****

**Ingeniería en Sistemas de Información**

**Trabajo de Campo**

**Profesor:** Ing. Pablo Audoglio

**2022**

**Proyecto Final**

**Sistema de Gestión**



**Alumnos:**

Marcelo Palonsky

Matías Pena

Contenido

[Project Charter 5](#_Toc97234859)

[Alcance 7](#_Toc97234860)

[Registros de Stakeholders 7](#_Toc97234861)

[MINUTA DE REUNION 8](#_Toc97234862)

[Iteración I 9](#_Toc97234863)

[Lista de requerimiento 9](#_Toc97234864)

[Requerimientos no funcionales 9](#_Toc97234865)

[Modelo de dominio 10](#_Toc97234866)

[Diagramas de casos de uso 10](#_Toc97234867)

[Especificación de caso de uso 11](#_Toc97234868)

[Casos de usos de Alto Nivel 12](#_Toc97234869)

[Diagrama de robustez 14](#_Toc97234870)

[Diagrama de Secuencia 15](#_Toc97234871)

[Diagrama de clases 15](#_Toc97234872)

[Modelo de datos 16](#_Toc97234873)

[Patrón de Diseño 16](#_Toc97234874)

[Patrón Estructural - Facade 16](#_Toc97234875)

[Clasificación del patrón 16](#_Toc97234876)

[Intención 16](#_Toc97234877)

[Motivación 16](#_Toc97234878)

[Estructura del patrón 17](#_Toc97234879)

[Implementación: 17](#_Toc97234880)

[Iteración II 20](#_Toc97234881)

[Listado de Requerimientos 20](#_Toc97234882)

[Especificación de Casos de Uso 21](#_Toc97234883)

[Caso de uso de alto nivel 22](#_Toc97234884)

[Diagramas de Secuencia 27](#_Toc97234885)

[Diagrama de Clases 27](#_Toc97234886)

[Modelo de Datos 28](#_Toc97234887)

[Patrón de diseño 28](#_Toc97234888)

[SINGLETON (Creacional) 28](#_Toc97234889)

[Propósito 28](#_Toc97234890)

[Motivación 28](#_Toc97234891)

[Aplicabilidad 28](#_Toc97234892)

[Implementación: 29](#_Toc97234893)

[Iteración lll 30](#_Toc97234894)

[Listado de Requerimientos 30](#_Toc97234895)

[Modelo de dominio 31](#_Toc97234896)

[Diagramas de casos de uso 31](#_Toc97234897)

[Especificación de caso de uso 32](#_Toc97234898)

[Casos de usos de Alto Nivel 33](#_Toc97234899)

[Diagrama de robustez 34](#_Toc97234900)

[Diagrama de Secuencia 35](#_Toc97234901)

[Diagrama de clases 36](#_Toc97234902)

[Modelo de datos 36](#_Toc97234903)

[Patrón de Diseño 37](#_Toc97234904)

[Patrones de comportamiento - Memento 37](#_Toc97234905)

[Clasificación del patrón 37](#_Toc97234906)

[Intención 37](#_Toc97234907)

[Motivación. 37](#_Toc97234908)

[Estructura del patrón 37](#_Toc97234909)

[Implementación 38](#_Toc97234910)

[Clase Venta 38](#_Toc97234911)

[Clase Memento 38](#_Toc97234912)

[Clase CareTaker 39](#_Toc97234913)

[Iteración IV 40](#_Toc97234914)

[Listado de Requerimientos 40](#_Toc97234915)

[Modelo de dominio 40](#_Toc97234916)

[Diagramas de casos de uso 41](#_Toc97234917)

[Especificación de caso de uso 41](#_Toc97234918)

[Casos de usos de Alto Nivel 42](#_Toc97234919)

[Diagrama de robustez 43](#_Toc97234920)

[Diagrama de Secuencia 44](#_Toc97234921)

[Diagrama de clases 45](#_Toc97234922)

[Modelo de datos 46](#_Toc97234923)

[Patrón de Diseño 46](#_Toc97234924)

[Patrón Comportamiento - Estrategia 46](#_Toc97234925)

[Clasificación del patrón 47](#_Toc97234926)

[Intención 47](#_Toc97234927)

[Estructura del patrón 47](#_Toc97234928)

[Implementación: 48](#_Toc97234929)

[ANEXOS 49](#_Toc97234930)

[Procesos de Desarrollo de Software 49](#_Toc97234931)

[Modelos Prescriptivos 49](#_Toc97234932)

[El modelo de Cascada 49](#_Toc97234933)

[Modelos de Proceso Incremental 49](#_Toc97234934)

[Modelo DRA (Desarrollo Rápido de Aplicaciones) 50](#_Toc97234935)

[Modelo de Proceso Evolutivo 50](#_Toc97234936)

[Construcción de Prototipos 50](#_Toc97234937)

[Modelo en Espiral 51](#_Toc97234938)

[Modelo de Desarrollo Concurrente 52](#_Toc97234939)

[Modelos especializados de Proceso 52](#_Toc97234940)

[Desarrollo Basado en Componentes 52](#_Toc97234941)

[El Modelo de Métodos Formales 53](#_Toc97234942)

[Desarrollo del Software Orientado a Aspectos 53](#_Toc97234943)

[Proceso Unificado 53](#_Toc97234944)

[Modelo Iconix 54](#_Toc97234945)

[Características Principales 54](#_Toc97234946)

[Seleccionamos el modelo Iconix 55](#_Toc97234947)

[Estudio de viabilidad 55](#_Toc97234948)

[Viabilidad Legal 55](#_Toc97234949)

[Viabilidad Técnica 55](#_Toc97234950)

[Viabilidad Operativa 55](#_Toc97234951)

[Viabilidad Económico/Financiera 55](#_Toc97234952)

[Métricas orientada 56](#_Toc97234953)

[Puntos de casos de uso 56](#_Toc97234954)

[Método 56](#_Toc97234955)

[Puntos de caso de uso sin ajustar (UUCP) 56](#_Toc97234956)

[Factor de peso de los actores sin ajustar (UAW) 56](#_Toc97234957)

[Factor de peso de los casos de uso sin ajustar (UUCW) 57](#_Toc97234958)

[Puntos de caso de uso ajustados (UCP) 58](#_Toc97234959)

[Factores de complejidad técnica 58](#_Toc97234960)

[Factores ambientales 60](#_Toc97234961)

[Gestión de Riesgos 62](#_Toc97234962)

[Tabla de Riesgos 62](#_Toc97234963)

[Valores de Impacto: 63](#_Toc97234964)

[Tabla ordenada según Probabilidad e Impacto 63](#_Toc97234965)

[Hoja de información de Riesgo – Plan de reducción, supervisión y gestión del riesgo 64](#_Toc97234966)

[Informe Final 67](#_Toc97234967)

[Testing - Pruebas 67](#_Toc97234968)

[Introducción 67](#_Toc97234969)

[Testing Caja Blanca 68](#_Toc97234970)

[1 { 68](file:///C:\Users\Mateo\Google%20Drive\UAI\FRECICAR%20APP%20-%20DOC\Trabajo%20de%20Campo%20-%20Proyecto%20Final%20-%20Palonsky,%20Marcelo.docx#_Toc97234971)

[Diagrama de Flujo 69](#_Toc97234972)

[Determinación del conjunto básico de rutas 69](#_Toc97234973)

[Testing Caja Negra 70](#_Toc97234974)

[Resultados de la prueba: 71](#_Toc97234975)

[Análisis de valores limites 71](#_Toc97234976)

[Valores correctos 71](#_Toc97234977)

[Valores incorrectos 71](#_Toc97234978)

[Conclusión 72](#_Toc97234979)

[Política de Back Up 72](#_Toc97234980)

[Backup 72](#_Toc97234981)

[Política de Backup 72](#_Toc97234982)

[Tipos de Backups 73](#_Toc97234983)

[Arquitectura 73](#_Toc97234984)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | Marcelo Palonsky Matías Pena |  | Ing. Pablo Audoglio | 25/05/21 | Lanzar el inicio del proyecto |

# Project Charter

|  |  |
| --- | --- |
| **NOMBRE DEL PROYECTO** | **SIGLAS DEL PROYECTO** |
| **Frecicar** | **FR** |
| **DESCRIPCIÓN DEL PROYECTO:** | |
| El proyecto consiste en elaborar un sistema de Gestión con el fin de solucionar los inconvenientes administrativos del mismo y a su vez brindar las herramientas para que crezca. La Empresa cliente que solicito dicho sistema se llama Frecicar. El lugar donde se concretó dicho acuerdo fue en la ciudad de Rosario en el transcurso del año 2021 en el mes Marzo. | |

|  |  |  |
| --- | --- | --- |
| **OBJETIVOS DEL PROYECTO:** | | |
| ***CONCEPTO*** | ***OBJETIVOS*** | ***CRITERIO DE ÉXITO*** |
| ***1. ALCANCE*** | Diseñar y programar un software de gestión para una Empresa de Productos Plásticos. | El cliente queda satisfecho con el producto y se cumple con los requisitos de la catedra |
| ***2. TIEMPO*** | 6.5 meses | Cumplir con los tiempos prometidos |
| ***3. COSTO*** | No tiene un objetivo económico, solo académico. | -- |

|  |
| --- |
| **FINALIDAD DEL PROYECTO:** |
| El proyecto tiene como fin cumplir con los objetivos impuestos por las cátedras de Campo y Diploma, además de obtener un producto perfectamente implementable para una empresa como Frecicar o similar. |

|  |  |  |
| --- | --- | --- |
| **DESIGNACIÓN DEL PROJECT MANAGER DEL PROYECTO** | | |
| ***NOMBRE*** | Marcelo Palonsky – Matías Pena | ***NIVELES DE AUTORIDAD*** |
| ***REPORTA A*** | Ing. Pablo Audoglio | Full |
| ***SUPERVISA A*** | ----- |
|  |  |  |

|  |  |
| --- | --- |
| **CRONOGRAMA DE HITOS DEL PROYECTO** | |
| ***HITO O EVENTO SIGNIFICATIVO*** | ***FECHA PROGRAMADA*** |
| Inicio del Proyecto. | 03/01/21 |
| Presentación Iteración 1 | ------------- |
| Presentación Iteración 2 | ------------- |
| Presentación parcial del proyecto | 08/06/2021 |
| Fin del Proyecto. | TC |
| **ORGANIZACIONES O GRUPOS ORGANIZACIONALES QUE INTERVIENEN EN EL PROYECTO** | |
| ***ORGANIZACIÓN O GRUPO ORGANIZACIONAL*** | ***ROL QUE DESEMPEÑA*** |
| Profesor: Ing. Pablo Audoglio | Guía, seguimiento, control y aprobación del proyecto. |
| Marcelo Palonsky | Programación y Análisis |
| Matías Pena | Programación y Análisis |

|  |
| --- |
| **PRINCIPALES AMENAZAS DEL PROYECTO** |
| No cumplir con los requerimientos pedidos.  No aceptación del software por parte de los empleados de la empresa  Mala estimación de los tiempos  Mala definición de los requisitos |

|  |
| --- |
| **PRINCIPALES OPORTUNIDADES DEL PROYECTO** |
| Tomar como modelo para realizar una implementación web a futuro.  Ampliar con módulos liquidación de sueldo y Facturación. |

|  |
| --- |
| **EXCLUSIONES DEL PROYECTO:** |
| No se entrega al cliente el código fuente.  El costo no contempla un mantenimiento del sistema más allá de la implementación y puesta a punto. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **RESTRICCIONES DEL PROYECTO:** *FACTORES QUE LIMITAN EL RENDIMIENTO DEL PROYECTO, EL RENDIMIENTO DE UN PROCESO DEL PROYECTO, O LAS OPCIONES DE PLANIFICACIÓN DEL PROYECTO. PUEDEN APLICAR A LOS OBJETIVOS DEL PROYECTO O A LOS RECURSOS QUE SE EMPLEA EN EL PROYECTO.* | | | | |
| ***INTERNOS A LA ORGANIZACIÓN*** | | ***AMBIENTALES O EXTERNOS A LA ORGANIZACIÓN*** | | |
| Limitada cantidad de recursos humanos para el proyecto  Limitado tiempo de trabajo, debido a otros proyectos paralelos | | Distancia del cliente | | |
| **SUPUESTOS DEL PROYECTO:** *FACTORES QUE PARA PROPÓSITOS DE LA PLANIFICACIÓN DEL PROYECTO SE CONSIDERAN VERDADEROS, REALES O CIERTOS.* | | | | |
| ***INTERNOS A LA ORGANIZACIÓN*** | | | ***AMBIENTALES O EXTERNOS A LA ORGANIZACIÓN*** | |
| NA | | | NA | |
| **PRESUPUESTO PRELIMINAR DEL PROYECTO:** | | | | |
| ***CONCEPTO*** | | | | ***MONTO ($)*** |
| ***1. PERSONAL*** | Horas programación 🡪 260 Hs. | | | 195000 |
| Horas análisis, reuniones, pruebas 🡪 200 Hs. | | | 150000 |
| ***2. MATERIALES*** | El Software será en versión libre o developer | | | 0 |
| ***3. MAQUINAS*** | Se utilizarán maquinaria propia | | | 0 |
| ***4. OTROS COSTOS*** | Viáticos | | | 5000 |
| ***TOTAL LÍNEA BASE*** | | | | **345000** |
| ***5. RESERVA DE CONTINGENCIA*** |  | | | 10000 |
| ***6. RESERVA DE GESTIÓN*** |  | | | 0 |
| ***TOTAL PRESUPUESTO*** | | | | **355000** |

|  |  |  |  |
| --- | --- | --- | --- |
| **SPONSOR QUE AUTORIZA EL PROYECTO** | | | |
| ***NOMBRE*** | ***EMPRESA*** | ***CARGO*** | ***FECHA*** |
| Esteban Rawson | Frecicar | Dueño | 03/03/21 |
| Ing. Pablo Audoglio | UAI | Docente | 03/03/21 |

# Alcance

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | Marcelo Palonsky  Matías Pena |  | Audoglio, P | 08/03/2021 | Presentación |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTROL DE VERSIONES** | | | | | |
| Versión | Hecha por | Revisada por | Aprobada por | Fecha | Motivo |
| 1.0 | Matías Pena  Marcelo Palonsky |  | Audoglio p | 08/03/21 |  |
|  |  |  |  |  |  |

# Registros de Stakeholders

|  |  |
| --- | --- |
| NOMBRE DEL PROYECTO | SIGLAS DEL PROYECTO |
| Frecicar | **FR** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IDENTIFICACION | | | | EVALUACION | | | CLASIFICACION | |
| NOMBRE | ORGAN. | LOCALIZ. | ROL EN EL PROYECTO | EXPECTATIVAS PRINCIPALES | INF. POT. | FASE DE MAYOR INTERES | INTERNO / EXTERNO | APOYO / NEUTRAL / OPOSITOR |
| Esteban Rawson | Emp. | Rosario | Clienta | Software terminado | -- | Pedido a Proveedor | Externo | --- |
| Marcelo Palonsky | Prog. | Rosario | Progr. | Cumplimiento de los requisitos | --- | Todas | Interno | --- |
| Matías Pena | Prog. | Rosario | Prog. | Cumplimiento de los requisitos | --- | Todas | Interno | --- |
| Pablo Audoglio | Docente UAI | Rosario | Supervisor de proyecto | Cumplimiento de los requisitos | -- | Todas | Externo | soporte |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MINUTA DE REUNION | | | | | | |
| **Fecha:** | 21/04/2021 | **Hora Inicio:** | 18:15 | | **Hora Fin:** | 20:30 |
| **Lugar:** | Provincias unidas 1956 | | | | | |
| **Participantes** | | | | | | |
| **Cliente** | | | | **Empresa** | | |
| 1) Miguel Carlos Pena | | | | 1. Marcelo Palonsky 2. Matías Pena | | |
| **Objetivos de la Reunión** | | | | | | |
| 1) Tomar conocimiento de las necesidades y objetivos a lograr con el sistema  2) Los propósitos del mismo para establecer la metodología idónea y la tecnología a usar  3) Pautar la forma de entrega y forma. | | | | | | |
| **Temas Principales Tratados** | | | | | | |
| 1) Se estableció las metas primordiales a lograr  2) Se discutió de las futuras entregas y plazos  3) Al tener un mayor conocimiento de las necesidades se optó por diferentes caminos al planificado en un principio | | | | | | |
| **Temas Secundarios Tratados** | | | | | | |
| 1) Se tomó un mayor conocimiento de la primera entrega planificada.  2) Se estableció un contacto y horarios para poder consultar acerca del mismo | | | | | | |
| **Próximos Pasos** | | | | | | |
| 1) Comenzar a plantearnos el sistema  2) Documentar y codificar a la par de las consultas con el cliente  3) Pautar una entrega y fecha tentativa | | | | | | |

# Iteración I

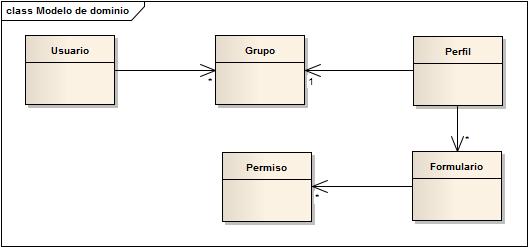
## Lista de requerimiento

|  |  |  |
| --- | --- | --- |
| **Ref #** | **Descripción** | **Cat** |
| **R1** | Un usuario iniciará sesión en el sistema ingresando su usuario y clave.  El sistema deberá autenticar su identidad, verificar si se encuentra activo y proporcionarle acceso a las acciones que puede realizar según los grupos a los que pertenece. Si un usuario pertenece a varios grupos las acciones que puede realizar se sumarán. | Evidente |
| **R2** | Seguridad el usuario debe tener un perfil de administrador. | Evidente |
| **R3** | El Administrador podrá agregar, modificar, y eliminar grupos del sistema teniendo en cuenta para estas operaciones: Nombre y descripción del grupo. Un grupo podrá ser eliminado si no tiene usuarios relacionados. Para llevar a cabo esta gestión podrá visualizar un listado de los grupos existentes. | Evidente |
| **R4** | El Administrador podrá agregar y modificar del sistema teniendo en cuenta para estas operaciones: código, grupo, vista, permiso y estado (activo/inactivo) del perfil. Un perfil podrá ser eliminado si no tiene grupos o vistas relacionados. Para llevar a cabo esta gestión podrá visualizar un listado de los perfiles existentes y Agregar o eliminar según grupo, formulario y permiso. El Administrador podrá especificar qué acciones puede realizar un perfil considerando que estas acciones pertenecen al conjunto vista/grupo. Esta clasificación permitirá lograr un mejor entendimiento a la hora de asignar acciones a un grupo. | Evidente |
| **R5** | El Administrador podrá agregar, modificar, y eliminar usuarios, así como también resetear la clave de los usuarios del sistema teniendo en cuenta para estas operaciones los siguientes atributos: Usuario, nombre y apellido e email. Un usuario podrá ser desactivado en cualquier momento. Para llevar a cabo esta gestión Podrá visualizar un listado de los usuarios existentes y filtrar por nombre y apellido, por usuario o email. | Evidente |

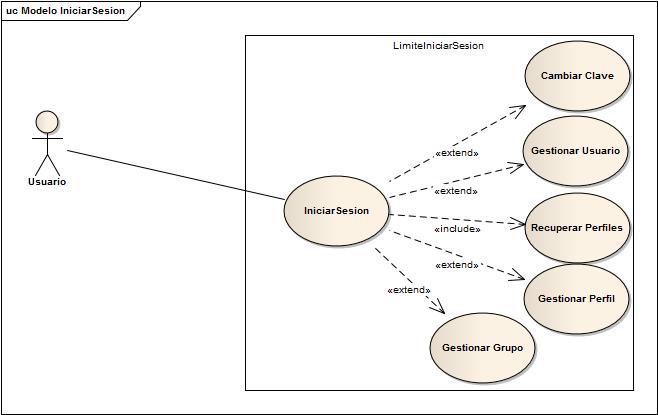
## Requerimientos no funcionales

|  |  |  |
| --- | --- | --- |
| **Ref #** | **Descripción** | **Cat** |
| R1 | Las claves de los usuarios deben estar encriptadas | Evidente |
| R2 | Los datos del sistema deben guardarse en una base de datos relacional | Evidente |
| R3 | Apto para ser utilizado en distintas plataformas y dispositivos | Evidente |
| R4 | La clave de un usuario administrador solo debe ser conocida por el mismo usuario. | Evidente |
| R5 | La tasa de ocurrencia de fallas no debe superar el umbral de 5 fallas por día. | Evidente |

## Modelo de dominio



## Diagramas de casos de uso



## Especificación de caso de uso

|  |  |  |  |
| --- | --- | --- | --- |
| Caso de uso: | Login | | **Fecha: 02/06/2022 Versión: 1.0** |
| **ID de caso de uso:** | CU-11-Login | | |
| **Tipo:** | Primario Real | | |
| **Interfaz de usuario:** | FrmInicio | | |
| **Creado por:** | Matías Pena – Marcelo Palonsky | | |
| **Fecha de creación:** | 02/06/22 | | |
| **Actor:** | Encargado de Logística, Administrador, Administrativo | | |
| **Descripción:** | El actor podrá iniciar sesión en el sistema proporcionando su nombre de usuario y su contraseña. Una vez iniciada la sesión el sistema deberá recuperar el perfil del usuario para redireccionarlo a la sección indicada y armar el menú. | | |
| **Pre – condiciones:** | El actor debe estar registrado en el sistema y estar Habilitado. | | |
| **Post – condiciones:** | El actor inicia sesión en el sistema | | |
| **CU Include** | CU-004 Recuperar Perfil | | |
| **CU Extend** | CU-002 Resetear Password | | |
| **Disparador** | El usuario hace clic en el botón Login. | | |
| **Flujo normal** | | **Flujo alternativo** | |
| 1. El actor hace clic en el Ingresar FrmInicio del sistema Plasticos Refine. | | \*.a.1 En cualquier momento ocurre un error grave.  \*.a.2 El Sistema muestra un cuadro de Plásticos Refine con error, registra los datos del error en el visor de sucesos del sistema operativo. | |
| 2. El Sistema abre el formulario login con los campos **Usuario** y **Contraseña** obligatorios vacíos a completar, el botón Ingresar y el link **Olvide la contraseña**. | |  | |
| 2. El Actor completa los campos **Usuario** y **Contraseña,** y hace clic en el botón **Ingresar.** | | 2.1 El Sistema muestra un cartel que informa **“Usuario o contraseña invalida.”,** y el botón **Aceptar.**  2.1.2 El Actor hace clic en el botón **Aceptar**.  2.1.3 El Sistema cierra el cartel y continúa su ejecución. | |
| 3. El sistema re direcciona al usuario y arma el menú según el Perfil al que pertenezca. | |  | |

## Casos de usos de Alto Nivel

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -12 Resetear Password | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Secretario/a, Encargado de Ventas, Encargado de Compras, Administrador | |
| **Descripción** | El Actor podrá resetear a password de su usuario actual. | |
| **Pre-Condiciones** | El Actor debe estar registrado en el sistema. |  |
| **Post-Condiciones** | El Actor realizó el reseteo de su password. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -13 Gestionar Usuarios | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El Actor podrá agregar usuarios, eliminar, habilitar, inhabilitar, modificar y filtrar usuarios ya existentes. Los usuarios pertenecerán a uno o varios grupos. | |
| **Pre-Condiciones** | El Actor debe contar con permisos de Administrador |  |
| **Post-Condiciones** | El Actor realizó la gestión de usuarios correctamente. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

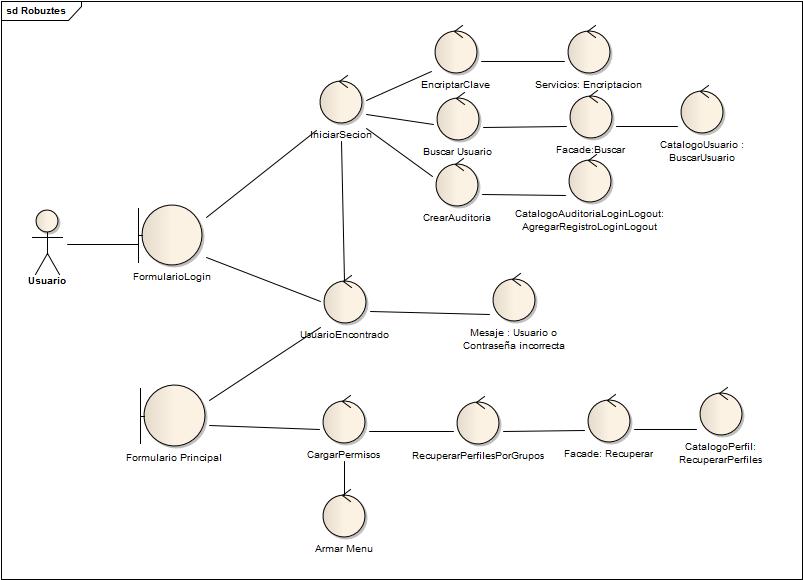
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU-14 Recuperar Perfil | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Sistema | |
| **Descripción** | El actor recupera los permisos del usuario logueado según los grupos a los que pertenece el usuario y el formulario al que se desea acceder. | |
| **Pre-Condiciones** | El actor debe estar logueado en el sistema. |  |
| **Post-Condiciones** | Perfiles Recuperados. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -15 Gestionar Grupos | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá agregar nuevos grupos y modificar la descripción o eliminar grupos existentes. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar grupos. |  |
| **Post-Condiciones** | El actor agregó, modificó o eliminó un grupo. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

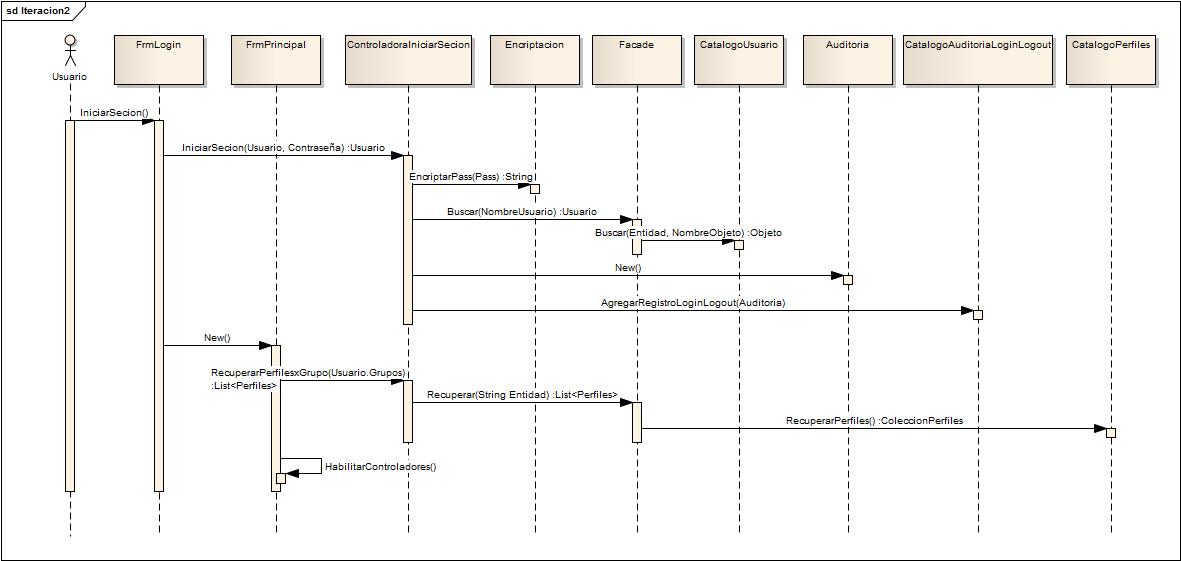
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -16 Gestionar Perfiles | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El Actor podrá crear perfiles para los diferentes grupos existentes asignándole permisos sobre los diferentes formularios, también podrá modificarlos agregándole o eliminándole permisos y/o formularios. | |
| **Pre-Condiciones** | El Actor debe poseer permisos de Administrador |  |
| **Post-Condiciones** | El Actor realizó la gestión de perfiles correctamente. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -17 Cambiar Contraseña | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Secretario/a, Encargado de Ventas, Encargado de Compras, Administrador | |
| **Descripción** | El Actor podrá cambiar la contraseña de usuario actual. | |
| **Pre-Condiciones** | El Actor debe iniciar sesión antes de efectuar el cambio de contraseña |  |
| **Post-Condiciones** | El Actor cambió su contraseña correctamente. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

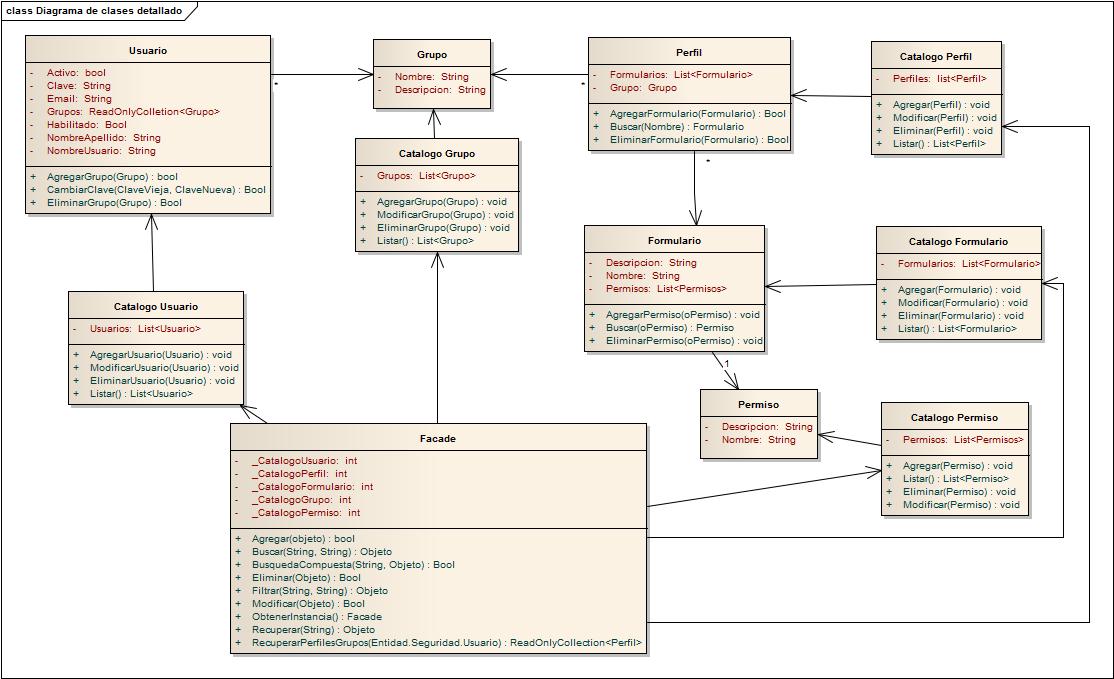
## Diagrama de robustez



## Diagrama de Secuencia



## Diagrama de clases



## Modelo de datos



## Patrón de Diseño

### Patrón Estructural - Facade

El patrón **FACADE** proporciona una interfaz unificada para un conjunto de interfaces de un sistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

Clasificación del patrón  
Estructural

IntenciónProporcionar una interfaz simplificada para un grupo de subsistemas o un sistema complejo.

Motivación  
Crear una clase Facade con el fin de poder acceder de manera simple, a través de la misma, a todos los catálogos del subsistema *Seguridad*, ocultando la visibilidad de los mismos.

### Estructura del patrón



|  |  |
| --- | --- |
| **Pattern** | Facade |
| **Based on** | Facade |
| **Because** | Definir una interfaz para los catálogos del  modelo de seguridad |
| **Where** | Usuario is user Administrator Facade is Facade |
| **Comments** | Facade permite unificar los distintos subsistemas |

### Implementación:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections.ObjectModel;

namespace Modelo.Seguridad

{

public class Facade

{

private static Facade \_Instancia;

public static Facade ObtenerInstancia()

{

if (\_Instancia == null) \_Instancia = new Facade();

return \_Instancia;

}

public bool Agregar(object objeto)

{

if (objeto is Entidades.Seguridad.Usuario)

{

return RepositorioUsuarios.ObtenerInstancia().AgregarUsuario((Entidades.Seguridad.Usuario)objeto);

}

if (objeto is Entidades.Seguridad.Grupo)

{

return RepositorioGrupos.ObtenerInstancia().AgregarGrupo((Entidades.Seguridad.Grupo)objeto);

}

else

{

return RepositorioPerfiles.ObtenerInstancia().AgregarPerfil((Entidades.Seguridad.Perfil)objeto);

}

}

public object Recuperar(string entidad)

{

if (entidad == "Usuario") return RepositorioUsuarios.ObtenerInstancia().RecuperarUsuarios();

if (entidad == "Perfil") return RepositorioPerfiles.ObtenerInstancia().RecuperarPerfiles();

if (entidad == "Grupo") return RepositorioGrupos.ObtenerInstancia().RecuperarGrupos();

if (entidad == "Formulario") return RepositorioFormularios.ObtenerInstancia().RecuperarPermisos();

if (entidad == "Permiso") return RepositorioPermisos.ObtenerInstancia().RecuperarPermisos();

else throw new Exception("No se puede Recuperar con la entidad especificada.");

}

public bool Eliminar(object Objeto)

{

if (Objeto is Entidades.Seguridad.Usuario)

{

return RepositorioUsuarios.ObtenerInstancia().EliminarUsuario((Entidades.Seguridad.Usuario)Objeto);

}

if (Objeto is Entidades.Seguridad.Grupo)

{

return RepositorioGrupos.ObtenerInstancia().EliminarGrupo((Entidades.Seguridad.Grupo)Objeto);

}

else

{

return RepositorioPerfiles.ObtenerInstancia().EliminarPerfil((Entidades.Seguridad.Perfil)Objeto);

}

}

public bool Modificar(object Objeto)

{

if (Objeto is Entidades.Seguridad.Usuario) return RepositorioUsuarios.ObtenerInstancia().ModificarUsuario((Entidades.Seguridad.Usuario)Objeto);

if (Objeto is Entidades.Seguridad.Grupo) return RepositorioGrupos.ObtenerInstancia().ModificarGrupo((Entidades.Seguridad.Grupo)Objeto);

else return RepositorioPerfiles.ObtenerInstancia().ModificarPerfil((Entidades.Seguridad.Perfil)Objeto);

}

public object Buscar(string Entidad, string NombreObjeto)

{

if (Entidad == "Usuario") return RepositorioUsuarios.ObtenerInstancia().BuscarUsuario(NombreObjeto);

if (Entidad == "Grupo") return RepositorioGrupos.ObtenerInstancia().BuscarGrupo(NombreObjeto);

if (Entidad == "Perfil") return RepositorioPerfiles.ObtenerInstancia().BuscarPerfil(NombreObjeto);

else throw new Exception("No se puede Buscar con la entidad especificada.");

}

public object Filtrar(string Entidad, string Criterio)

{

if (Entidad == "Usuario") return RepositorioUsuarios.ObtenerInstancia().FiltrarUsuarios(Criterio);

if (Entidad == "Grupo") return RepositorioGrupos.ObtenerInstancia().FiltrarGrupos(Criterio);

if (Entidad == "Perfiles") return RepositorioPerfiles.ObtenerInstancia().FiltrarPerfiles(Criterio);

else throw new Exception("No se puede filtar con la entidad especificada.");

}

public bool BusquedaCompuesta(string Entidad, object Objeto)

{

if (Entidad == "Usuario") return RepositorioUsuarios.ObtenerInstancia().BuscarGruposUsuario((Entidades.Seguridad.Grupo)Objeto);

if (Entidad == "Grupo") return RepositorioPerfiles.ObtenerInstancia().BuscarGrupoPerfil((Entidades.Seguridad.Grupo)Objeto);

else throw new Exception("No se puede Buscar con la entidad especificada.");

}

public ReadOnlyCollection<Entidades.Seguridad.Perfil> RecuperarPerfilesGrupos(Entidades.Seguridad.Usuario Usuario)

{

return RepositorioPerfiles.ObtenerInstancia().RecuperarPerfilesGrupos(Usuario);

}

}

}

# Iteración II

## Listado de Requerimientos

|  |  |  |
| --- | --- | --- |
| **Ref #** | **Función** | **Cat.** |
| **R1** | **Realizar Orden de Compra:** La Orden de Compra tendrán Forma de Pago, Forma de Envió, Proveedor, Detalles de Orden, Cantidad (Materia Prima), Fecha, Código, Estado y un Total. | Evidente |
| **R2** | **Gestionar Proveedor:** Visualiza todos los proveedores del sistema. El proveedor tendrá Email, Razón social, Teléfono, Dirección de calle, Dirección Numero, Cuit. | Evidente |
| **R3** | El sistema deberá filtrar las materias primas mediante el proveedor seleccionado en la orden de compra. | Evidente |
| **R4** | El sistema deberá calcular el subtotal por medio de la cantidad de materia prima agregada o eliminada del detalle de orden de compra. | Evidente |
| **R5** | El sistema deberá calcular por medio de los subtotales de cada detalle de orden de compra, el total de orden de compra. | Evidente |
| **R6** | El sistema deberá validar que no existan Materias Primas duplicadas en un mismo detalle de Orden de Compra. | Evidente |
| **R7** | El sistema deberá asignar la fecha actual y un código autogenerado. | Evidente |
| **R8** | El sistema deberá generar un reporte por cada orden de compra realizada. | Evidente |
| **R9** | El sistema deberá generar un reporte sobre las Materias Primas más solicitadas y de que proveedor proviene la misma.  El reporte debe solicitar como datos obligatorios una Fecha desde y una Fecha Hasta, de esta manera filtrara de forma necesaria los datos a representar. | Evidente |
| **R10** | El sistema deberá indicar aquella materia prima que este por debajo de su cantidad mínima pretendida en “Amarillo”, Mostrar en “Rojo” cuando no se encuentren materias primas en stock. | Evidente |
| **R11** | **Realizar Remito:** El Remito tendrá Fecha, Proveedor, Orden de Compra a asociada, cantidad (Materia Prima) y Detalles de remitos. | Evidente |
| **R12** | El sistema deberá validar que la cantidad ingresada sea menor a la cantidad representada en la orden de compra. | Evidente |
| **R13** | El sistema deberá validar que no exista detalle de remitos duplicados. | Evidente |
| **R14** | El sistema deberá generar un reporte por cada remito realizado. | Evidente |
| **R15** | El sistema deberá cambiar el estado de la orden de compra de “Realizado” a “Pendiente” si se registra un remito sin sus faltantes de detalles en 0 (cero).  El sistema deberá cambiar el estado de la orden de compra de “Realizado” a “Finalizado” o de “Pendiente” a “Finalizado” si se registra un remito con sus faltantes de detalles en 0 (cero). | Evidente |

## Especificación de Casos de Uso

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Realizar Orden de Compra** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **ID de caso de uso:** | CU – 01 – Realizar Orden de Compra | | |
| **Tipo:** | Primario Real | | |
| **Interfaz de usuario:** | FrmOrdenCompra | | |
| **Creado por:** | Matías Pena – Marcelo Palonsky | | |
| **Fecha de creación:** | 02/06/21 | | |
| **Actor:** | Encargado de Logística, Administrador, Administrativo | | |
| **Descripción:** | Permite generar una orden de compra cuando se necesite realizar un pedido de materia prima a proveedor.  La orden de compra es enviada al Proveedor con los siguientes datos: Fecha de emisión, Forma de envió y Forma de pago de la materia prima, nombre del proveedor, Detalle de orden de compra con la cantidad deseada de cada materia prima. | | |
| **Pre – condiciones:** | El actor debe estar registrado en el sistema y estar Habilitado. | | |
| **Post – condiciones:** | El Actor Realizo una Orden de Compra Correctamente. | | |
| **CU Extend** |  | | |
| **Disparador** | El usuario hace clic en el botón Realizar Orden Compra. | | |
| **Flujo normal** | | **Flujo alternativo** | |
| 1: El actor selecciona en el menú de acción la opción Realizar Orden. | | 1. A: El Actor hace click a cancelar.  1. B: El sistema cierra el formulario. | |
| 2: El sistema abre el formulario FRMOrdenCompra mostrando la fecha actual, un botón para seleccionar el proveedor, los combos de Forma de envió y Forma de Pago, la grilla de materia prima (filtrado por el proveedor seleccionado.), un textbox (para ingresar la cantidad de materia prima deseada), botones de agregar y quitar detalle, una grilla de detalle de orden de compra, botón de realizar Orden de Compra, Botón cancelar. | | 2. A: El Actor hace click a cerrar formulario.  2. B: El sistema cierra el formulario | |
| 3: El actor hace click en seleccionar proveedor | | 3. A: El Actor no selecciona un proveedor en el caso de uso Gestionar Proveedor.  3. B: El sistema mostrara un mensaje de que no se seleccionó un proveedor. | |
| 4: El sistema abre el formulario “Gestionar Proveedor”, con solo permisos de lectura. | |  | |
| 5: El actor seleccionar un proveedor de la grilla. | | 5.A: El actor puede cancelar la operación. | |
| 6: El sistema filtra la grilla materia prima por medio del proveedor seleccionado. | |  | |
| 7: El actor selecciona una materia prima de la grilla, da la cantidad deseada, y da al botón agregar. (Este paso puede reiterarse las veces necesarias). | | 7. A: El Actor puede seleccionar un detalle de orden de compra y dar al botón quitar. (Este paso puede reiterarse las veces necesarias).  7. B: El sistema Actualiza la grilla de detalle de orden de compra. | |
| 8: El sistema actualiza la grilla de detalle de orden de compra. | |  | |
| 9: El actor da a Realizar Pedido. | | 9. A: El Actor cancela el pedido.  9. B: El sistema cierra el formulario. | |
| 10: El sistema agrega un pedido. | |  | |

## Caso de uso de alto nivel

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -02 Gestionar Proveedor | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá Agregar o Modificar nuevos proveedores los cuales poseen un Cuit (único), Razón Social, Teléfono, Dirección Domicilio y Email. También podrá eliminar proveedores existentes. También podrá visualizar los proveedores existentes utilizando un filtro por Razón Social | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar proveedores. |  |
| **Post-Condiciones** | El actor Agregó, Modificó o Eliminó un Proveedor. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -03 Gestionar Forma de envío. | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá Agregar y Modificar Formas de Envíos los cuales poseen un Nombre y una Descripción. También podrá eliminar Formas de envíos existentes. También podrá visualizar las Formas de Envíos con un filtro por Nombre. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Forma de envíos. |  |
| **Post-Condiciones** | El actor agregó, modificó o eliminó un Forma de envío. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -04 Gestionar Forma de Pago | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá Agregar y Modificar Formas de Pagos los cuales poseen un Nombre y una Descripción. También podrá eliminar Formas de Pagos existentes. También podrá visualizar las Formas de Pagos con un filtro por Nombre. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Formas de pagos. |  |
| **Post-Condiciones** | El actor agregó, modificó o eliminó un Forma de pago. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

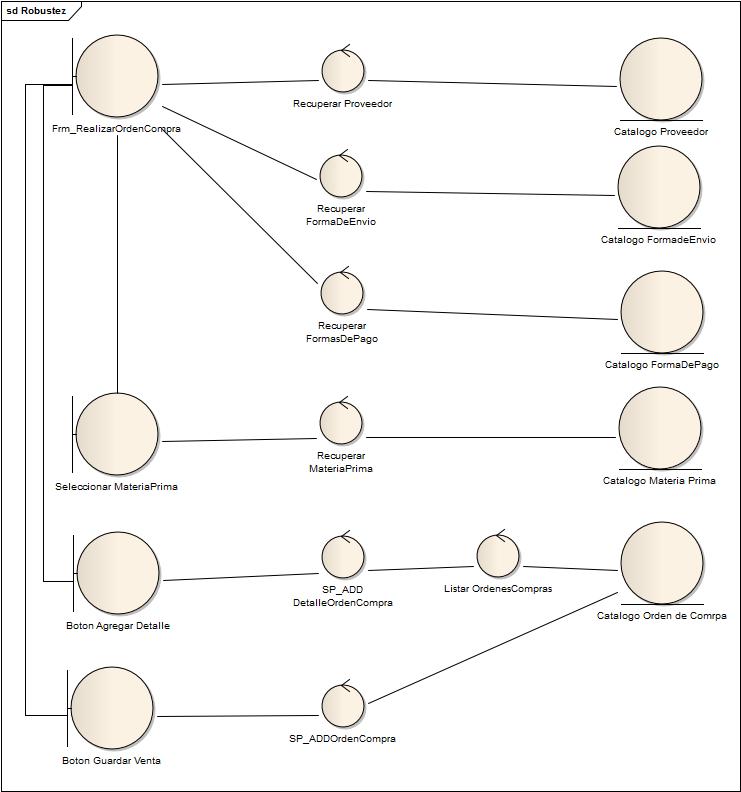
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -05 Gestionar Categoría | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá Agregar y Modificar Categorías los cuales poseen un Nombre y una Descripción. También podrá eliminar Categorías existentes. También podrá visualizar las Categorías con un filtro por Nombre. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Categorías. |  |
| **Post-Condiciones** | El actor agregó, modificó o eliminó una Categoría. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -06 Gestionar Materias Primas | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador | |
| **Descripción** | El actor podrá Agrega o Modificar Materias Primas los cuales posee una cantidad en stock, una cantidad mínima, una categoría, un código identificativo, un costo unitario, el proveedor que la brinda, y una descripción. También podrá Eliminar Materias Primas existentes. También podrá visualizar mediante un filtrado por descripción. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Materias Primas. |  |
| **Post-Condiciones** | El actor agregó, modificó o eliminó una Materias Primas. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -07 Gestionar Ordenes Compras | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador, Administrativo | |
| **Descripción** | El actor podrá Consultar, Anular o imprimir Órdenes de compras existentes. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Órdenes de Compras. |  |
| **Post-Condiciones** | El actor consultó, Anuló o imprimió una Orden de compra. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

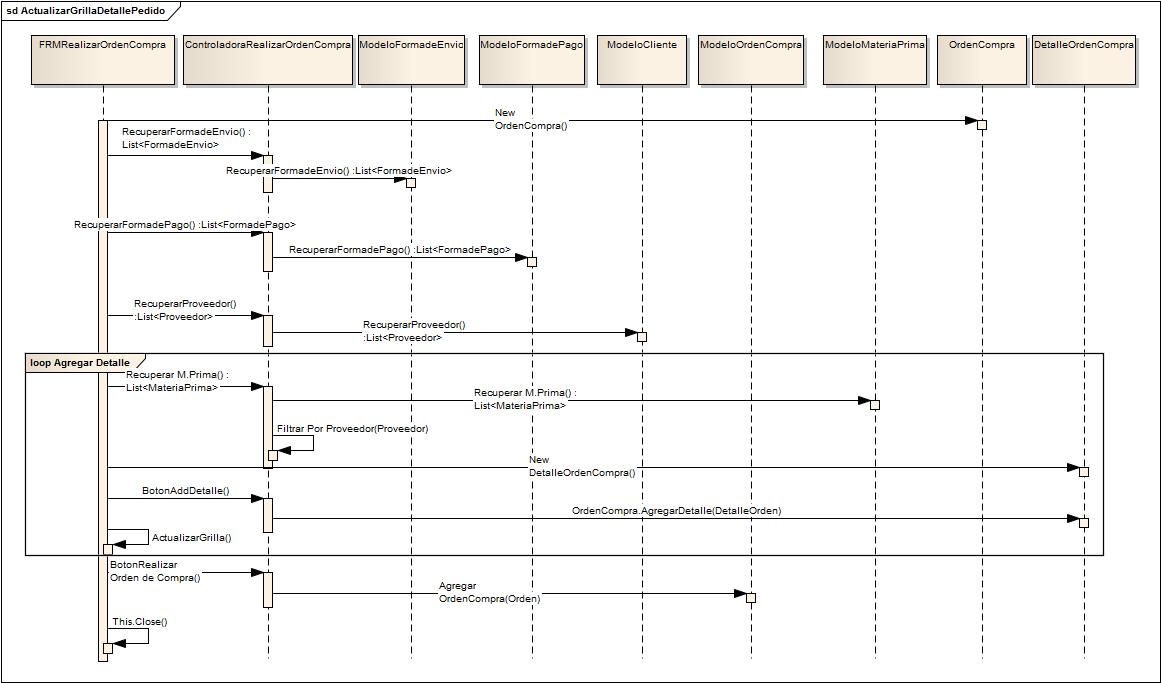
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -08 Gestionar Remitos Proveedor | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador, Encargado de Logística | |
| **Descripción** | El actor podrá Consultar o imprimir Remitos existentes. | |
| **Pre-Condiciones** | El actor debe poseer permisos para gestionar Remitos. |  |
| **Post-Condiciones** | El actor agregó, consultó o Imprimió un Remito. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -09 Realizar Ordenes Compras | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador, Administrativo | |
| **Descripción** | El actor podrá Agregar nuevas Órdenes de Compra las cuales poseen un código, una forma de envía, una forma de pago, un proveedor, un total, una fecha, un estado y un Detalle el cual tendrá una a su vez una materia prima, una cantidad, el faltante a incorporar y un subtotal. | |
| **Pre-Condiciones** | El actor debe poseer permisos para Realizar Órdenes de Compras. |  |
| **Post-Condiciones** | El actor Agregó una Orden de compra. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

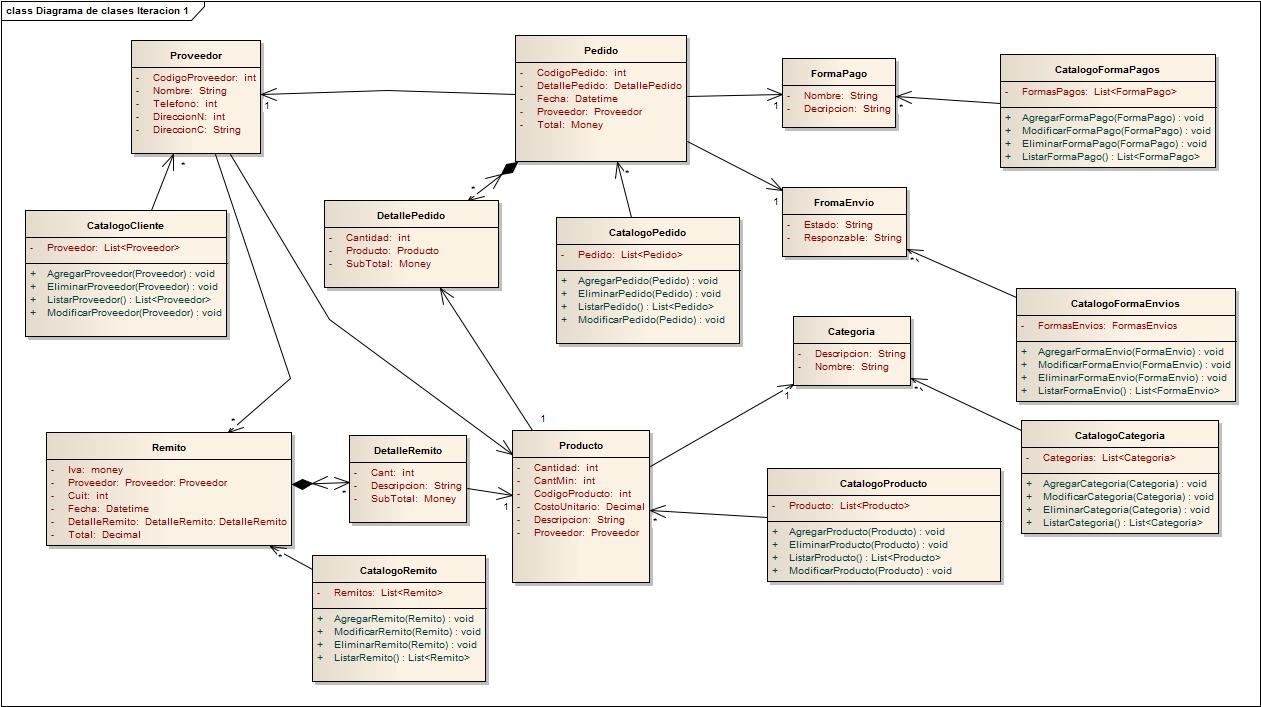
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021**  **Versión: 1.0** |
| **Nombre** | CU -10 Realizar Remitos Proveedor | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrador, Encargado de Logística | |
| **Descripción** | El actor podrá Agregar nuevos Remitos los cuales poseen un código, un proveedor, un total, una fecha, un Código de orden de compra a la cual hace referencia y un Detalle el cual tendrá una a su vez una materia prima, una cantidad y un subtotal. | |
| **Pre-Condiciones** | El actor debe poseer permisos para Realizar Remitos. |  |
| **Post-Condiciones** | El actor agregó un Remito. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |



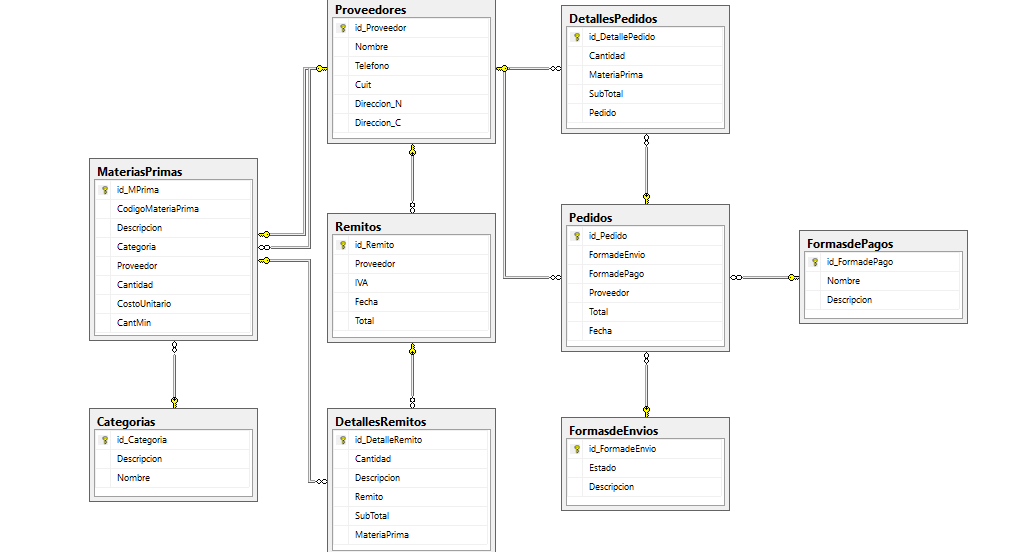
## Diagramas de Secuencia



## Diagrama de Clases



## Modelo de Datos



## Patrón de diseño

### SINGLETON (Creacional)

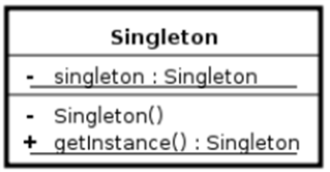
### Propósito

Garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella.

### Motivación

A veces es importante que algunas clases tengan exactamente una instancia. Una solución es hacer que sea la propia clase la responsable de su única instancia.

### Aplicabilidad

Deba haber exactamente una instancia de una clase, y esta debe ser accesible a los clientes desde un punto de acceso conocido. La única instancia debería ser extensible mediante herencia, y los clientes deberían ser capaces de usar una instancia extendida sin modificar su código.

### Implementación:

private static MappingCategoria \_Instancia;

public static MappingCategoria ObtenerInstancia()

{

if (\_Instancia == null) \_Instancia = new MappingCategoria();

return \_Instancia;

}

private MappingCategoria()

{

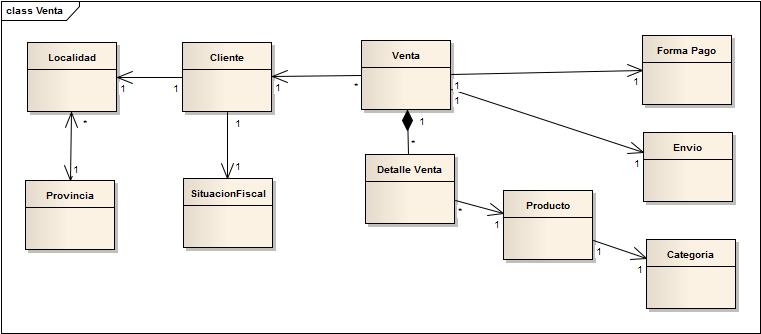
}

# Iteración lll

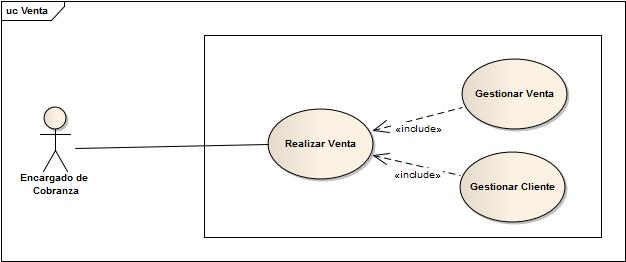
## Listado de Requerimientos

|  |  |  |
| --- | --- | --- |
| **Ref #** | **Función** | **Cat.** |
| **R1** | **Realizar Venta:** La Orden de Compra tendrán Forma de Pago, Forma de Envió, Proveedor, Detalles de Orden, Cantidad (Materia Prima), Fecha, Código, Estado y un Total. | Evidente |
| **R2** | **Gestionar Cliente:** El Cliente tiene un Código, Ciudad, Situación Fiscal, Razón Social, Teléfono, Domicilio, Cuit, Email. | Evidente |
| **R3** | El sistema deberá cargar la Fecha actual y un código autogenerado para la venta. | Evidente |
| **R4** | El sistema deberá validar que la cantidad de producto a vender no supere la actual en stock. | Evidente |
| **R5** | El sistema deberá validar que no existan productos iguales al agregar un detalle de venta. | Evidente |
| **R6** | El sistema deberá calcular el subtotal de cada detalle de venta agregada mediante la cantidad del producto solicitado. | Evidente |
| **R7** | El sistema Deberá calcular el Total de venta a medida que se agrega o elimina un detalle de venta. | Evidente |
| **R8** | El sistema deberá generar un reporte sobre los Productos más solicitados y a que Cliente fue realizada la venta.  El reporte debe solicitar como datos obligatorios una Fecha desde y una Fecha Hasta, de esta manera filtrara de forma necesaria los datos a representar. | Evidente |
| **R9** | El sistema deberá cambiar el estado de la venta de “Realizada” a “Pendiente” si la suma de los Pagos asociados no cubre el total del importe a pagar.  El sistema deberá cambiar el estado de la venta de “Realizado” a “Finalizado” o de “Pendiente” a “Finalizado” si la suma de los Pagos asociados cubre el total del importe a pagar. | Evidente |
| **R10** | El sistema deberá generar un movimiento de Debito en la cuenta corriente del Cliente al realizarse una Venta. | Evidente |
| **R11** | El sistema deberá actualizar la cuenta corriente cuando se efectué un movimiento de tipo Debito a un cliente. | Evidente |

## Modelo de dominio



## Diagramas de casos de uso



## Especificación de caso de uso

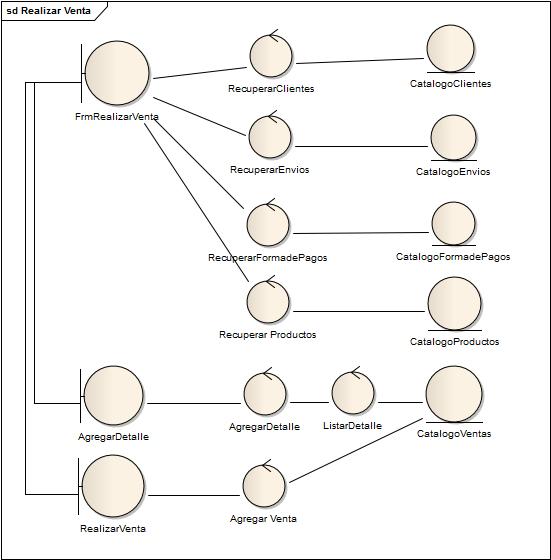
|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Venta** | | **Versión: 0.1** |
| **ID de caso de uso:** | CU-18-Venta | | |
| **Tipo:** | Primario Real | | |
| **Interfaz de usuario:** | FrmRealizarVenta | | |
| **Creado por:** | Matías Pena – Marcelo Palonsky | | |
| **Fecha de creación:** | 02/06/21 | | |
| **Actor:** | Administrativo | | |
| **Descripción:** | El actor podrá realizar Ventas de los productos deseados seleccionando el cliente, una forma de envió y una forma de pago. | | |
| **Pre – condiciones:** | El actor debe estar registrado en el sistema y Habilitado. También deberá tener los permisos necesarios para utilizar el formulario Realizar Venta | | |
| **Post – condiciones:** | El Realizó una Venta. | | |
| **CU Include** | CU-002-Gestionar Clientes. CU-003-Gestionar Ventas | | |
| **Disparador** | El usuario hace clic en el botón Realizar Venta. | | |
| **Flujo normal** | | **Flujo alternativo** | |
| 1. El sistema abre el formulario Realizar Venta.  Se visualiza las formas de envíos y las formas de pagos mediante combos para su selección, también se muestra y se carga las grillas de Productos y Detalles de Venta.  Se muestra un label de cliente vacío.  Se muestra un TextBox de Cantidad para indicar cuantos productos se desea vender.  Se visualiza los botones de Seleccionar Cliente, Realizar Venta, Cancelar, Agregar Detalle (ADD), Eliminar detalle (DEL) y Deshacer. | |  | |
| 2. El actor Selecciona el cliente mediante el botón “Seleccionar Cliente”. Incluye al caso de uso Gestionar Cliente | | \*En cualquier momento puede cancelar la operación mediante el botón “Cancelar” | |
| 3. El sistema carga el label de cliente con el cliente Seleccionado. | |  | |
| 4. El actor selecciona la forma de envió y forma de pago desde sus combos.  Selecciona el producto a vender desde la grilla de productos.  Luego ingresa la cantidad deseada a vender en el TextBox de Cantidad y hace click en el botón ADD para agregar el detalle de venta.  Una vez cargado todos los detalles de la venta el actor realiza la venta mediante el Botón “Realizar Venta”. | | 4.1 Puede eliminar mediante el botón DEL el detalle de Venta seleccionada.  4.2 Puede Deshacer el último detalle de venta ingresado mediante el botón “Deshacer” | |
| 5. El sistema Realiza la Venta ingresada. | |  | |

## Casos de usos de Alto Nivel

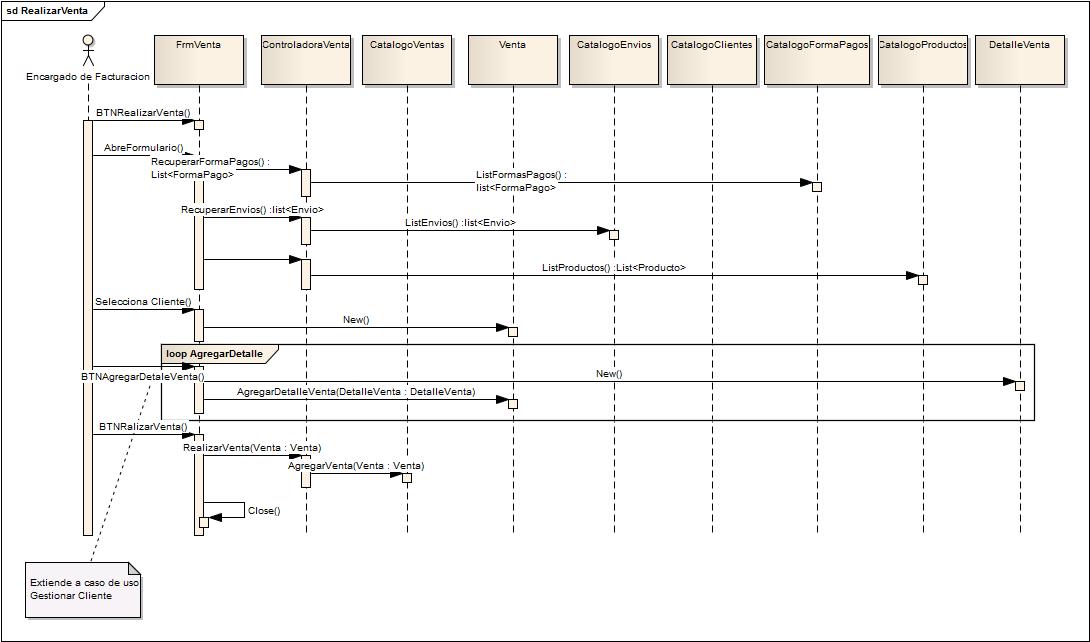
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2021 Versión: 1.0** |
| **Nombre** | CU -19 Gestionar Clientes | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Administrativo | |
| **Descripción** | El actor puede Agregar, Modificar, Seleccionar y Eliminar clientes. El cliente está conformado por un Código, Razón Social, Teléfono, Domicilio, Cuit, Email, Situación Fiscal, Ciudad. | |
| **Pre-Condiciones** | El actor debe poseer permisos para Gestionar Clientes |  |
| **Post-Condiciones** | El actor Gestión Clientes | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de Uso** | | | **Fecha: 02/06/2021 Versión: 1.0** |
| **Nombre** | CU -20 Gestionar Venta | | |
| **Tipo** | Primario Alto nivel | | |
| **Actores** | Administrador | | |
| **Descripción** | El actor podrá Cancelar y Consultar Ventas realizadas las cuales poseen un código, un Cliente, un total, una fecha, Forma de pago, Forma de envió, un Detalle Y Detalles de Ventas el cual está conformado por un Sub Total, Un producto y una cantidad. | | |
| **Pre-Condiciones** | El actor debe poseer permisos para Gestionar Ventas. |  | |
| **Post-Condiciones** | El actor Gestiono una Venta. | | |
| **Autor** | Matías Pena – Marcelo Palonsky | | |
| **Revisor** | - | | |
| **Autorizante** | - | | |

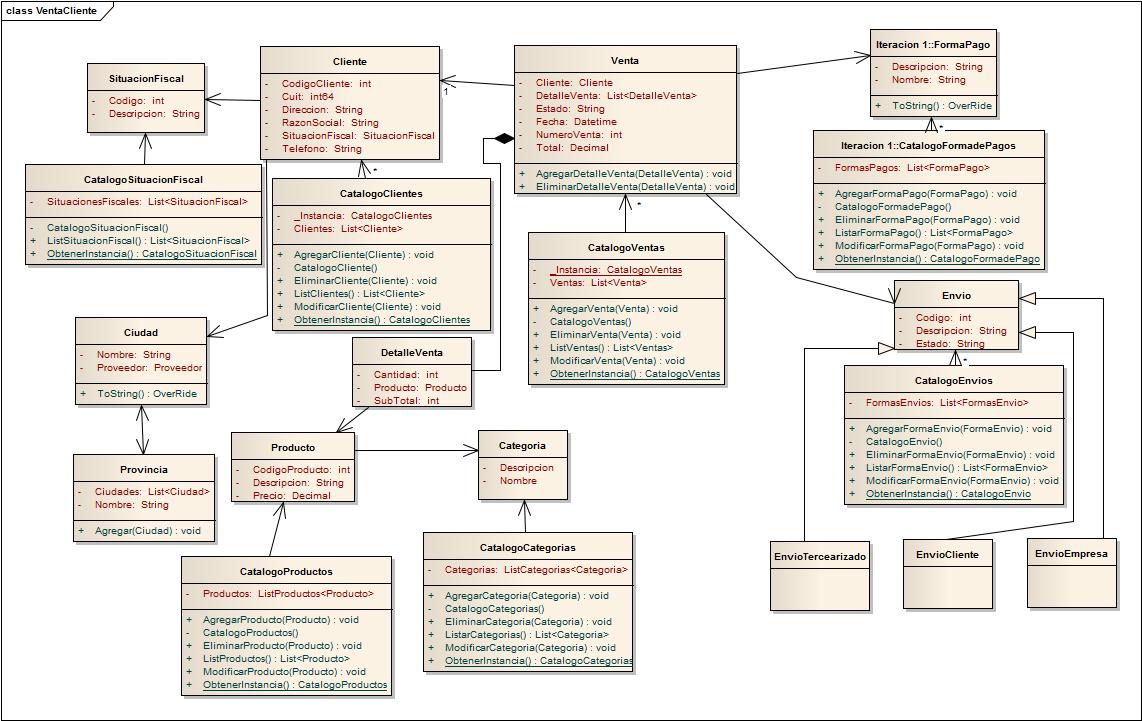
## Diagrama de robustez



## Diagrama de Secuencia



## Diagrama de clases



## Modelo de datos



## Patrón de Diseño

### Patrones de comportamiento - Memento

Sin violar la encapsulación, captura y exteriorizar el estado interno de un objeto de forma que el objeto puede ser restaurado a este estado después.

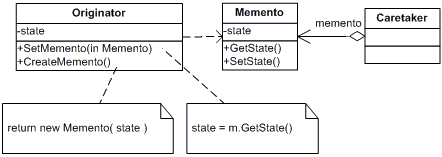
Clasificación del patrónComportamiento

Intención  
Proporcionar una funcionalidad adicional para el usuario.

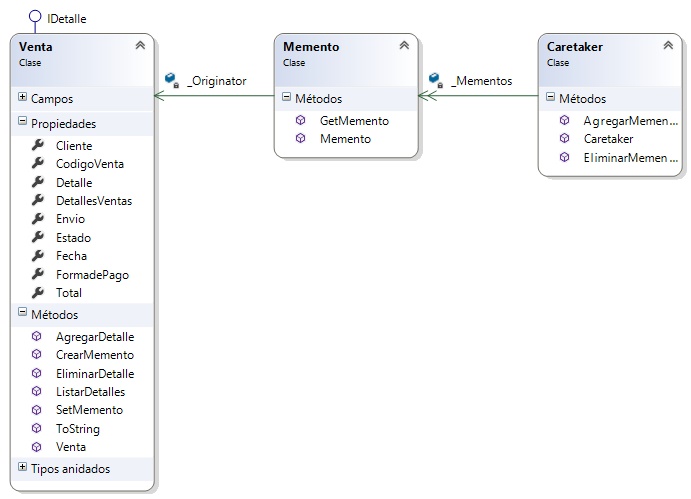
### Motivación.

Permitir volver a un estado anterior que se realiza.

### Estructura del patrón



|  |  |
| --- | --- |
| **Pattern** | Memento Ventas |
| **Based on** | Memento |
| **Because** | Para poder deshacer acciones sobre la venta que se está realizando. |
| **Where** | Originador es Venta.  SetMemento() es SetMemento()  CreateMemento() es CrearMemento()  State es DetalleVenta  Memento es Memento  GetState() es GetMemento()  SetState() es Memento()  CareTaker es CareTaker  Memento es \_Mementos |
| **Comments** | Al realizarse estar realizando una venta, se puede ir volviendo a los estados de detalles anteriores mediante el Botón Deshacer. |



### Implementación

### Clase Venta

public Memento CrearMemento()

{

return new Memento(this);

}

public void SetMemento(Memento oMemento)

{

this.\_DetallesVentas = oMemento.GetMemento().DetallesVentas;

}

### Clase Memento

public class Memento

{

private Venta \_Originator;

public Memento (Venta State)

{

MemoryStream m = new MemoryStream();

BinaryFormatter b = new BinaryFormatter();

b.Serialize(m, State);

m.Position = 0;

\_Originator = (Entidades.Sistema.Venta)b.Deserialize(m);

}

public Venta GetMemento()

{

return \_Originator;

}

}

### Clase CareTaker

public class Caretaker

{

private List<Entidades.Sistema.Memento> \_Mementos;

public Caretaker()

{

\_Mementos = new List<Memento>();

}

public void AgregarMemento(Memento oMemento)

{

\_Mementos.Add(oMemento);

}

public Memento EliminarMemento()

{

if (\_Mementos.Count > 0)

{

Memento memento = \_Mementos[\_Mementos.Count - 1];

\_Mementos.Remove(memento);

return memento;

}

else

{

return null;

}

}

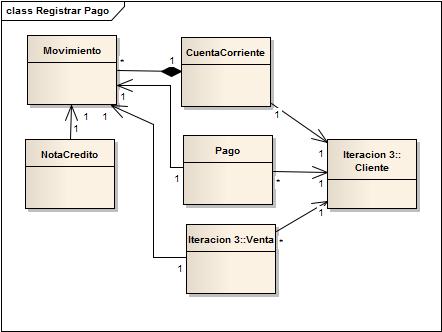
}

# Iteración IV

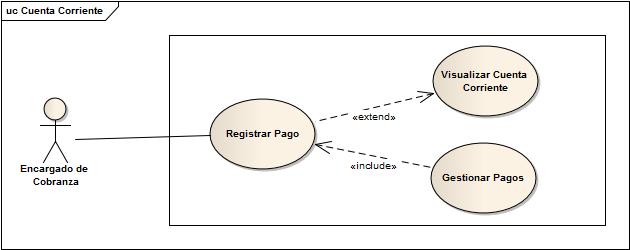
## Listado de Requerimientos

|  |  |  |
| --- | --- | --- |
| **Ref #** | **Descripción** | **Cat** |
| **R1** | **Registrar Pago:** El Pago tiene una Fecha, Importe, Cliente, Detalle y Venta asociada. | Evidente |
| **R2** | El sistema deberá generar un movimiento de Crédito a la cuenta corriente del Cliente al registrarse un pago. | Evidente |
| **R3** | El sistema deberá actualizar la cuenta corriente cuando se efectué un movimiento de tipo crédito a un cliente. | Evidente |
| **R4** | El sistema deberá visualizar únicamente las ventas con estado distinto a “Finalizado”. | Evidente |
| **R5** | **Gestionar Pagos:** Se visualizara todos los pagos registrados en el sistema en una grilla. | Evidente |
| **R5** | El sistema deberá poder Filtrar por únicamente las ventas con estado distinto a “Finalizado”. | Evidente |
| **R6** | **Gestionar Cuentas Corrientes:** El sistema visualizara todos los movimientos generados para un cliente. El movimiento está conformado por un Detalle, Importe, Fecha y Tipo de Movimiento. | Evidente |
| **R7** | El sistema deberá poder filtrar los movimientos por Monto, Fecha y Tipo de Movimiento. | Evidente |

## Modelo de dominio



## Diagramas de casos de uso



## Especificación de caso de uso

|  |  |  |  |
| --- | --- | --- | --- |
| **Caso de uso:** | **Registrar Pago** | | **Versión: 0.1** |
| **ID de caso de uso:** | CU-21-Registrar Pago | | |
| **Tipo:** | Primario Real | | |
| **Interfaz de usuario:** | FrmRealizarPago | | |
| **Creado por:** | Matías Pena – Marcelo Palonsky | | |
| **Fecha de creación:** | 02/06/22 | | |
| **Actor:** | Encargado de Facturación | | |
| **Descripción:** | El actor podrá Realizar Pagos indicando la fecha, importe y Cliente al cual se realizara la acreditación en su Cuenta Corriente. | | |
| **Pre – condiciones:** | El actor deberá estar logueado y con los permisos necesarios para acceder al formulario de Realizar Pago. | | |
| **Post – condiciones:** | El actor realizó un pago a un cliente. | | |
| **CU Extend** | CU-19-VisualizarCuentaCorriente | | |
| **Disparador** | El actor hace click en Realizar Pago | | |
| **Flujo normal** | | **Flujo alternativo** | |
| 1. El actor hace clic en el botón Realizar Pedido del Formulario FrmGestionarPago. | | \*.a.1 En cualquier momento ocurre un error grave.  \*.a.2 El Sistema muestra un cuadro de Plasticos Refine con error. | |
| 2. El Sistema abre el formulario FrmRealizarPago con los campos **Fecha, Importe**, **Cliente y Detalle** obligatorios vacíos a completar, el botón Agregar y Cancelar. | |  | |
| 2. El Actor completa los campos **Fecha, Importe**, **Cliente, Detalle** y hace clic en el botón **Agregar.** | | 2.1 El Sistema muestra un cartel que informa **“Debe completar todos los campos.”,** y el botón **Aceptar.**  2.1.2 El Actor hace clic en el botón **Aceptar**.  2.1.3 El Sistema cierra el cartel y continúa su ejecución. | |
| 3. El sistema Registra el Pago efectuado por el actor y actualiza la cuenta corriente del Cliente seleccionado. | |  | |

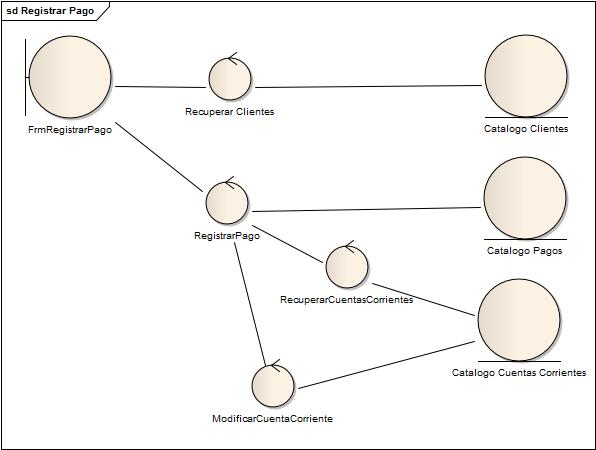
## Casos de usos de Alto Nivel

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -22 Visualizar Cuenta Corriente | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Encargado de Facturación. | |
| **Descripción** | El Actor seleccionara el cliente deseado para visualizar el estado de su cuenta corriente, el cual tendrá los movimientos de Débito o Créditos correspondientes y su saldo. | |
| **Pre-Condiciones** | El Actor debe estar logueado y con los permisos necesarios. |  |
| **Post-Condiciones** | El Actor Visualizo la cuenta corriente del Cliente seleccionado. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

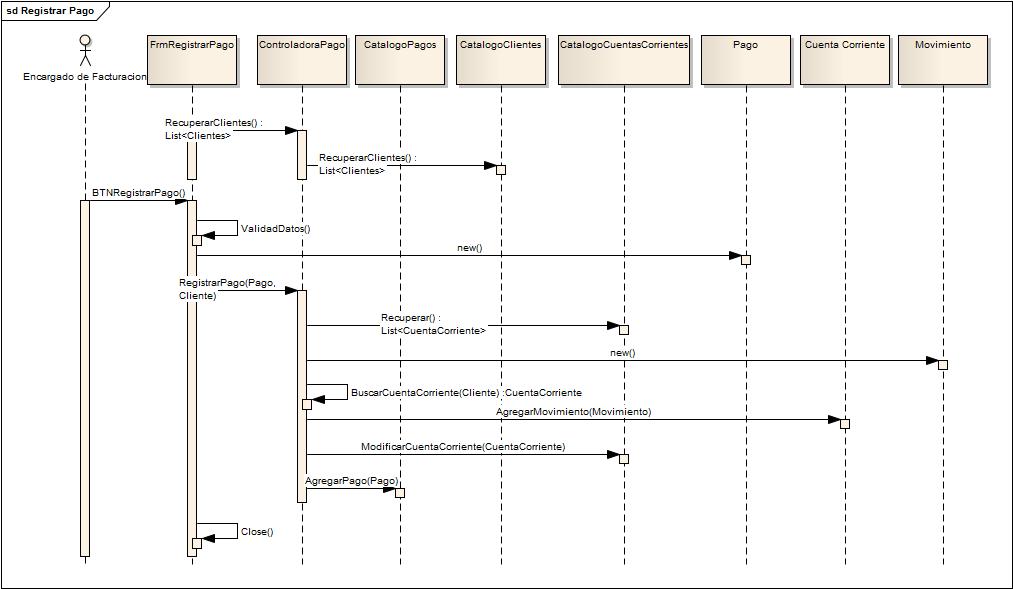
|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -23 Registrar Pago | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Encargado de Facturación. | |
| **Descripción** | El Actor Registra la información del Pago los cuales está conformado por la Fecha, Importe, Cliente y Detalle. Una vez registrado el pago se debe actualizar la cuenta corriente del cliente acreditándole el importe del pago efectuado. | |
| **Pre-Condiciones** | El Actor debe estar logueado y con los permisos necesarios. |  |
| **Post-Condiciones** | El Actor Registró el pago de un Cliente. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

|  |  |  |
| --- | --- | --- |
| **Caso de Uso** | | **Fecha: 02/06/2022 Versión: 1.0** |
| **Nombre** | CU -24 Gestionar Pago | |
| **Tipo** | Primario Alto nivel | |
| **Actores** | Encargado de Facturación. | |
| **Descripción** | El Actor podrá Visualizar los Pagos registrados mediante el filtro deseado, ya sea por Fecha, Cliente, Detalle o Producto. | |
| **Pre-Condiciones** | El Actor debe estar logueado y con los permisos necesarios. |  |
| **Post-Condiciones** | El Actor Visualizó los pagos registrados. | |
| **Autor** | Matías Pena – Marcelo Palonsky | |
| **Revisor** | - | |
| **Autorizante** | - | |

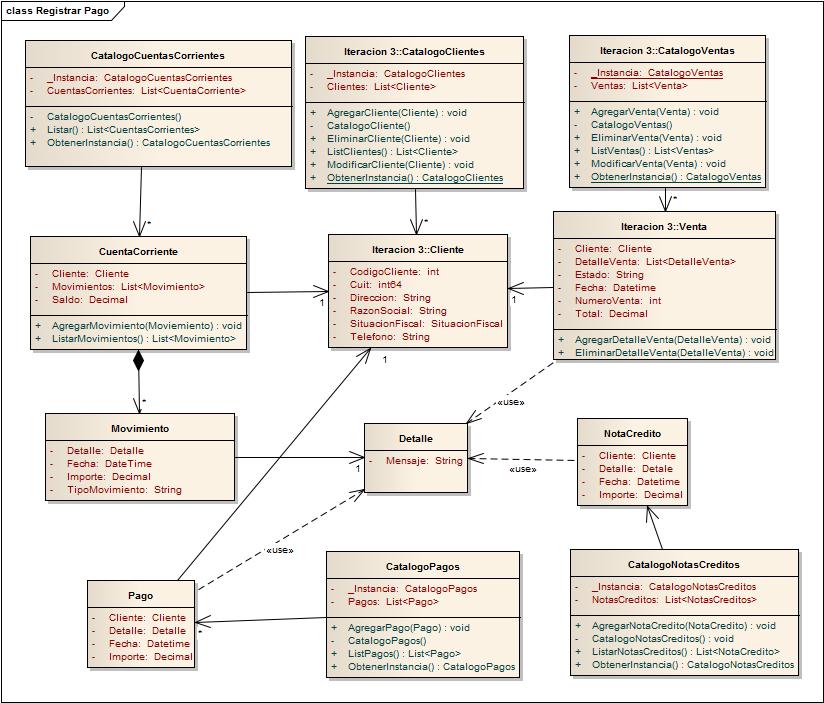
## Diagrama de robustez



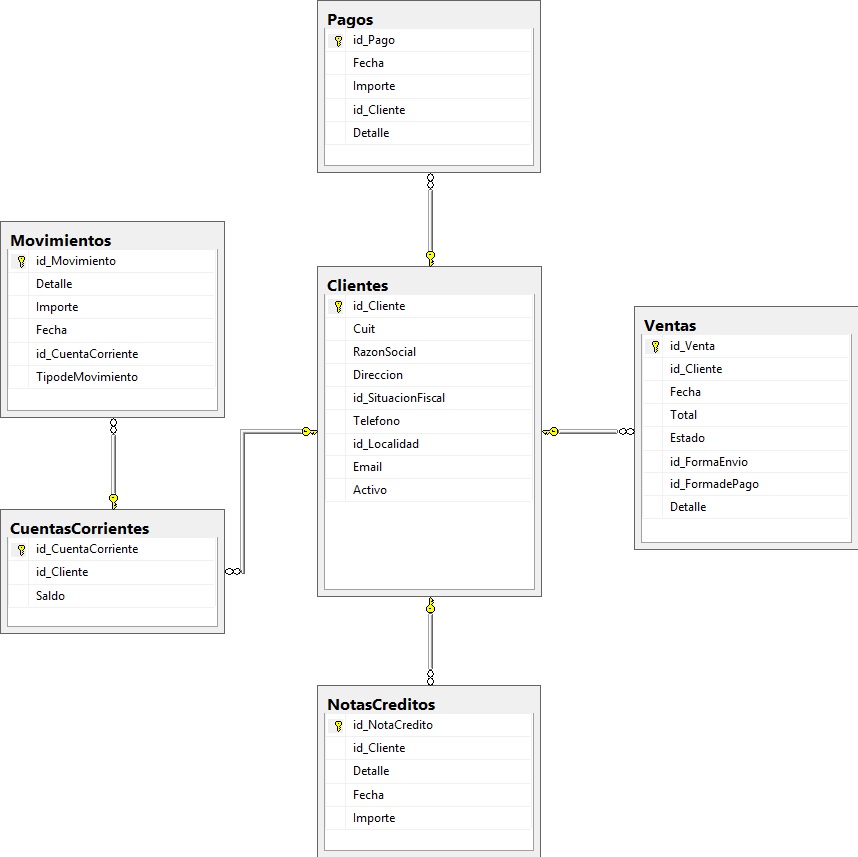
## Diagrama de Secuencia



## Diagrama de clases



## Modelo de datos



## Patrón de Diseño

### Patrón Comportamiento - Estrategia

Definir una familia de algoritmos, encapsular cada uno de ellos, y hacerlos intercambiables. Estrategia permite que el algoritmo de variar independientemente de los clientes que lo utilizan.

Clasificación del patrónComportamiento

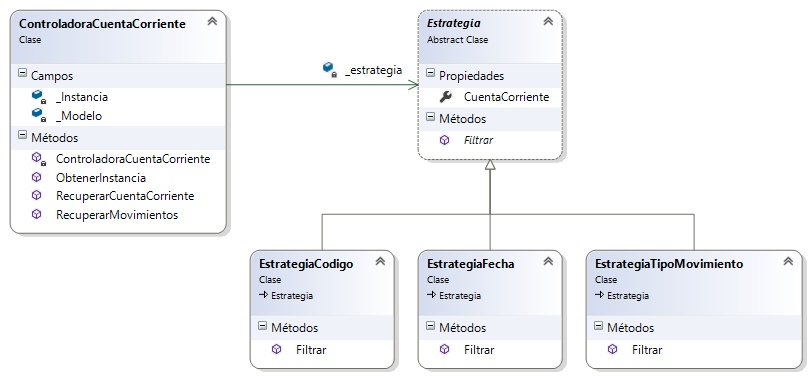
Intención  
Proporcionar al usuario varias maneras sencillas de filtrado.

### Estructura del patrón



|  |  |
| --- | --- |
| **Pattern** | Estrategia |
| **Based on** | Strategy |
| **Because** | Definir una estrategia para el filtrado de cuenta corriente. |
| **Where** | Strategy es Estrategia   * AlgorithmInterface() es Filtrar()   Concrete StrategyA, StrategyB , StrategyC son EstrategiaCodigo, EstategiaFecha y EstrategiaTipoMovimiento   * AlgorithmInterface() es Filtrar()   Context es ControladoraCuentaCorriente   * Strategy es \_estrategia * ContextInterface() es RecuperarMovimientos() |

### Implementación:



# ANEXOS

## Procesos de Desarrollo de Software

Aparecen con el objetivo de ordenar el caos del desarrollo del software.

¿Qué son? Se definen como un conjunto de actividades, tareas, acciones, fundamentos y productos de trabajo que se requieren para desarrollar un software. Proporcionan estabilidad, control y organización en el proceso de software. Llenan un marco de trabajo con un conjunto de tareas explícitas para las acciones de la Ingeniería del Software.

### Modelos Prescriptivos

Llena el marco de trabajo con conjuntos de tareas explícitas para los acciones de la Ingeniería del Software. Se llaman así por que prescriben un conjunto de elementos del proceso: actividades del marco de trabajo, acciones de Ingeniería del Software, productos de trabajo, aseguramiento de la calidad y mecanismos del control del cambio para cada proyecto. Cada modelo de proceso prescribe también un flujo de trabajo

### El modelo de Cascada

Sugiere un enfoque sistemático, secuencial, hacia el desarrollo de software. Inicia con la especificación de requerimientos del cliente y continúa con la planeación, modelado, construcción y despliegue.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Se tiene todo bien organizado y no se mezclan las fases. * Es perfecto para proyectos que son rígidos, y además donde se especifiquen muy bien los requerimientos y se conozca muy bien la herramienta a utilizar | * Los requerimientos se detectan en el principio, una vez avanzado el proyecto, es difícil volver hacia atrás para incorporar nuevos. * Los cambios confunden al equipo. * Es difícil para el cliente especificar todos los requerimientos en la primera etapa. * El cliente debe tener paciencia, un prototipo se obtiene cuando el proyecto está muy avanzado. * Produce estado de bloqueo. |

### Modelos de Proceso Incremental

Quizás haya una imperiosa necesidad de proporcionar de manera rápida un conjunto limitado de funcionalidades para el usuario y luego refinar y expandir en base a las conclusiones.

El modelo incremental entrega una serie de lanzamientos, llamados incrementos, que proporcionan en forma progresiva más funcionalidad para los clientes a medida que se entrega cada uno de los incrementos. Los primeros incrementos son versiones “incompletas” del producto final, pero proporciona al usuario la funcionalidad que necesita y una plataforma para evaluarlo.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Combina elementos del modelo en cascada en forma iterativa. * Tranquiliza al cliente la entrega de incrementos. * Se va modificando en base a las conclusiones arrojadas por la experiencia que tuvo el usuario. | * El uso de un desarrollo incremental no garantiza por sí solo el éxito de su uso. * Hay costos ocultos en su implementación, ya que se incorporan varias actividades a realizar por el equipo, y hay que saber medir ese impacto para no fracasar en el intento. |

### 

### Modelo DRA (Desarrollo Rápido de Aplicaciones)

El desarrollo rápido de aplicaciones es un modelo de proceso de software incremental que resalta un ciclo de desarrollo corto. Es una adaptación a “Alta velocidad” del modelo en cascada en el que se logra el desarrollo rápido mediante un enfoque des contracción basado en componentes.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Proporciona un sistema a corto plazo de tiempo. | * Requiere de mayor cantidad de RRHH en proyectos grandes. * Si hay falta de compromiso el proyecto fallará. * Es inadecuado cuando el proyecto adopta nuevas tecnologías. |

### Modelo de Proceso Evolutivo

El software evoluciona con el tiempo. Muy probablemente los requisitos de los negocios y productos cambien conforme se realiza el desarrollo, por lo tanto, la ruta lineal que conduce a un producto real, no será real, por lo que se debe presentar una versión limitada y liberar presión competitiva. Se tiene claro un conjunto de requerimientos del producto o sistema esencial.

Los modelos Evolutivos son iterativos, permiten que los ingenieros desarrollen versiones cada vez mas completa del software. A pesar que este modelo tiene incuestionables beneficios posee también algunas dificultades: No establecen la velocidad máxima de la evolución, si éstas suceden demasiado rápido, existe la certidumbre de que el proceso caerá en el caos mientras que si la velocidad es demasiado lenta, la productividad se podría ver afectada. Otra dificultad es que los procesos de software se deben enfocar en la flexibilidad y extensibilidad en vez de la alta calidad.

### Construcción de Prototipos

A menudo un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. El paradigma de construcción de prototipos se inicia con la comunicación. El ingeniero de software y el cliente encuentran y definen los objetivos globales del software.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * El prototipo se ajusta para satisfacer las necesidades del cliente con cada avance. * De manera ideal, el prototipo serviría para identificar los requisitos del software. * Ayuda al ingeniero de sistemas y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos. | * El cliente ve una versión del prototipo sin entender que carece de muchas cosas esenciales en el proceso de software. * Implican un problema para la planeación del proyecto debido al número incierto de ciclos requeridos para construir el producto. * En el apuro de desarrollar un prototipo y que a la vez funcione con rapidez, se puede utilizar un Sistema Operativo incorrecto, un lenguaje de programación inadecuado, un algoritmo equivocado, etc. Con el tiempo uno se familiariza con dichas cosas y estas elecciones se convierten en una parte integral del sistema. |

### 

### Modelo en Espiral

Se puede aplicar a través del ciclo de vida completo de una aplicación. Es evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con aspectos controlados y sistemáticos del modelo en cascada. Proporciona el material para el desarrollo rápido de versiones incrementales del software.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Tiene un enfoque cíclico para el crecimiento incremental, así disminuye el grado de riesgo. * Se crean prototipos y se desarrolla una serie de entregas evolutivas. * El software evoluciona conforme avance el proyecto. * Refleja una forma más verídica del mundo real. * Puede adaptarse y aplicarse a lo largo de la vida del software de computadora. * El desarrollador y el cliente entienden y reaccionan de mejor manera ante los riesgos en cada etapa evolutiva. | * Es difícil convencer a los clientes de utilizar este modelo (en particular cuando hay contrato de por medio). * No se puede establecer un presupuesto fijo. |

### Modelo de Desarrollo Concurrente

Se representa de forma esquemática como una serie de actividades del marco de trabajo, acciones y tareas de la ingeniería del software y sus estados asociados. El modelo de desarrollo concurrente define una serie de eventos que dispararán transiciones de estado a estado para cada una de las actividades.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Proporciona una visión exacta del estado actual del proyecto. * Todas las actividades aparecen de forma concurrente. | * Si no se dan las condiciones previamente señaladas no es aplicable (uso de actividades con estados). |

### 

### Modelos especializados de Proceso

Tienden a aplicarse cuando se ha elegido un enfoque de ingeniería de software definido de una manera muy estrecha

### Desarrollo Basado en Componentes

Proporcionas funcionalidades dirigidas con interfaces bien definidas que permiten que el componente se integre en el software. Este modelo incorpora características del modelo espiral, es evolutivo por naturaleza y exige un enfoque iterativo.

Incorporan los siguientes pasos:

• Los productos basados en componentes se investigan y evalúan para el dominio de la aplicación.

• Se consideran los aspectos de integración.

• Se diseña una arquitectura.

• Se integra a la anterior.

• Pruebas detalladas.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Conduce a la reutilización del software. * El ensamblaje de componentes conduce a una reducción del ciclo de desarrollo, del costo y aumenta la productividad. | * Cuenta con una planificación ardua. |

### El Modelo de Métodos Formales

Conduce a la especificación matemática del software de computadoras. Puede desarrollar, especificar y verificar con notación matemática rigurosa.

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Permite que el ingeniero del software descubra y corrija errores complejos que de otra manera podrían no haberse detectado. * Brindan soluciones rápida para (ambigüedad, estado incompleto, inconsistencia). | * Es muy caro y consume mucho tiempo. * Se requiere una investigación detallada. * Es muy difícil utilizar este mecanismo como lenguaje para la comunicación con los clientes. |

### 

### Desarrollo del Software Orientado a Aspectos

Es un nuevo paradigma de la ingeniería del software que proporciona un proceso y enfoque metodológico para definir, especificar, diseñar y construir aspectos (mecanismos más allá de subrutinas y legados para localizar la expresión de un interés en general).

### Proceso Unificado

Es un intento encaminado a reunir los mejores rasgos y características de modelos de proceso del software. Utiliza los principios de desarrollo ágil. Sugiere un flujo de proceso iterativo-incremental y proporciona el sentido evolutivo esencial.

Las fases del proceso unificado tienen propósitos generales al marco de trabajo, ellos son:

1) Fase de inicio.

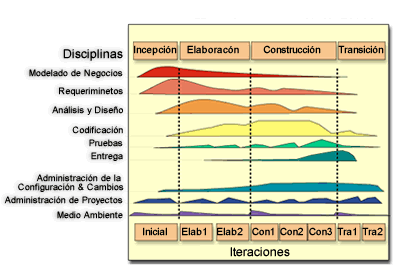
2) Fase de elaboración.

3) Fase de construcción.

4) Fase de transición.

5) Fase de producción.

#### Gráfico del Proceso Unificado

******

### Modelo Iconix

ICONIX es un proceso simplificado en comparación con otros más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

Es una metodología pesada-ligera de Desarrollo del Software que se halla entre RUP (Rational Unified Process) y XP (eXtreme Programming), unifica un conjunto de métodos de orientación a objetos con el objetivo de tener un control estricto sobre todo el ciclo de vida del producto a realizar.

Fue elaborado por Doug Rosenberg y Kendall Scott a partir de una síntesis del proceso unificado de los “tres amigos” Booch, Rumbaugh y Jacobson y que ha dado soporte y conocimiento a la metodología ICONIX desde 1993. Presenta claramente las actividades de cada fase y exhibe una secuencia de pasos que deben ser seguidos.

### Características Principales

ICONIX cuenta con tres características fundamentales:

* Iterativo e Incremental: durante el desarrollo del modelo del dominio y la definición de los casos de uso se producen varias iteraciones. El ciclo de vida incremental consiste en desarrollar por partes el producto de manera que puedas integrarlas funcionalmente. Ciclo de vida Iterativo, en cada ciclo de iteración se revisa y mejora el producto. El desarrollo se organiza en series de mini-proyectos cortos, llamados iteraciones.
* Trazabilidad: Cada paso que se realiza está definido por un requisito, se define la trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos de software producidos.
* Dinámica del UML: Ofrece un uso dinámico del UML porque utiliza algunos diagramas UML, sin exigir la utilización de todos, como en el caso de RUP (Rational Unified Process).

|  |  |
| --- | --- |
| Ventajas | Desventajas |
| * Proceso ágil para obtener un sistema informático. * Dedicada a la construcción de sistemas de gestión de pequeña y mediana complejidad con la participación de los usuarios finales. | * Necesita información rápida y puntual de los requisitos, del diseño y de las estimaciones. * Es una metodología que no debe ser usada en proyectos de larga duración. |

### 

### Seleccionamos el modelo Iconix

Es una metodología que utiliza un modelo de trabajo basado en procesos, lo cual permite una mayor velocidad en el desarrollo del proyecto, con la seguridad y solidez que este permite al utilizarlo de manera adecuada.

ICONIX permite tener proyectos de calidad, en un tiempo corto y que permiten a los usuarios o clientes a estar al pendiente y tener más conocimientos acerca del proyecto que requieren, esto para que el proyecto se valla generando tal y como el cliente lo necesita, esta metodología no se utiliza en proyectos que requieran mucho tiempo

## Estudio de viabilidad

### Viabilidad Legal

No existen impedimentos legales.

### 

### Viabilidad Técnica

El sistema funciona bajo un entorno Windows 7 o superior, en una aplicación de escritorio gestionando base de datos con SQL*.*

Se recomendó instalar dos equipos con el correspondiente sistema operativo y sus respectivas licencias. Y agregar una impresora láser-jet entre las dos oficinas.

### 

### Viabilidad Operativa

Se proveerá una capacitación para los usuarios finales del sistema.

### 

### Viabilidad Económico/Financiera

Costo total del proyecto (desde el relevamiento hasta la implementación)…………$ 13000

Software recomendado: 2 licencias Windows 10………………………………………………......$1100

Instalación: …………………………………………………………………………………………………………..…$ 300

Capacitación para usuarios (2 personas)…………………………………………………………………$1600

## 

## Métricas orientada

### Puntos de casos de uso

El método utiliza los actores y casos de uso relevados para calcular el esfuerzo que significará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, entendidas como una interacción entre el usuario y el sistema, mientras que a los actores se les asigna una complejidad basada en su tipo, es decir, si son interfaces con usuarios u otros sistemas. También se utilizan factores de entorno y de complejidad técnica para ajustar el resultado.

### 

### Método

El método de punto de casos de uso consta de cuatro etapas, en las que se desarrollan los siguientes cálculos:

1. Factor de peso de los actores sin ajustar (UAW)

2. Factor de peso de los casos de uso sin ajustar (UUCW)

3. Puntos de caso de uso ajustados (UCP)

4. Esfuerzo horas-hombre

### 

### Puntos de caso de uso sin ajustar (UUCP)

Al inicio de un proyecto de software, cuando apenas se conocen los casos de uso y sus actores asociados, se puede proyectar una breve descripción de cada caso de uso, en el cual se describe de forma breve la funcionalidad que éste debe brindar.

El UUCP son los puntos de casos de uso sin ajustar, esto nos puede servir para tener una idea un poco más precisa de la dificultad de los casos de uso e interfaces, tomando en cuenta los pesos de los actores (UAW) y los pesos de los casos de uso (UUCW). UUCP = UAW + UUCW

UUCP = 9 + 140= 149

Estas siglas significan:

• UUCP: Puntos de casos de uso sin ajustar.

• UAW: Factor de peso de los actores sin ajustar.

• UUCW: Factor de peso de los casos de uso sin ajustar.

Aplicando el análisis de puntos de función a estos casos de uso, se puede obtener una estimación trivial del tamaño y a partir de ella una estimación del esfuerzo.

### 

### Factor de peso de los actores sin ajustar (UAW)

Consiste en la evaluación de la complejidad de los actores con los que tendrá que interactuar el sistema. Este puntaje se calcula determinando si cada actor es una persona u otro sistema, de la forma en la que este interactúa con el caso de uso, y la cantidad de actores de cada tipo.

|  |  |  |
| --- | --- | --- |
| Tipo de  actor | Descripción | Factor |
| Simple | Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API). | 1 |
| Medio | Otro sistema interactuando a través de un protocolo (ej. TCP/IP) o una persona interactuando a través de una interfaz en modo texto. | 2 |
| Complejo | Una persona que interactúa con el sistema mediante una interfaz gráfica (GUI). | 3 |

|  |  |  |
| --- | --- | --- |
| Tipo de actor | Factor | Cantidad |
| Simple | 1 | 0 |
| Medio | 2 | 0 |
| Complejo | 3 | 3 |
| Total= |  | UAW = (3\*3)= 9 |

Tabla 1: Peso de los actores sin ajustar.

**La fórmula sería: UAW = Sum(cantidadDeUnTipoDeActor\*Factor)**

Para realizar esta operación sería necesario contar cuántos actores de cada tipo existen en el sistema, este representaría el valor cantidadDeUnTipoDeActor en la fórmula y se tiene que multiplicar por el valor que tenga su factor correspondiente, para obtener el resultado por cada tipo de actor. Una vez terminado esto se procede a sumar cada producto para obtener el UAW.

### 

### Factor de peso de los casos de uso sin ajustar (UUCW)

Este punto funciona muy similar al anterior, pero para determinar el nivel de complejidad se puede realizar mediante dos métodos: basado en transacciones o basado en clases de análisis.

Una transacción es un conjunto de actividades atómicas, lo que quiere decir que se ejecutan todas o no se ejecuta ninguna.

• Basado en transacciones: Toma en cuenta el número de transacciones que se pueden realizar en un caso de uso y lo evalúa según la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Tipo de caso de uso | Descripción | Factor |
| Simple | 3 transacciones o menos | 5 |
| Medio | 4 a 7 transacciones | 10 |
| Complejo | Más de 7 transacciones | 15 |

Tabla 2: Peso de las transacciones.

Basado en clases de análisis.

Toma en cuenta el número de clases que tiene un caso de uso y lo evalúa según la siguiente tabla:

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de caso de uso | Descripción | Factor |  |
| Simple | Menos de 5 clases | 5 |  |
| Medio | 5 a 10 clases | 10 |  |
| Complejo | Más de 10 clases | 15 |  |
| Tipo de caso de uso | Factor | Cantidad | Total |
| Simple | 5 | 0 | 0 |
| Medio | 10 | 14 | 140 |
| Complejo | 15 | 0 | 0 |
| TOTAL |  |  | UUCW :140 |

Tabla 3: Peso de las clases de análisis.

Ahora independientemente del camino utilizado para determinar el tipo de caso de uso, la fórmula es la misma y se presenta a continuación: La fórmula sería: UUCW = Sum (CantidadDeUnTipoDeCasoUso\*Factor) Para realizar esta operación se debe contar cuántos casos de uso de cada tipo hay en el sistema y esta cantidad se sustituiría en el campo nombrado como Cantidad De Un Tipo De Caso Uso y se multiplica por el valor que tenga su factor correspondiente, para obtener el resultado por cada tipo de caso de uso. Una vez hecho esto se suma cada producto para obtener el factor de peso de los casos de uso sin ajustar (UUCW).

Esta estimación es bastante imprecisa debido principalmente a la escasa información que se tiene, pero permitirá obtener una idea del esfuerzo necesario para llevar adelante el mismo, y podrá ser refinada a medida que se obtenga más información.

### 

### Puntos de caso de uso ajustados (UCP)

Para esto se utilizan las siglas UCP y se obtiene al multiplicar el UUCP el TCF y el EF quedando la operación de la siguiente forma:

UCP = UUCP x TCF x EF **UCP = 149 x 0,905 x 0,875 = 117,989375** Estas siglas significan:

* UCP: Puntos de casos de uso ajustados.
* UUCP: Puntos de casos de uso sin ajustar.
* TCF: Factores técnicos.
* EF: Factores ambientales.

### 

### Factores de complejidad técnica

Este se compone de 13 puntos que evalúan la complejidad de los módulos del sistema que se desarrolla, cada uno de estos factores tienen un peso definido con los cuales se obtendrá puntos ponderados por cada uno de ellos, según la valoración que se le asigne. Para una mejor comprensión, a continuación, se mostrará una tabla con los ítems:

|  |  |  |
| --- | --- | --- |
| Factor | Descripción | Peso |
| T1 | Sistema distribuido. | 1 |
| T2 | Objetivos de performance o tiempo de respuesta. | 2 |
| T3 | Eficiencia del usuario final. | 1 |
| T4 | Procesamiento interno complejo. | 1 |
| T5 | El código debe ser reutilizable. | 1 |
| T6 | Facilidad de instalación. | 1 |
| T7 | Facilidad de uso. | 1 |
| T8 | Portabilidad. | 2 |
| T9 | Facilidad de cambio. | 0.5 |
| T10 | Concurrencia. | 0.5 |
| T11 | Incluye objetivos especiales de seguridad. | 1 |
| T12 | Provee acceso directo a terceras partes. | 1 |
| T13 | Se requiere facilidades especiales de entrenamiento a usuario. | 1 |

Tabla 4: Peso de los factores de complejidad técnica.

Cada uno de estos puntos se debe evaluar según la siguiente escala:

|  |  |
| --- | --- |
| Descripción | Valor |
| Irrelevante | De 0 a 2. |
| Medio | De 3 a 4. |
| Esencial | 5 |

Tabla 5: Escala de los factores de complejidad técnica.

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Peso | Valor | Total |
| T1 | 2 | 1 | 2 |
| T2 | 1 | 3 | 3 |
| T3 | 1 | 3 | 3 |
| T4 | 1 | 5 | 5 |
| T5 | 1 | 1 | 1 |
| T6 | 0.5 | 2 | 1 |
| T7 | 0.5 | 5 | 2,5 |
| T8 | 2 | 1 | 2 |
| T9 | 1 | 0 | 0 |
| T10 | 1 | 1 | 1 |
| T11 | 1 | 5 | 5 |
| T12 | 1 | 0 | 0 |
| T13 | 1 | 5 | 5 |
| TOTAL |  |  | TFactor :30,5 |

Las fórmulas para este punto son:

* TFactor = Sum (Valor\*Peso)
* TCF = 0.6 + (0.01 \* TFactor)= 0.6 + (0.01 \*30,5)= 0,905

Para realizar este cálculo, se debe evaluar cada factor, asignándole un valor como se menciona anteriormente, después se multiplican y se suma cada producto para obtener el TFactor. Luego, se debe seguir la segunda fórmula multiplicando el TFactor por 0.01 y sumar el resultado a 0.6, esto nos va a dar el TCF.

### 

### Factores ambientales

Los factores sobre los cuales se realiza la evaluación son 8 puntos, que están relacionados con las habilidades y experiencia del grupo de personas involucradas con el desarrollo del proyecto. Estos factores se muestran a continuación:

|  |  |  |
| --- | --- | --- |
| Factor | Descripción | Peso |
| E1 | Familiaridad con el modelo de proyecto utilizado. | 1.5 |
| E2 | Experiencia en la aplicación. | 0.5 |
| E3 | Experiencia en orientación a objetos. | 1 |
| E4 | Capacidad del analista líder. | 0.5 |
| E5 | Motivación. | 1 |
| E6 | Estabilidad de los requerimientos | 2 |
| E7 | Personal part-time | -1 |
| E8 | Dificultad del lenguaje de programación | -1 |

Tabla 6: Peso de los factores ambientales.

|  |  |  |  |
| --- | --- | --- | --- |
| Factor | Peso | Valor | Total |
| E1 | 1,5 | 0 | 0 |
| E2 | 0,5 | 0 | 0 |
| E3 | 1 | 5 | 5 |
| E4 | 0,5 | 5 | 2,5 |
| E5 | 1 | 5 | 5 |
| E6 | 2 | 5 | 10 |
| E7 | -1 | 0 | 0 |
| E8 | -1 | 0 | 0 |
| TOTAL |  |  | EFactor : 17,5 |

Cada uno de estos factores se debe calificar con un valor de 0 a 5.

Las fórmulas para este punto son:

EFactor = Sum(Valor \* Peso)

**EF = 1.4 + (-0.03 \* EFactor)= 1.4 + (-0.03 \*17,5)= 0,875**

Para obtener el EFactor se debe sumar todos los productos obtenidos al multiplicar el peso de cada punto por el valor asignado, después se multiplica por -0.03 y se le suma el 1.4. Así, se obtiene el peso de los factores ambientales (EF).

=== Esfuerzo horas-hombre (E)

Este cálculo se realiza con el fin de tener una aproximación del esfuerzo, pensando solo en el desarrollo según las funcionalidades de los casos de uso. Anteriormente, se sugería utilizar 20 horas persona por UCP, pero a través del tiempo se ha ido mejorando. Está basado en los factores ambientales y se calcula de la siguiente manera:

Primero se debe contar la cantidad de factores ambientales del E1 al E6 que tienen una puntuación menor a 3, también contar la cantidad de estos mismos del E7 y E8 que son mayores que 3.

|  |  |
| --- | --- |
| Factor | Filtro |
| De E1 a E6 | Factor < 3 |
| De E7 a E8 | Factor > 3 |

Tabla 7: Factor del esfuerzo horas-persona.

|  |  |
| --- | --- |
| Factor | Cantidad |
| De E1 a E6 | 2 |
| De E7 a E8 | 0 |

Para evaluar el resultado o la cantidad total según la siguiente tabla:

|  |  |
| --- | --- |
| Horas-Persona (CF) | Descripción |
| 20 | Si el valor es<=2 |
| 28 | Si el valor es<=4 |
| 36 | Si el valor es>=5 |

Tabla 8: Cantidad de horas-persona según el valor.

El esfuerzo en horas-persona viene dado por: **E = UCP**

**x CF = 117,989375 x 20= 1974 x 1(personas) = 2359,7875** Estas siglas significan:

E: Esfuerzo estimado en horas-persona.

UCP: Puntos de Casos de Uso ajustados.

CF: Horas-Persona.

Al realizar la multiplicación del UCP por las horas- persona, se consigue un esfuerzo estimado, que representa una parte del total del esfuerzo de todo el proyecto, generalmente un 40%. Este 40% se refiere al esfuerzo total para el desarrollo de las funcionalidades especificadas en los Casos de Uso. En la siguiente tabla se detallan la distribución en porcentaje, para el esfuerzo total en el desarrollo del proyecto:

Salario por cada empleado $9000 x 6,5 meses = $58.500

|  |  |  |
| --- | --- | --- |
| Actividad | Porcentajes | Pesos |
| Análisis | 10% | 14.625 |
| Diseño | 20% | 29.250 |
| Programación | 40% | 58.500 |
| Pruebas | 15% | 21.937 |
| Sobrecarga | 15% | 21.937 |

**Total:** $146249

## Gestión de Riesgos

### Tabla de Riesgos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Riesgo | Categoría | Prob. | Impacto | RSGR |
| Rechazo por falta de entrenamiento | Riegos Negocio | 50% | 2 |  |
| Otra compañía ofrece otro producto similar | Riegos Negocio | 35% | 2 |  |
| Se va del negocio la persona interesada | Riegos Negocio | 10% | 1 |  |
| Cierre de la Inmobiliaria | Riegos Negocio | 20% | 1 |  |
| Perdida de interés | Riegos Negocio | 25% | 1 |  |
| Miembro clave del proyecto renuncia | Riesgo Proyecto | 5% | 1 |  |
| Cambios administrativos | Riesgo Proyecto | 5% | 3 |  |
| Hardware indispensable no disponible | Riesgo Proyecto | 60% | 2 |  |
| Retrasos por cambios de requerimientos | Riesgo Proyecto | 80% | 3 |  |
| Tamaño del proyecto no evaluado | Riesgo Proyecto | 30% | 3 |  |
| Cantidad de defectos retrasan la entrega | Riesgo Proyecto | 40% | 2 |  |
| Cambios de requerimientos originan mala funcionalidad | Riesgo Producto | 15% | 2 |  |
| Nivel de transacciones elevado para el motor de base de datos | Riesgo Producto | 5% | 2 |  |
| Algoritmos inadecuados, baja respuesta. | Riesgo Producto | 25% | 2 |  |

### Valores de Impacto:

1. catastrófico

2. crítico

3. marginal

4. despreciable

### Tabla ordenada según Probabilidad e Impacto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Riesgo | Categoría | Prob. | Impacto | RSGR |
| Retrasos por cambios de requerimientos | Riesgo Proyecto | 80% | 3 | R1 |
| Hardware indispensable no disponible | Riesgo Proyecto | 60% | 2 | R2 |
| Rechazo por falta de entrenamiento | Riegos Producto | 50% | 2  0 | R3 |
| Cantidad de defectos retrasan la entrega | Riesgo Proyecto | 40% | 2 | R4 |
| Otra compañía ofrece otro producto similar | Riegos Negocio | 35% | 2 | R5 |
| Línea de corte | | | | |
| Tamaño del proyecto no evaluado | Riesgo Proyecto | 30% | 3 |  |
| Perdida de interés | Riegos Negocio | 25% | 1 |  |
| Algoritmos inadecuados, baja respuesta. | Riesgo Producto | 25% | 2 |  |
| Cierre de la Inmobiliaria | Riegos Negocio | 20% | 1 |  |
| Cambios de requerimientos originan mala funcionalidad | Riesgo Producto | 15% | 2 |  |
| Se va del negocio la persona interesada | Riegos Negocio | 10% | 1 |  |
| Miembro clave del proyecto renuncia | Riesgo Proyecto | 5% | 1 |  |
| Nivel de transacciones elevado para el motor de base de datos | Riesgo Producto | 5% | 2 |  |
| Cambios administrativos | Riesgo Proyecto | 5% | 3 |  |

### Hoja de información de Riesgo – Plan de reducción, supervisión y gestión del riesgo

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoja de información del Riesgo** | | | |
| **ID Riesgo: R1** | **Fecha: 14/07/21** | **Probabilidad: 80%** | **Impacto: 3** |
| **Descripción** | | | |
| Retrasos por cambios de requerimientos, demoras por no tener bien definido los requerimientos del sistema. | | | |
| **Refinamiento/Contexto** | | | |
| La/las últimas reuniones previas al inicio del proyecto se cambiaron algunos requerimientos obligando a demorar la implementación. Estos cambios del cliente obligan a pensar posibles cambios futuros | | | |
| **Reducción/supervisión** | | | |
| Preparar minuta con requerimientos y firmar por ambas partes acordando el cumplimiento de los mismos. Informar al cliente de que cada cambio tiene un costo adicional. | | | |
| **Gestión/plan de contingencia/disparador** | | | |
| Preparar código reutilizable y reducir el acoplamiento para minimizar os cambios Recalcular tiempo de entrega y costos. Fecha límite 15/07/21 | | | |
| **Estado actual: En Proceso** | | | |
| **Elaborado: Matías Pena** | | **Asignado a: ---** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoja de información del Riesgo** | | | |
| **ID Riesgo: R2** | **Fecha: 14/04/21** | **Probabilidad: 60%** | **Impacto: 2** |
| **Descripción** | | | |
| Hardware indispensable no disponible | | | |
| **Refinamiento/Contexto** | | | |
| El hardware actual de la empresa no cumple con lo esperado para poder cumplir con el proyecto | | | |
| **Reducción/supervisión** | | | |
| Se prepara listado con los equipos que hay que agregar antes de la instalación para poder comprarlos y para que el cliente se vaya preparando. | | | |
| **Gestión/plan de contingencia/disparador** | | | |
| Se muestran los adelantos del sistema en Notebook hasta la compra e instalación del equipo correspondiente Fecha límite 15/07/16 | | | |
| **Estado actual: En Proceso** | | | |
| **Elaborado: Matías Pena** | | **Asignado a: ---** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoja de información del Riesgo** | | | |
| **ID Riesgo: R3** | **Fecha: 14/07/21** | **Probabilidad: 50%** | **Impacto: 2** |
| **Descripción** | | | |
| Rechazo por falta de entrenamiento | | | |
| **Refinamiento/Contexto** | | | |
| Posible rechazo del cliente hacia el software entregado por falta de entrenamiento y conocimiento. Miedo al cambio. | | | |
| **Reducción/supervisión** | | | |
| Creación de software sencillo, amistoso y fácil de utilizar Se realizará manual de usuario y se entrenara al personal. | | | |
| **Gestión/plan de contingencia/disparador** | | | |
| En cada reunión se muestra al cliente como utilizar los diferentes formularios para que se vaya amigando con el sistema. También se acomodan los formularios según necesidades. Fecha límite 15/07/16 | | | |
| **Estado actual: En Proceso** | | | |
| **Elaborado: Matías Pena** | | **Asignado a: ---** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoja de información del Riesgo** | | | |
| **ID Riesgo: R4** | **Fecha: 14/04/21** | **Probabilidad: 40%** | **Impacto: 2** |
| **Descripción** | | | |
| Cantidad de defectos retrasan la entrega | | | |
| **Refinamiento/Contexto** | | | |
| Posible demora en la entrega del sistema por errores en la codificación. | | | |
| **Reducción/supervisión** | | | |
| Armar el sistema por módulos para poder corregir cualquier error antes de cada presentación para llegar al día de la entrega con un sistema sin errores | | | |
| **Gestión/plan de contingencia/disparador** | | | |
| Programación, prueba y presentación por módulos. Fecha límite 15/07/21 | | | |
| **Estado actual: En Proceso** | | | |
| **Elaborado: Matías Pena** | | **Asignado a: ---** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Hoja de información del Riesgo** | | | |
| **ID Riesgo: R5** | **Fecha: 14/04/21** | **Probabilidad: 40%** | **Impacto: 2** |
| **Descripción** | | | |
| Otra compañía ofrece otro producto similar | | | |
| **Refinamiento/Contexto** | | | |
| Posibilidad de otras ofertas y calidad de diseño en las cuales el cliente puede perder el interés por el sistema presentado | | | |
| **Reducción/supervisión** | | | |
| Mantener motivado al cliente con reuniones y presentaciones del sistema ofreciéndole el mejor servicio posible. | | | |
| **Gestión/plan de contingencia/disparador** | | | |
| Llamadas periódicas para dar feedback sobre los avances del sistema. Proponer mejoras interesantes. Fecha límite 15/07/16 | | | |
| **Estado actual: En Proceso** | | | |
| **Elaborado: Matías Pena** | | **Asignado a: ---** | |

### Informe Final

Riesgos controlables: Asignación de recursos, herramientas e información necesaria, así como mecanismos de seguimiento y control.

Riesgos No controlables: Se continuará con los procedimientos que permitan obtener información puntual sobre los mismos. Estos procedimientos proporcionan señales de aviso de manera que se puedan definir e implantar planes de contingencia.

## Testing - Pruebas

### Introducción

Existen dos maneras de probar cualquier producto construido. Se explican brevemente las pruebas de “caja negra y caja blanca”:

1. Caja Negra. Si se conoce la función específica para la que se diseñó el producto, se aplican pruebas que demuestren que cada función es plenamente operacional, mientras se buscan los errores de cada función. Estas pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software.
2. Caja Blanca. Si se conoce el funcionamiento interno del producto, se aplican pruebas para asegurarse de que todas las piezas encajan, es decir que las operaciones internas se realizan de acuerdo con las especificaciones y que se han probado todos los componentes internos de manera adecuada. Estas pruebas de caja blanca del software se basan en un examen cercano al detalle procedimental. Se prueban las rutas lógicas del software y la colaboración entre componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos. Al usar este método el ingeniero de software podrá derivar casos de prueba que: Garanticen que todas las rutas independientes se han ejercitado por lo menos una vez; Ejerciten los lados verdaderos y falsos de todas las decisiones lógicas; Ejecuten todos los bucles en sus límites y dentro de ellos; Ejerciten estructuras de datos internos para asegurar su validez.

### Testing Caja Blanca

Se aplica el método de prueba del camino básico.

public object IniciarSesion(string nombreUsuario, string claveUsuario)

## 1 {

Entidades.Seguridad.Usuario oUsuario ;

2 try

{

oUsuario = (Entidades.Seguridad.Usuario)Modelo.Seguridad.Facade.Buscar(nombreUsuario,"Usuarios");

3 if (oUsuario != null)

{

4 if (oUsuario.Estado)

{

string ClaveEncriptada = Servicios.Encriptacion.EncriptarPassword(claveUsuario);

5 if (oUsuario.Clave == ClaveEncriptada)

{

Entidades.Seguridad.Auditoria oAuditoria = new Entidades.Seguridad.Auditoria();

oAuditoria.Usuario = oUsuario.NombreUsuario;

oAuditoria.Fecha = DateTime.Now;

oAuditoria.Equipo = Environment.MachineName;

oAuditoria.Operacion = "Login"; Modelo.Seguridad.CatAuditoria.ObtenerInstancia().AgregarAuditoria(oAuditoria);

6 return oUsuario;

}

7 else

{

return "Clave erronea";

}

}

8 else return "Usuario deshabilitado";

}

}

9 catch (Exception ex)

Servicios.EventManager.RegistarErrores(ex);

}

return "Usuario Inexistente";

}

### Diagrama de Flujo

****

**** 

**** 

****



### Determinación del conjunto básico de rutas

V(G)= 4 rutas

* Ruta 1: 1- 2-9-10
* Ruta 2: 1-2-3-8-10
* Ruta 3: 1-2-4-5-7-10
* Ruta 4: 1-2-4-5-6-10

Ruta 1: Se prueba ingresando un usuario que no exista en la Base de Datos.

Esto sucede cuando en el campo “Usuario” se ingresan datos que no corresponden a un usuario registrado en el sistema, en el campo “contraseña” se ingresan datos y hace clic en aceptar. El sistema informa "Usuario Inexistente."

Ruta 2: Cuando el usuario existe en la base de datos del programa, pero su estado Habilitado es *False.*

Esto sucede cuando en el campo “Usuario” y “Contraseña” se ingresan datos que corresponden a un usuario registrado en el sistema, pero se encuentra inhabilitado para operar en el sistema.

El sistema informa "Usuario Deshabilitado."

Ruta 3: Cuando el usuario existe*,* pero su clave es incorrecta.

Esto sucede cuando en el campo “Usuario” se ingresa un dato que corresponde a un usuario registrado en el sistema pero su clave no coincide.

El sistema informa "Clave Errónea.".

Ruta 4: Cuando existe el usuario, su estado Habilitado es *True* y su clave es correcta.

Esto sucede cuando en el campo “Usuario” y “Contraseña” se ingresan datos que corresponden a un usuario registrado en el sistema, y se encuentra habilitado para operar en el sistema.

Resultados de la prueba:

Los resultados obtenidos en la ejecución de los casos de prueba fueron los esperados, por lo cual, la prueba ha sido exitosa. No hay que realizar correcciones.

### Testing Caja Negra

Utilizaremos el método de partición equivalente dentro del formulario Iniciar Sesión.

El formulario Iniciar Sesióncuenta con dos condiciones de entrada:

* Nombre de Usuario
* Clave

Datos correctos

1. Ingreso de nombre de usuario alfanumérico.
2. Ingreso de clave alfanumérica.

Datos incorrectos

1. Ingreso Nombre de Usuario en blanco

Respuesta *"Debe ingresar el usuario"*

1. Ingreso Nombre de Usuario no existente en la base de datos

Respuesta  *"El Usuario es incorrecto."*

1. Ingreso Clave en blanco

Respuesta *"Debe ingresar la clave."*

1. Ingreso Clave que no coincide con la del usuario ingresado

Respuesta: *“Clave Erronea."*

1. Ingreso Nombre de Usuario y Clave de un usuario que no está habilitado para ingresar al sistema

Respuesta: *“Usuario Deshabilitado”.*

### Resultados de la prueba:

Los resultados obtenidos en la ejecución de los casos de prueba fueron los esperados, por lo cual, la prueba ha sido exitosa. No hay que realizar correcciones.

### Análisis de valores limites

Valor máximo de alfanuméricos para el Nombre de Usuario es igual a 15 y mínimo 1.

### Valores correctos

1. Se ingresa Nombre de Usuario = alfanumérico con una longitud de 15.
2. Se ingresa Nombre de Usuario = alfanumérico con una longitud de 1.

### Valores incorrectos

1. Se ingresa Nombre de Usuario = alfanumérico con una longitud mayor a 15.

Respuesta *“El nombre de usuario no debe tener más de 15 caracteres”.*

1. Se ingresa Nombre de Usuario = alfanumérico con una longitud de 0.

Respuesta *“Por favor, complete todos los campos”.*

#### Resultados de la prueba:

Los resultados obtenidos en la ejecución de los casos de prueba fueron los esperados, por lo cual, la prueba ha sido exitosa. No hay que realizar correcciones.

### Conclusión

Probar es buscar las fallas a un programa, la función del tester es tratar de romper el programa, es la forma más efectiva para estar seguros de que nuestro programa funciona correctamente.

Un plan de pruebas deberá cumplir con ciertos puntos, en primer lugar se deberá tener en claro que tipo de pruebas se van a aplicar y su correcto orden y diseño.  
Las pruebas informales donde el resultado se aprecia muy sutilmente no deberán ser parte del plan de pruebas.

En síntesis, de las pruebas realizadas anteriormente se corroboró el correcto funcionamiento del sistema habiendo cumplido con lo requerido.

## Política de Back Up

### Backup

Las copias de seguridad o Backups son una manera de proteger los datos de tal manera que permiten la recuperación de la misma en el caso de que ocurra una pérdida de la información, ya sea debido a desastres naturales, fallos de discos, espionaje, errores del sistema operativo, etc.

Cada uno de los aspectos relacionados con las copias de seguridad deben estar especificados en lo que se conoce como **Política de Backup.**

### Política de Backup

Se trata de un documento donde se tienen en cuenta, entre otros aspectos:

* El ámbito de la aplicación, tipos de datos que van a incluir en el Backup. La especificación de los tipos de datos puede hacer hincapié en los siguientes apartados: si se trata de datos de usuarios, copias del sistema operativo, copia de aplicaciones, etc.
* La periodicidad de las copias que se van a realizar.
* La elección del soporte para el back up.
* Los métodos empleados: Incremental, Diferencial, Completo.
* La ubicación final de las copias.
* El/los Responsable/s de hacer las copias.

### Tipos de Backups

Las aplicaciones que se encargan de realizar copias de seguridad ofrecen varios tipos de operaciones de Backup. Los tipos de Backups más comunes son: el Backup completo, el

incremental y el diferencial. Existen otros tipos de Backups como: el Backup completo sintético, la copia en espejo, el Backup incremental inverso y la protección de datos continua (CDP).

* **Backups Completos:** El tipo de Backup más básico y completo es justamente el Completo. Este tipo de Backup se encarga de copiar la totalidad de los datos en otro juego de soportes, que pueden consistir en discos rígidos, discos extraíbles, discos externos, CDs o DVDs. La ventaja principal de este tipo de Backup es que cada operación se obtiene la totalidad de los datos en único juego de soportes. Esto permite restaurar los datos en un tiempo mínimo, lo cual se mide en términos de objetivo de tiempo de recuperación (RTO).

La desventaja del Backup completo es que lleva más tiempo realizarlo que los otros tipos de Backups, se multiplica por un factor de 10 o más, y a su vez requiere de mayor espacio de almacenamiento.

* **Backups Incrementales:** Éste tipo de Backup se encarga de copiar los datos que han variado desde la última operación de Backup de cualquier tipo. Se suele utilizar la fecha y la hora de modificación estampada en los archivos, comparándola con la fecha y la hora del último Backup.
  + Debido a que los Backups Incrementales solo copia los datos a partir del último Backup de cualquier tipo, se puede ejecutar tantas veces como se desee, ya que sólo guarda los cambios más recientes. La ventaja de éste tipo de Backups es que copia una menor cantidad de datos que un Backup completo. Es por ellos, que esas operaciones se realizan más rápido y exigen menos espacio de almacenamiento.
* **Backups Diferenciales:** Son similares a los Backups incrementales, debido que la primera vez que se lleve a cabo, copiará todos los datos que hayan cambiado desde el Backup anterior. Sin embargo, cada vez que se vuelva a ejecutar, seguirá copiando todos los datos que hayan cambiado desde el anterior completo. Debido a esto, en las operaciones subsiguientes almacenará más datos que un Backup incremental, aunque muchos menos que uno completo. Además, la ejecución de los Backups diferenciales requieren más espacio de almacenamiento y tiempo que los incrementales, pero a su vez menos que los completos.

## Arquitectura

Para Plásticos Refine se utiliza una arquitectura de 6 capas, con el fin de dividir en múltiples capas el sistema para agilizar y facilitar el mantenimiento de la aplicación. La arquitectura utilizada separa las lógicas del negocio de las vistas, obteniendo facilidad para la realización de pruebas unitarias de los componentes (Gestiones individuales), con un grado elevado de reutilización de código y módulos, facilitando el mantenimiento futuro del sistema.

Las capas son:

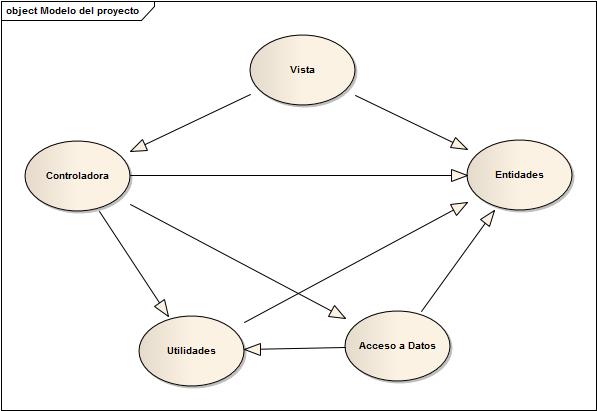
**Vista**: En esta capa se encuentran los formularios que utilizarán los usuarios. En esta capa se ubicaran las validaciones correspondientes al flujo de información. Es la encargada de realizar las interacciones entre el actor y el sistema.

**Utilidades**: Se encuentran las clases con métodos que servirán para realizar tareas comunes.

**Acceso a Datos**: Se ubican las colecciones de entidades, actualizadas. En esta capa se realizan persistencias de objetos. También se encargada de impactar la información del sistema en la base de datos. Es la intermediaria entre todo el sistema y la Base de datos utilizada.

**Entidades**: Se ubican las clases junto con los atributos y métodos definidos.

**Controladora**: Es la capa en la que se encuentran la mayoría de las reglas de negocio, y donde se instancian y guardan datos en los objetos.



Como conclusión a exponer sobre la elección de la plataforma tecnológica empleada, se evaluaron tecnologías diversas aplicables con lo cual, un punto a destacar es el conocimiento previo que poseen los recursos a la hora de desarrollar el sistema informático.

Por ello es que se opta por la elección de trabajar con C# y ASP.net de Microsoft que es con quien ha trabajo el grupo de desarrollo y si se tiene que evaluar la curva de aprendizaje del mismo en general, es ampliamente superior trabajar con la tecnología conocida por ellos.

La misma situación tenemos con el entorno de Base de datos y arquitectura de desarrollo de software.

Inclusive se realiza un esfuerzo mayor ampliando la arquitectura de desarrollo de MVC para tener una mayor visibilidad de los procesos del negocio, como persistir los datos e Interfaz de usuario.

Estimación de horas

Trazabilidad

Cambiar De diseño arquitectura

Requerimientos de seguridad separar

Manual de Usuario