)

{

this.Art\_Id = pId;

this.Art\_Codigo = pCodigo;

this.Art\_Tipo = pTipo;

this.Art\_Marca = pMarca;

this.Art\_Color = pColor;

this.Art\_Tamano = pTamano;

this.Art\_Precio = pPrecio;

this.Art\_Descripcion = pDescripcion;

}

}

}

namespace FunPartyApplication

{

public class ArticuloD

{

public static int Agregar(Articulo pArticulo)

{

int retorno = 0;

using (SqlConnection Conn = BDComun.ObtenerConexion())

{

SqlCommand Comando = new SqlCommand(string.Format("Insert Into Articulo (Art\_Codigo, Art\_Tipo, Art\_Marca, Art\_Color, Art\_Tamano, Art\_Precio, Art\_Descripcion ) values ('{0}','{1}','{2}','{3}','{4}', '{5}','{6}' )",

pArticulo.Art\_Codigo, pArticulo.Art\_Tipo, pArticulo.Art\_Marca, pArticulo.Art\_Color, pArticulo.Art\_Tamano, pArticulo.Art\_Precio, pArticulo.Art\_Descripcion), Conn);

retorno = Comando.ExecuteNonQuery();

Conn.Close();

}

return retorno;

}

public static List<Articulo> Buscar(String pCodigo)

{

List<Articulo> Lista = new List<Articulo>();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Art\_Id, Art\_Codigo, Art\_Tipo, Art\_Marca, Art\_Color, Art\_Tamano, Art\_Precio, Art\_Descripcion from Articulo where Art\_Codigo like '%{0}%'", pCodigo), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

Articulo pArticulo = new Articulo();

pArticulo.Art\_Id = reader.GetInt32(0);

pArticulo.Art\_Codigo = reader.GetString(1);

pArticulo.Art\_Tipo = reader.GetString(2);

pArticulo.Art\_Marca = reader.GetString(3);

pArticulo.Art\_Color = reader.GetString(4);

pArticulo.Art\_Tamano = reader.GetString(5);

pArticulo.Art\_Precio = reader.GetInt32(6);

pArticulo.Art\_Descripcion = reader.GetString(7);

Lista.Add(pArticulo);

}

conn.Close();

return Lista;

}

}

public static Articulo ObtenerArticulo(Int32 pId)

{

Articulo pArticulo = new Articulo();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Art\_Id, Art\_Codigo, Art\_Tipo, Art\_Marca, Art\_Color, Art\_Tamano, Art\_Precio, Art\_Descripcion from Articulo where Art\_Id={0}", pId), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

pArticulo.Art\_Id = reader.GetInt32(0);

pArticulo.Art\_Codigo = reader.GetString(1);

pArticulo.Art\_Tipo = reader.GetString(2);

pArticulo.Art\_Marca = reader.GetString(3);

pArticulo.Art\_Color = reader.GetString(4);

pArticulo.Art\_Tamano = reader.GetString(5);

pArticulo.Art\_Precio = reader.GetInt32(6);

pArticulo.Art\_Descripcion = reader.GetString(7);

}

conn.Close();

return pArticulo;

}

}

public static int Modificar(Articulo pArticulo)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Update Articulo set Art\_Codigo='{0}', Art\_Tipo='{1}', Art\_Marca='{2}', Art\_Color='{3}', Art\_Tamano='{4}', Art\_Precio='{5}', Art\_Descripcion='{6}' where Art\_Id={7}",

pArticulo.Art\_Codigo, pArticulo.Art\_Tipo, pArticulo.Art\_Marca, pArticulo.Art\_Color, pArticulo.Art\_Tamano, pArticulo.Art\_Precio, pArticulo.Art\_Descripcion, pArticulo.Art\_Id), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

public static int Eliminar(Int32 pId)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Delete From Articulo where Art\_Id={0}", pId), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

}

}

INVENTARIOS

namespace FunPartyApplication

{

public class Inventario

{

public Int32 Inv\_Id { get; set; }

public String Inv\_Codigo { get; set; }

public String Inv\_Tipo { get; set; }

public String Inv\_Fecha { get; set; }

public String Inv\_Periodo { get; set; }

public Int32 Inv\_Cantidad { get; set; }

public String Inv\_Descripcion { get; set; }

public Int32 Inv\_Precio { get; set; }

public Int32 Inv\_Total { get; set; }

public Inventario() { }

public Inventario(Int32 pId, String pCodigo, String pTipo, String pFecha, String pPeriodo, Int32 pCantidad, String pDescripcion, Int32 pPrecio, Int32 pTotal)

{

this.Inv\_Id = pId;

this.Inv\_Codigo = pCodigo;

this.Inv\_Tipo = pTipo;

this.Inv\_Fecha = pFecha;

this.Inv\_Periodo = pPeriodo;

this.Inv\_Cantidad = pCantidad;

this.Inv\_Descripcion = pDescripcion;

this.Inv\_Precio = pPrecio;

this.Inv\_Total = pTotal;

}

}

}

namespace FunPartyApplication

{

public class InventarioD

{

public static int Agregar(Inventario pInventario)

{

int retorno = 0;

using (SqlConnection Conn = BDComun.ObtenerConexion())

{

SqlCommand Comando = new SqlCommand(string.Format("Insert Into Inventario (Inv\_Codigo, Inv\_Tipo, Inv\_Fecha, Inv\_Periodo, Inv\_Cantidad, Inv\_Descripcion, Inv\_Precio, Inv\_total) values ('{0}','{1}','{2}','{3}','{4}', '{5}', '{6}', '{7}')",

pInventario.Inv\_Codigo, pInventario.Inv\_Tipo, pInventario.Inv\_Fecha, pInventario.Inv\_Periodo, pInventario.Inv\_Cantidad, pInventario.Inv\_Descripcion, pInventario.Inv\_Precio, pInventario.Inv\_Total), Conn);

retorno = Comando.ExecuteNonQuery();

Conn.Close();

}

return retorno;

}

public static List<Inventario> Buscar(String pCodigo)

{

List<Inventario> Lista = new List<Inventario>();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Inv\_Id, Inv\_Codigo, Inv\_Tipo, Inv\_Fecha, Inv\_Periodo, Inv\_Cantidad, Inv\_Descripcion, Inv\_Precio, Inv\_total from Inventario where Inv\_Codigo like '%{0}%' ", pCodigo), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

Inventario pInventario = new Inventario();

pInventario.Inv\_Id = reader.GetInt32(0);

pInventario.Inv\_Codigo = reader.GetString(1);

pInventario.Inv\_Tipo = reader.GetString(2);

pInventario.Inv\_Fecha = reader.GetString(3);

pInventario.Inv\_Periodo = reader.GetString(4);

pInventario.Inv\_Cantidad = reader.GetInt32(5);

pInventario.Inv\_Descripcion = reader.GetString(6);

pInventario.Inv\_Precio = reader.GetInt32(7);

pInventario.Inv\_Total = reader.GetInt32(8);

Lista.Add(pInventario);

}

conn.Close();

return Lista;

}

}

public static Inventario ObtenerInventario(Int32 pId)

{

Inventario pInventario = new Inventario();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Inv\_Id, Inv\_Codigo, Inv\_Tipo, Inv\_Fecha, Inv\_Periodo, Inv\_Cantidad, Inv\_Descripcion, Inv\_Precio, Inv\_total from Inventario where Art\_Id={0}", pId), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

pInventario.Inv\_Id = reader.GetInt32(0);

pInventario.Inv\_Codigo = reader.GetString(1);

pInventario.Inv\_Tipo = reader.GetString(2);

pInventario.Inv\_Fecha = reader.GetString(3);

pInventario.Inv\_Periodo = reader.GetString(4);

pInventario.Inv\_Cantidad = reader.GetInt32(5);

pInventario.Inv\_Descripcion = reader.GetString(6);

pInventario.Inv\_Precio = reader.GetInt32(7);

pInventario.Inv\_Total = reader.GetInt32(8);

}

conn.Close();

return pInventario;

}

}

public static int Modificar(Inventario pInventario)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Update Inventario set Inv\_Codigo='{0}',Inv\_Tipo='{1}', Inv\_Fecha='{2}', Inv\_Periodo='{3}', Inv\_Cantidad='{4}', Inv\_Descripcion='{5}', Inv\_Precio='{6}', Inv\_Total='{7}' where Inv\_Id={8}",

pInventario.Inv\_Codigo, pInventario.Inv\_Tipo, pInventario.Inv\_Fecha, pInventario.Inv\_Periodo, pInventario.Inv\_Cantidad, pInventario.Inv\_Descripcion, pInventario.Inv\_Precio, pInventario.Inv\_Total, pInventario.Inv\_Id), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

public static int Eliminar(Int32 pId)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Delete From Inventario where Fact\_Id={0}", pId), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

}

}

CLIENTES

namespace FunPartyApplication

{

public class Cliente

{

public Int32 Cli\_Id { get; set; }

public string Cli\_Codigo { get; set; }

public string Cli\_Nombre { get; set; }

public string Cli\_Apellido { get; set; }

public string Cli\_Telefono { get; set; }

public string Cli\_Domicilio { get; set; }

public Cliente() { }

public Cliente (Int32 pId, String pCodigo, String pNombre, String pApellido, String pTelefono, String pDomicilio)

{

this.Cli\_Id = pId;

this.Cli\_Codigo = pCodigo;

this.Cli\_Nombre = pNombre;

this.Cli\_Apellido = pApellido;

this.Cli\_Telefono = pTelefono;

this.Cli\_Domicilio = pDomicilio;

}

}

}

namespace FunPartyApplication

{

public class ClienteD

{

public static int Agregar(Cliente pCliente)

{

int retorno = 0;

using (SqlConnection Conn = BDComun.ObtenerConexion())

{

SqlCommand Comando = new SqlCommand(string.Format("Insert Into Cliente (Cli\_Codigo, Cli\_Nombre, Cli\_Apellido, Cli\_Telefono, Cli\_Domicilio) values ('{0}','{1}','{2}','{3}','{4}' )",

pCliente.Cli\_Codigo, pCliente.Cli\_Nombre, pCliente.Cli\_Apellido, pCliente.Cli\_Telefono, pCliente.Cli\_Domicilio), Conn);

retorno = Comando.ExecuteNonQuery();

Conn.Close();

}

return retorno;

}

public static List<Cliente> Buscar(String pCodigo, String pNombre)

{

List<Cliente> Lista = new List<Cliente>();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Cli\_Id, Cli\_Codigo, Cli\_Nombre, Cli\_Apellido, Cli\_Telefono, Cli\_Domicilio from Cliente where Cli\_Codigo like '%{0}%' and Cli\_Nombre like '%{1}%'", pCodigo, pNombre), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

Cliente pCliente = new Cliente();

pCliente.Cli\_Id = reader.GetInt32(0);

pCliente.Cli\_Codigo = reader.GetString(1);

pCliente.Cli\_Nombre = reader.GetString(2);

pCliente.Cli\_Apellido = reader.GetString(3);

pCliente.Cli\_Telefono = reader.GetString(4);

pCliente.Cli\_Domicilio = reader.GetString(5);

Lista.Add(pCliente);

}

conn.Close();

return Lista;

}

}

public static Cliente ObtenerCliente(Int32 pId)

{

Cliente pCliente = new Cliente();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Cli\_Id, Cli\_Codigo, Cli\_Nombre, Cli\_Apellido, Cli\_Telefono, Cli\_Domicilio from Cliente where Cli\_Id={0}", pId), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

pCliente.Cli\_Id = reader.GetInt32(0);

pCliente.Cli\_Codigo = reader.GetString(1);

pCliente.Cli\_Nombre = reader.GetString(2);

pCliente.Cli\_Apellido = reader.GetString(3);

pCliente.Cli\_Telefono = reader.GetString(4);

pCliente.Cli\_Domicilio = reader.GetString(5);

}

conn.Close();

return pCliente;

}

}

public static int Modificar(Cliente pCliente)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Update Cliente set Cli\_Codigo='{0}', Cli\_Nombre='{1}', Cli\_Apellido='{2}', Cli\_Telefono='{3}', Cli\_Domicilio='{4}' where Cli\_Id={5}",

pCliente.Cli\_Codigo, pCliente.Cli\_Nombre, pCliente.Cli\_Apellido, pCliente.Cli\_Telefono, pCliente.Cli\_Domicilio, pCliente.Cli\_Id), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

public static int Eliminar(Int32 pId)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Delete From Cliente where Cli\_Id={0}", pId), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

}

}

FACTURA

namespace FunPartyApplication

{

public class Factura

{

public Int32 Fact\_Id { get; set; }

public String Fact\_Codigo { get; set; }

public Int32 Fact\_Cantidad { get; set; }

public Int32 Fact\_SubTotal { get; set; }

public Int32 Fact\_Descuento { get; set; }

public Int32 Fact\_Itbis { get; set; }

public Int32 Fact\_Total { get; set; }

public Factura() { }

public Factura(Int32 pId, String pCodigo, Int32 pCantidad, Int32 pSubTotal, Int32 pDescuento, Int32 pTotal)

{

this.Fact\_Id = pId;

this.Fact\_Codigo = pCodigo;

this.Fact\_Cantidad = pCantidad;

this.Fact\_SubTotal = pSubTotal;

this.Fact\_Descuento = pDescuento;

this.Fact\_Total = pTotal;

}

}

}

namespace FunPartyApplication

{

public class FacturaD

{

public static int Agregar(Factura pFactura)

{

int retorno = 0;

using (SqlConnection Conn = BDComun.ObtenerConexion())

{

SqlCommand Comando = new SqlCommand(string.Format("Insert Into Factura (Fact\_Codigo, Fact\_Cantidad, Fact\_SubTotal, Fact\_Descuento, Fact\_Itbis, Fact\_ToTal ) values ('{0}','{1}','{2}','{3}','{4}', '{5}')",

pFactura.Fact\_Codigo, pFactura.Fact\_Cantidad, pFactura.Fact\_SubTotal, pFactura.Fact\_Descuento, pFactura.Fact\_Itbis, pFactura.Fact\_Total), Conn);

retorno = Comando.ExecuteNonQuery();

Conn.Close();

}

return retorno;

}

public static List<Factura> Buscar(String pCodigo)

{

List<Factura> Lista = new List<Factura>();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Fact\_Id, Fact\_Codigo, Fact\_Cantidad, Fact\_SubTotal, Fact\_Descuento, Fact\_Itbis, Fact\_ToTal from Factura where Fact\_Codigo like '%{0}%' ", pCodigo), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

Factura pFactura = new Factura();

pFactura.Fact\_Id = reader.GetInt32(0);

pFactura.Fact\_Codigo = reader.GetString(1);

pFactura.Fact\_Cantidad = reader.GetInt32(2);

pFactura.Fact\_SubTotal = reader.GetInt32(3);

pFactura.Fact\_Descuento = reader.GetInt32(4);

pFactura.Fact\_Itbis = reader.GetInt32(5);

pFactura.Fact\_Total = reader.GetInt32(6);

Lista.Add(pFactura);

}

conn.Close();

return Lista;

}

}

public static Factura ObtenerFactura(Int32 pId)

{

Factura pFactura = new Factura();

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(string.Format("Select Fact\_Id, Fac\_Codigo, Fact\_Cantidad, Fact\_SubTotal, Fact\_Descuento, Fact\_Itbis, Fact\_ToTal from Factura where Art\_Id={0}", pId), conn);

SqlDataReader reader = comando.ExecuteReader();

while (reader.Read())

{

pFactura.Fact\_Id = reader.GetInt32(0);

pFactura.Fact\_Codigo = reader.GetString(1);

pFactura.Fact\_Cantidad = reader.GetInt32(2);

pFactura.Fact\_SubTotal = reader.GetInt32(3);

pFactura.Fact\_Descuento = reader.GetInt32(4);

pFactura.Fact\_Itbis = reader.GetInt32(5);

pFactura.Fact\_Total = reader.GetInt32(6);

}

conn.Close();

return pFactura;

}

}

public static int Modificar(Factura pFactura)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Update Factura set Fact\_Codigo='{0}', Fact\_Cantidad='{1}', Fact\_SubTotal='{2}', Fact\_Descuento='{3}', Fact\_Itbis='{4}', Fact\_Total='{5}' where Inv\_Id={6}",

pFactura.Fact\_Codigo, pFactura.Fact\_Cantidad, pFactura.Fact\_SubTotal, pFactura.Fact\_Descuento, pFactura.Fact\_Itbis, pFactura.Fact\_Total, pFactura.Fact\_Id), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

public static int Eliminar(Int32 pId)

{

int retorno = 0;

using (SqlConnection conn = BDComun.ObtenerConexion())

{

SqlCommand comando = new SqlCommand(String.Format("Delete From Factura where Fact\_Id={0}", pId), conn);

retorno = comando.ExecuteNonQuery();

conn.Close();

}

return retorno;

}

}

}

.

# Conclusiones y recomendaciones finales sobre la propuesta de diseño

# La ingeniería de software es una disciplina de la ingeniería que nos ayudan a desarrollar sistemas de software a tiempo y a la vez que se cumpla con las expectativas de calidad y que permanezca dentro del presupuesto. Sus 3 elementos importantes son: algoritmos, estructura de datos y documentos. El proyecto de software cumple con un ciclo de vida, para todo proyecto de software se debe elegir el modelo en el que se trabajara, es muy importante realizar prototipos de los productos de software para el mejor diseño y entendimiento de lo que requiere el cliente. Para esto necesitamos información adecuada, podemos utilizar cualquier técnica de recopilación de información siempre y cuando se haga de la forma correcta y constante comunicación con el cliente.

# Bibliografía

* Systems Analysis and Design in a Changin Word Fifth Edition PDF
* ACM/IEEE. Software Engineering Code of Ethics and Professional Practice, v5.2, 1999. Available at ((<http://www.acm.org/serving/se/code.htm>).
* ACM/IEEE. Computing Curricula 2001, Computer Science, Final Report. The Joint Task Force on Computing Curricula, IEEE Computer Society and Association for Computing Machinery. December 15, 2001.
* Elisabeth Anscombe. Intention. Basil Blackwell, 1958.
* John M. Artz. “Virtue vs. Utility: Alternative Foundations for Computer Ethics”. Proceedings of the Conference on Ethics in the Computer Age. Galtinburg, Tennessee, United States, November 11-13, 1994.