|  |  |  |
| --- | --- | --- |
| **Sistema de información parar la gestión de ventas de productos.** | **CASO DE ESTUDIO** | |
| Caso de estudio aplicando como marco de trabajo la metodología de Programación Extrema el Framework SCRUM. Y UML para los artefactos del software. | | Por. Msc. Ing. Ernesto Soto Roca |

ÍNDICE GENERAL

[CAPÍTULO I: ASPECTOS METODOLÓGICOS 8](#_Toc386526277)

[1. INTRODUCCIÓN. 9](#_Toc386526278)

[1.1. PLANTEAMIENTO DEL PROBLEMA. 9](#_Toc386526279)

[1.1.1. Situación problemática. 9](#_Toc386526280)

[1.1.2. Situación deseada. 9](#_Toc386526281)

[1.1.3. Delimitación de contenido. 9](#_Toc386526282)

[1.1.3.1. Pedidos de los productos y proveedores. 9](#_Toc386526283)

[1.1.3.2. De los precios de cado producto. 9](#_Toc386526284)

[1.1.3.3. Observaciones. 10](#_Toc386526285)

[1.2. ACTORES. 10](#_Toc386526286)

[2. METODOLOGÍA. 10](#_Toc386526287)

[2.1. PROCESO METODOLÓGICO. 10](#_Toc386526288)

[2.1.1. Fase de exploración. 11](#_Toc386526289)

[2.1.2. Fase de planificación. 11](#_Toc386526290)

[2.1.3. Fase de iteraciones. 11](#_Toc386526291)

[2.1.4. Fase de puesta en producción. 12](#_Toc386526292)

[2.1.1. Planificación temporal. 13](#_Toc386526293)

[2.2. SCRUM. 14](#_Toc386526294)

[2.2.1. Procesos. 14](#_Toc386526295)

[2.2.2. Roles y Responsabilidades. 16](#_Toc386526296)

[2.2.3. Prácticas. 16](#_Toc386526297)

[2.3. XTREME PROGRAMMING. 17](#_Toc386526298)

[2.3.1. Procesos. 17](#_Toc386526299)

[2.3.2. Roles y responsabilidades. 19](#_Toc386526300)

[2.3.3. Prácticas. 20](#_Toc386526301)

[2.4. COMBINACIÓN DE XTREME PROGRAMMING Y SCRUM. 22](#_Toc386526302)

[2.5. MÉTODOS Y TÉCNICAS DE SCRUM. 22](#_Toc386526303)

[2.6. MÉTODOS Y TÉCNICAS DE XTREME PROGRAMMING. 23](#_Toc386526304)

[CAPÍTULO II: REQUERIMIENTOS 24](#_Toc386526305)

[3. REQUERIMIENTOS DE SOFTWARE 25](#_Toc386526306)

[3.1. REQUERIMIENTOS FUNCIONALES (PRODUCT BACKLOG) 25](#_Toc386526307)

[3.2. REQUERIMIENTOS NO FUNCIONALES. 30](#_Toc386526308)

[3.2.1. Tiempo de aprendizaje. 30](#_Toc386526309)

[3.2.2. Identificación del usuario propio de la aplicación. 30](#_Toc386526310)

[3.2.2.1. Contraseña de usuario. 30](#_Toc386526311)

[3.2.3. Confiabilidad. 30](#_Toc386526312)

[3.2.3.1. Tiempo de disponibilidad del sistema. 30](#_Toc386526313)

[3.2.3.2. Tiempo fuera de servicio. 31](#_Toc386526314)

[3.2.4. Performance. 31](#_Toc386526315)

[3.2.4.1. Acceso de clientes en línea. 31](#_Toc386526316)

[3.2.4.2. Tiempo de respuesta. 31](#_Toc386526317)

[3.2.4.3. Cantidad de atención a usuarios. 31](#_Toc386526318)

[3.2.5. Soportabilidad o Facilidad de Mantenimiento. 31](#_Toc386526319)

[3.2.5.1. Actualización transparente al usuario. 31](#_Toc386526320)

[3.2.5.2. Estándares de codificación. 32](#_Toc386526321)

[3.2.6. Restricciones de Diseño. 32](#_Toc386526322)

[3.2.6.1. Estándares de diseño. 32](#_Toc386526323)

[3.2.1. Estándares de arquitectura. 32](#_Toc386526324)

[3.2.2. Motor de base de datos. 32](#_Toc386526325)

[3.2.3. Cliente del navegador. 32](#_Toc386526326)

[3.2.4. Servidor Web. 32](#_Toc386526327)

[3.2.4.1. Lenguaje de programación. 33](#_Toc386526328)

[3.2.5. Requerimientos de documentación, ayuda en línea y manuales, asistencia técnica. 33](#_Toc386526329)

[3.2.5.1. Ayuda en línea 33](#_Toc386526330)

[3.2.6. Interfaces. 33](#_Toc386526331)

[3.2.6.1. Interfaz de usuario. 33](#_Toc386526332)

[3.2.6.2. Interfaz de Hardware. 33](#_Toc386526333)

[3.2.6.3. Interfaz de Comunicaciones. 34](#_Toc386526334)

[3.2.6.4. Interfaces de Software. 34](#_Toc386526335)

[CAPÍTULO III: ANÁLISIS 35](#_Toc386526336)

[4. ANÁLISIS. 36](#_Toc386526337)

[4.1. ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE. 36](#_Toc386526338)

[4.1.1. Historias de usuario. 36](#_Toc386526339)

[4.1.1.1. Historia de usuario para registro de un pedido de venta. 37](#_Toc386526340)

[4.1.1.1. Historia de usuario para registro de un proveedor. 38](#_Toc386526341)

[4.1.1.1. Historia de usuario para registro de un artículo. 39](#_Toc386526342)

[CAPÍTULO IV: DISEÑO 40](#_Toc386526343)

[5. DISEÑO. 41](#_Toc386526344)

[5.1. ARQUITECTURA DE LA SOLUCIÓN. 41](#_Toc386526345)

[5.1.1. Vista. 41](#_Toc386526346)

[5.1.2. Modelo. 42](#_Toc386526347)

[5.1.3. Controlador. 42](#_Toc386526348)

[5.1.4. Acceso a Datos. 42](#_Toc386526349)

[5.2. DISEÑO DE BASE DE DATOS. 43](#_Toc386526350)

[5.2.1. Diagrama conceptual de la base de datos. 43](#_Toc386526351)

[5.2.2. Modelo Lógico. 44](#_Toc386526352)

[5.2.2.1. Modelo Lógico relacional de la base de datos 44](#_Toc386526353)

[5.2.2.2. Diseño lógico específico de la base de datos. 45](#_Toc386526354)

[5.2.2.3. Definiendo triggers. 47](#_Toc386526355)

[5.2.3. DISEÑO FISICO DE LA BASE DE DATOS. 49](#_Toc386526356)

[5.2.3.1. Creación de la base de datos 49](#_Toc386526357)

[5.2.3.2. Definiendo jobs. 50](#_Toc386526358)

[5.2.3.3. Plan de Backup 51](#_Toc386526359)

[CAPÍTULO V: IMPLEMENTACIÓN 54](#_Toc386526360)

[6. IMPLEMENTACIÓN. 55](#_Toc386526361)

[6.1. APLICACIÓN WEB. 55](#_Toc386526362)

[6.1.1. Implementación del catálogo de artículos. 55](#_Toc386526363)

[6.1.1. Implementación del carrito de compras 57](#_Toc386526364)

[6.1.2. Modelo de despliegue para la solución. 60](#_Toc386526365)

[CAPÍTULO VI: PRUEBAS 61](#_Toc386526366)

[7. Pruebas de software 62](#_Toc386526367)

[8. REFERENCIAS. 67](#_Toc386526368)

[8.1. BIBLIOGRAFÍA. 67](#_Toc386526369)

[8.2. ENLACES DE INTERNET. 68](#_Toc386526370)

ÍNDICE DE TABLAS

[Tabla 1 Actores del sistema Fuente: Elaboración propia 8](#_Toc386610165)

[Tabla 2 Fases de SCRUM Fuente: (*Elaboración propia)* 13](#_Toc386610166)

[Tabla 3 Roles y responsabilidades de SCRUM Fuente: 14](#_Toc386610167)

[Tabla 4 Prácticas y herramientas de SCRUM Fuente: 15](#_Toc386610168)

[Tabla 5 Fases de Xtreme Programming Fuente: 17](#_Toc386610169)

[Tabla 6 Roles y responsabilidades de XP Fuente: 18](#_Toc386610170)

[Tabla 7 Técnicas y prácticas de XP Fuente: 20](#_Toc386610171)

[Tabla 8 Product Backlog Fuente: 27](#_Toc386610172)

[Tabla 9 Historia de usuario: Registro de un pedido de venta Fuente: 35](#_Toc386610173)

[Tabla 10 Historia de usuario: Registro de un proveedor Fuente: 36](#_Toc386610174)

[Tabla 11 Historia de usuario: Registro de un artículo Fuente: 37](#_Toc386610175)

[Tabla 12 Diseño lógico específico de la base de datos Fuente: 46](#_Toc386610176)

[Tabla 29 Trigger para actualizar stock del artículo al guardar un detalle de venta. Fuente: 47](#_Toc386610177)

[Tabla 30 Triggers para actualizar el stock del producto al actualizar un detalle de venta 48](#_Toc386610178)

[Tabla 31 Implementación de un job para backup Fuente: 50](#_Toc386610179)

[Tabla 32 Test Case probar si agrega un mismo ítem en otra fila en el detalle del carrito de compras 62](#_Toc386610180)

[Tabla 33 Test Case Probar si guarda un pedido de venta controlando la integridad transaccional. 63](#_Toc386610181)

INDICE DE ILUSTRACIONES

Ilustración 1 Integración XP y SCRUM 9

Ilustración 2: Diagrama de Gantt 11

Ilustración 3 Proceso SCRUM*)* 12

Ilustración 4 Ciclo de Vida XP 16

Ilustración 5 Arquitectura y distribución de componentes de software 40

Ilustración 6 Diseño conceptual de la base de datos 43

Ilustración 7 Diseño lógico relacional de la base de datos 44

Ilustración 8 Conjunto de medios para el plan de backup 52

Ilustración 9 Modelo de despliegue en una arquitectura web. 60

# CAPÍTULO I: ASPECTOS METODOLÓGICOS

# INTRODUCCIÓN.

## PLANTEAMIENTO DEL PROBLEMA.

### Situación problemática.

Carlos y Pablo son dos viejos amigos que tiempo atrás trabajaron juntos en un exportador de golosinas. Ahora los dos amigos han decidido iniciar un nuevo emprendimiento juntos, se trata de un empresa de venta de cigarrillos, bebidas (gaseosas), golosinas, galletitas, etc, llamado “Maxi quiosco”.

### Situación deseada.

Se desea contar con una herramienta que facilite la gestión y seguimiento a las ventas de las de golosinas y productos en el “Maxi quiosco”.

### Delimitación de contenido.

#### Pedidos de los productos y proveedores.

Ellos, compran cigarrillos, gaseosas y golosinas a los proveedores, así que se les ocurre hacer un listado de proveedores con sus nombres, apellidos, dirección, teléfonos (s), e-mail (s), año nacimiento, y los artículos que le compran a cada proveedor. Pablo, asume la responsabilidad administrativa, así se encarga de cargar todos los datos en la computadora, que le permita automatizar la operatoria de sus negocios y la venta de artículos.

#### De los precios de cado producto.

Se desea poder obtener una lista de precios de cada proveedor por artículo y cantidad, y la lista de venta al público, así como saber el inventario que tienen de cada cosa.

Carlos, estudiante avanzado en Administración de empresa, asume la responsabilidad comercial, quiere poder consultar las listas de proveedores, la lista de precios del quiosco y las direcciones de los proveedores.

#### Observaciones.

* Cada artículo (golosinas, cigarrillos, caramelos, chocolates, etc.) poseen un stock crítico y un stock real.
* Los proveedores se encuentran codificados.
* Donde dice “teléfonos (s)”, significa que tiene más de uno.

## ACTORES.

Los actores que interactuarán con el sistema son:

|  |  |  |
| --- | --- | --- |
| CÓDIGO | ACTOR | DESCRIPCIÓN |
| ADM | Administrador | Se encarga de la administración del quiosco. Es el encargado de subir todos los datos en el sistema web, para así poder automatizar la operatoria de su negocio y la venta de artículos. |
| ECM | Encargado del área comercial | Se encarga del inventario y las órdenes de compra, para lo cual necesita consultar las listas de proveedores, la lista de precios del quiosco y las direcciones de los proveedores. |
| CLI | Cliente | Persona que solicita el servicio que ofrece el quiosco, comprar vía online (por un carrito de compras) ó de compra directa en el lugar. |

Tabla 1 Actores del sistema  
Fuente: Elaboración propia

# METODOLOGÍA.

## PROCESO METODOLÓGICO.

Para la planificación y gestión del proyecto se seguirá las fases que plantea la metodología Programación Extrema (XP), para el desarrollo del producto se utilizara el Framework SCRUM y UML para documentar los artefactos del sistema.

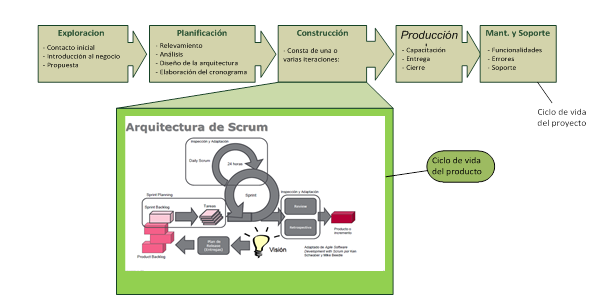


Ilustración 1 Integración XP y SCRUM  
Fuente: (Schwaber & Sutherland, 2012)

### Fase de exploración.

Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de “historias de usuarios”. (Schwaber & Sutherland, 2012)

### Fase de planificación.

La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas.

### Fase de iteraciones.

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.

En esta fase se empleara el Frameworks planteado por SCRUM. Para gestionar el desarrollo del producto.

### Fase de puesta en producción.

Al final de cada Sprint se entregan módulos funcionales, potencialmente usables, puede ser deseable por parte del cliente poner el sistema en producción.

En esta fase no se realizan más desarrollos funcionales, pero pueden ser necesarias tareas de ajuste. (Schwaber & Sutherland, 2012).

### Planificación temporal.

|  |  |
| --- | --- |
| **Proyecto** | Sistema de información para la gestión de venta Online Maxi- Kiosco |
| **Metodología** | Programación Extrema y Framework SCRUM |
| **Duración** | 14 meses |
| **Costo** | 12000 Sus. |
| **Contratista** | Ing. Ernesto Soto Roca |

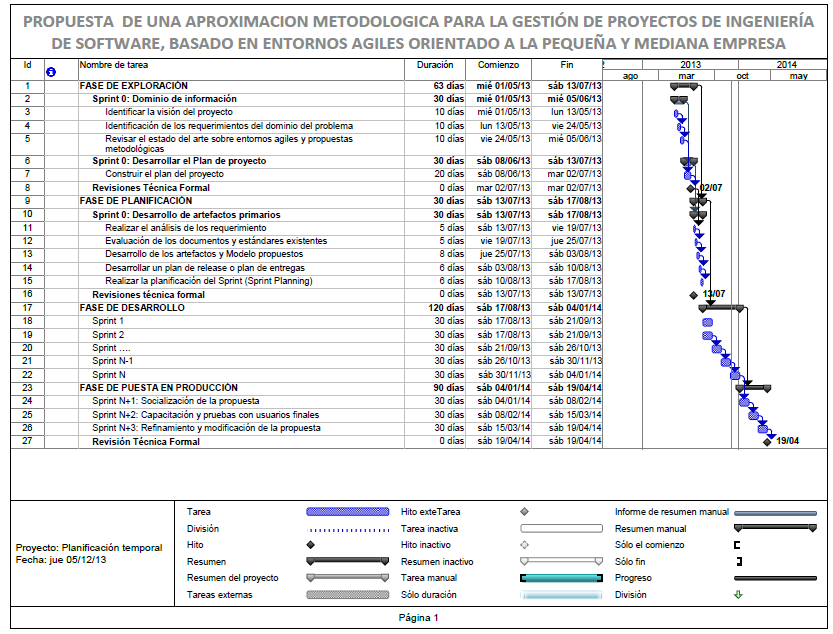


Ilustración 2: Diagrama de Gantt

Fuente Elaboración propia.

## SCRUM.

La primera vez que se asoció el término SCRUM a los procesos de desarrollo fue en 1986, cuando Nonaka y Takeuchi presentaron su artículo The New Product Development Game. Nonaka y Takeuchi presentaban en este artículo un proceso adaptativo, rápido y auto-organizado de desarrollo de productos. El término Scrum deriva del mismo término en rugby, que hace referencia a como se devuelve un balón que ha salido fuera del campo, al terreno de juego de una manera colectiva, la traducción al castellano sería melé.

Scrum surgió como práctica en el desarrollo de productos tecnológicos y no sería hasta 1993 que Jeff Sutherland aplicara el modelo al desarrollo de software en la Easel Corporation. En 1996 Sutherland presentó junto con Ken Schwaber las prácticas que empleaba como proceso formal para la gestión del desarrollo de software en OOPSLA 96. Estas prácticas de gestión pasarían a incluirse junto con otras muchas en la lista de modelos ágiles de Agile Alliance en el año 2001.

### Procesos.

En SCRUM se puede identificar tres fases: planificación del sprint, seguimiento del Sprint y revisión del sprint, como se puede apreciar en la Ilustración 2.

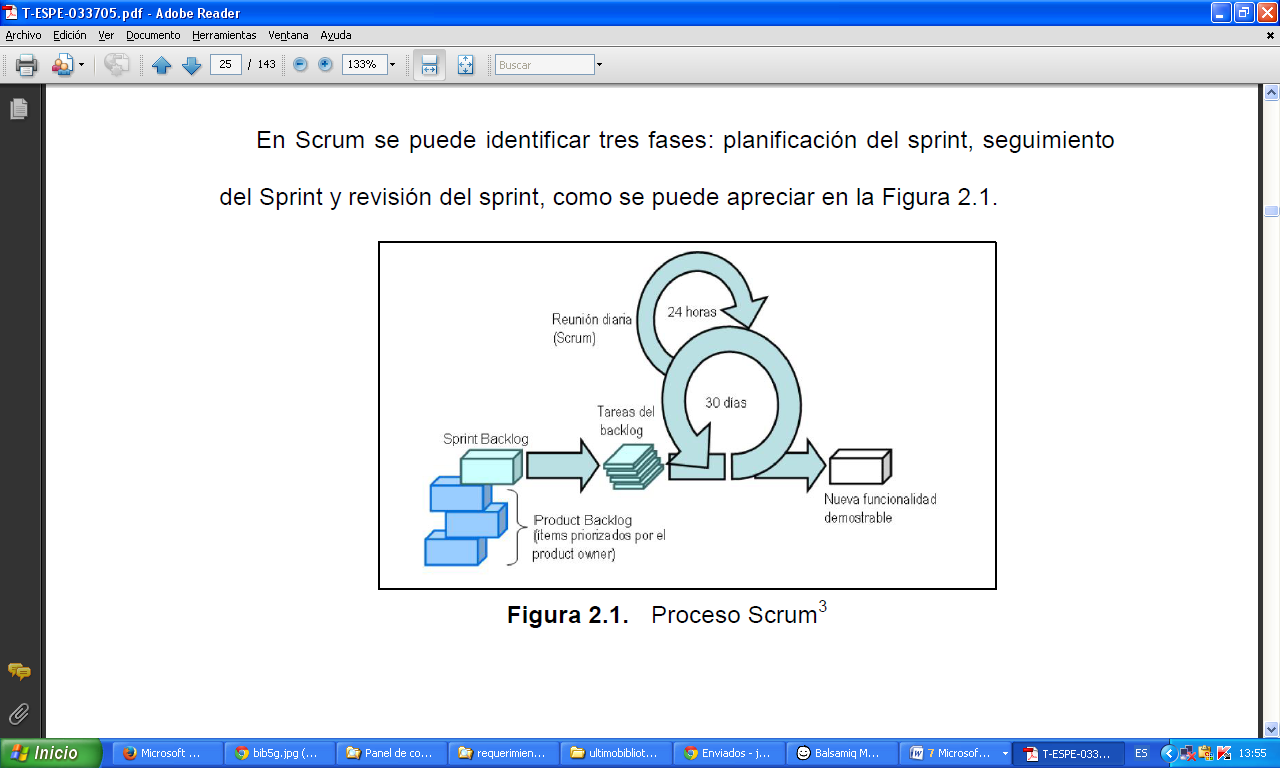


Ilustración 3 Proceso SCRUM  
Fuente: (*Flexibilidad con Scrum)*

En la Tabla 2 se puede observar las principales tareas que se llevan a cabo en cada fase.

|  |  |
| --- | --- |
| **Fase** | **Descripción** |
| **Planificación del Sprint** | En esta fase se define el Product Backlog. Si todavía no ha sido definido, consiste en una lista priorizada de requisitos del sistema y es un documento vivo, que puede ser continuamente actualizado. En cada iteración el Product Backlog es revisado por el equipo. También se lleva a cabo la planificación del primer Sprint. La planificación de cualquier sprint es la jornada de trabajo previa al inicio de cualquier sprint y en la cual se determinan cuáles son los objetivos y el trabajo que se deben cubrir en esa iteración. En esta reunión se obtiene una lista de tareas que se denomina Sprint Backlog, y el lema u objetivo principal del sprint. |
| **Seguimiento del Sprint** | A lo largo de esta fase se llevan a cabo breves reuniones diarias, para ver el avance de las tareas y el trabajo que está previsto para la jornada. En estas reuniones solo están presentes el SCRUM Master y el equipo, las preguntas que se realizan suelen ser tres:  1. Qué trabajo se ha realizado desde la reunión anterior.  2. Qué trabajo que se va a hacer hasta la próxima reunión.  3. Qué impedimentos que deben solventarse para proseguir con el trabajo. |
| **Revisión del Sprint** | Una vez finalizado el Sprint, se realiza un análisis y revisión del incremento generado. En esta reunión se presentan los resultados finales y se recomienda siempre tener preparada una demo. Existen múltiples razones para recomendar tener una demo al final de cada sprint, entre ellas la mejora del feedback con los interesados, reconocimiento del trabajo, un esfuerzo por finalizar las cosas o un correctivo en caso de tener una demo mal desarrollada. |

Tabla 2 Fases de SCRUM  
Fuente: (*Elaboración propia)*

### Roles y Responsabilidades.

En la Tabla 3 se describe los roles y responsabilidades de cada persona en SCRUM.

|  |  |
| --- | --- |
| **Rol** | **Descripción** |
| **SCRUM Master (Líder del Proyecto)** | Es el encargado de garantizar el funcionamiento de los procesos y de la metodología. Es importante darse cuenta que SCRUM Master es más que un rol, es la responsabilidad de funcionamiento de modelo, por tanto muchas veces es aconsejable utilizar a personas y puestos más adecuados según la organización. Un SCRUM master debe interactuar tanto con el equipo como con el cliente y con los gestores. |
| **Propietario del producto (Product Owner):** | Es la única persona del proyecto conocedora del entorno de negocio del cliente y de la visión del producto y es el responsable de obtener el resultado de mayor valor posible para el cliente. También es el responsable de la financiación necesaria para el proyecto, de tomar las decisiones que afecten a cómo va a ser el resultado final, fechas de lanzamiento y el retorno de inversión. Por regla general y si no se trata de proyectos internos, el propietario del producto suele ser el responsable del proceso de adquisición del cliente.  El equipo está involucrado en la estimación del esfuerzo de las tares del Product Backlog, en la creación del Sprint Backlog, etc. |
| **Equipo de desarrollo (SCRUM Team)** | Es el equipo del proyecto y tiene la autoridad para decidir en las acciones necesarias y para auto-organizarse con la finalidad de alcanzar los objetivos del Sprint. |

Tabla 3 Roles y responsabilidades de SCRUM  
Fuente: (*Elaboración propia)*

### Prácticas.

SCRUM no requiere y/o provee de ninguna práctica concreta para el desarrollo del software, sin embargo sí que dispone de prácticas y herramientas para la gestión de las diferentes fases de SCRUM. En la Tabla 4 se muestra las principales prácticas y herramientas de la metodología Scrum.

|  |  |
| --- | --- |
| **Prácticas** | **Descripción** |
| **Product Backlog (Pila del producto)** | Define los requisitos del sistema o el trabajo a hacer a lo largo del proyecto. Está compuesto por una lista de requisitos de negocios y técnicos, actualizados y priorizados. El responsable de mantener el product backlog es el propietario del producto. |
| **Sprint Backlog (Pila de tareas)** | Es una lista de trabajos que el equipo se compromete a realizar para generar el incremento previsto. Las tareas están asignadas a personas y tienen estimados el tiempo y los recursos necesarios. |
| **Estimación de esfuerzo** | Es un proceso iterativo en el cual las estimaciones de los ítems del product backlog son reajustadas acorde a la información obtenida en la última iteración. Este reajuste lo llevan a cabo el equipo de desarrollo y el propietario del producto. |
| **Gráfico Burn-down.** | Es una herramienta para gestionar y seguir el trabajo de cada sprint y representa gráficamente el avance del sprint. |
| **Gráfico Burn-up.** | Herramienta de gestión y seguimiento que sirve al propietario del producto para controlar las versiones de producto previstas, las funcionalidades de cada una, la velocidad estimada, fechas probables de cada versión, margen de error previsto en las estimaciones y avance real. |

Tabla 4 Prácticas y herramientas de SCRUM  
Fuente: (*Elaboración propia)*

## XTREME PROGRAMMING.

Xtreme Programming es sin duda es el abanderado de las metodologías ágiles. Nació como un intento, bastante exitoso, de establecer un conjunto de prácticas que facilitasen la finalización de los proyectos. Después de unas cuantas exitosas pruebas, estas prácticas se plasmaron de forma teórica, dando lugar a una metodología, que mantenía sus principales principios y prácticas. El término Xtreme es debido a que las prácticas que se utilizaron, fueron llevadas hasta el extremo.

### Procesos.

El ciclo de vida de Xtreme Programming está compuesto de seis fases, ver Ilustración 3.

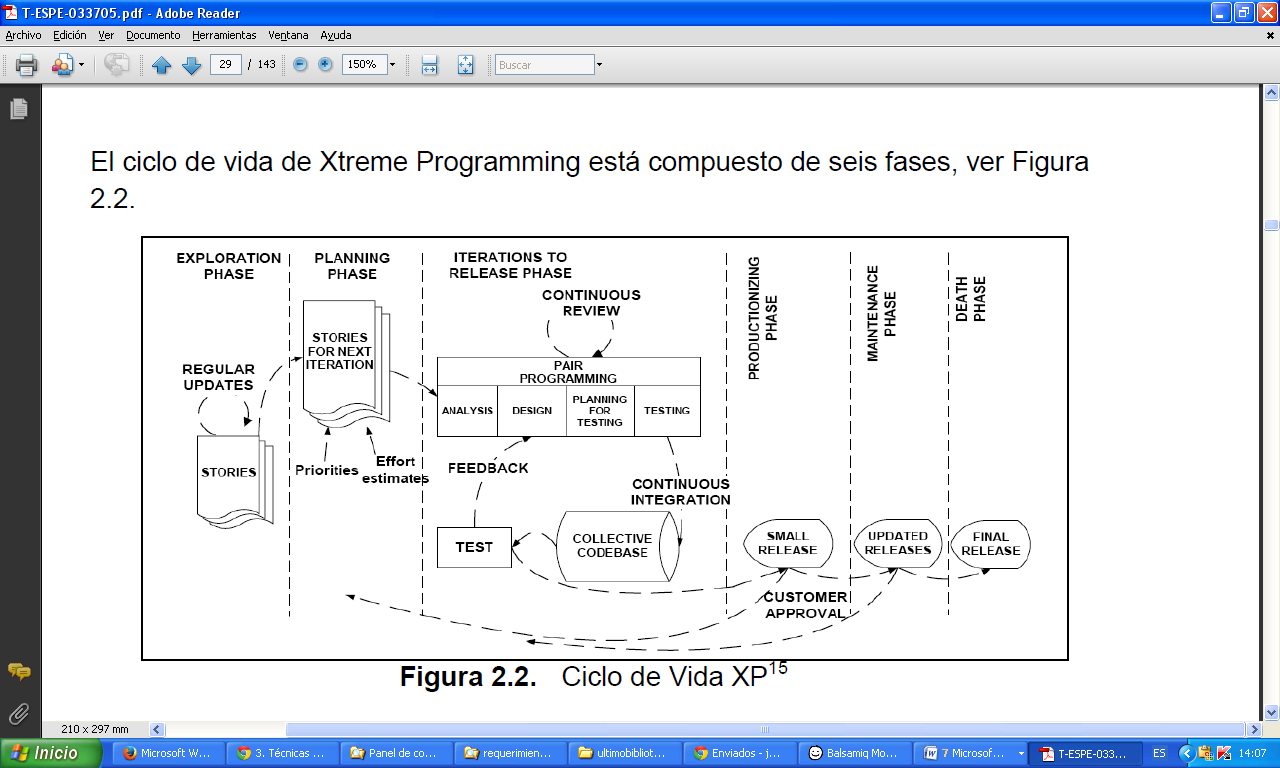


Ilustración 4 Ciclo de Vida XP  
 Fuente: (Escribano, 2002)

En la Tabla 5 se explica de una manera más detallada las fases de Xtreme Programming.

|  |  |
| --- | --- |
| **Fases** | **Descripción** |
| **Fase de exploración** | Los usuarios escriben las tarjetas de historia que ellos quieren que sean incluidas en la primera versión. Cada una de las tarjetas de historia describe una funcionalidad que será añadida al programa. El equipo de proyecto durante este tiempo se dedica a familiarizarse con las tecnologías y herramientas que utilizará a lo largo del proyecto, probando las herramientas y construyendo un prototipo simple para comprobar la arquitectura. El periodo de tiempo de esta fase puede variar desde unas pocas semanas hasta unos pocos meses, dependiendo de la familiaridad del equipo con las tecnologías. |
| **Fase de planificación** | Se establece la prioridad de las diferentes historias y se acuerda el contenido de la primera entrega del proyecto.  La estimación temporal se basa en un cálculo por parte de los desarrolladores de cada una de las historias, la primera entrega no suele tardar más de dos meses en realizarse.  La duración de esta fase no suele exceder el plazo de unos pocos días. |
| **Fase de iteraciones** | Esta fase incluye la realización de diferentes fases antes de liberar la primera versión del producto. La planificación realizada en la etapa anterior se divide en diferentes iteraciones, de una duración variable entre una semana y cuatro. Los usuarios son los que deciden que historias se van a realizar en cada iteración, sabiendo que en la primera se suele realizar un sistema con la arquitectura de todo el sistema, seleccionando aquellas historias que ayuden a construirla. Las pruebas funcionales creadas por el cliente son ejecutadas al final de cada iteración, de tal manera que al final de esta fase se obtiene una versión lista para producción. |
| **Fase de mantenimiento** | Una vez que se ha liberado la primera versión a los usuarios, el proyecto se debe mantener en el entorno de producción siempre y cuando aún haya iteraciones en fase de producción. Esto supone un esfuerzo considerable en la fase de mantenimiento e incluso se sugiere la contratación de nuevo personal para dar soporte a los clientes hasta puede cambiar la estructura del equipo.19 |
| **Fase de cierre del proyecto.** | Los clientes ya no tienen más historias que deban ser implementadas. Es necesario para que se considere que en esta fase se satisfagan todas las necesidades de los clientes y otros aspectos como fiabilidad, rendimiento, etc. La documentación del proyecto se realiza en esta fase, ya que ni la arquitectura, ni el diseño, ni el código sufrirán cambio alguno. Cuando el coste del producto es demasiado elevado para su desarrollo también se puede entrar en esta fase. |

Tabla 5 Fases de Xtreme Programming  
Fuente: (*Elaboración propia)*

### Roles y responsabilidades.

En la Tabla 6 se nombra los diferentes roles que se puede observar en un proyecto XP, según las tareas que desarrollan y los propósitos que persiguen.

|  |  |
| --- | --- |
| **Rol** | **Descripción** |
| **Cliente** | El cliente es el encargado de escribir las historias y las pruebas funcionales, y es el que decide cuando un requisito es satisfecho. También es el encargado de establecer la prioridad de las funcionalidades a implementar. |
| **Programador** | Escribe tanto las pruebas como el código de la aplicación y se debe comunicar fluidamente y frecuentemente con sus compañeros. |
| **Probador** | Ayuda a escribir las pruebas funcionales al cliente, las ejecuta regularmente, transmite los resultados de las pruebas y mantiene las herramientas. |

|  |  |
| --- | --- |
| **Probador** | Ayuda a escribir las pruebas funcionales al cliente, las ejecuta regularmente, transmite los resultados de las pruebas y mantiene las herramientas. |
| **Rastreador** | Es el encargado dar el “feedback” en XP. Se encarga de seguir las estimaciones hechas por los clientes y de ir avisando de las desviaciones posibles y como cuanto se ajustan a la realidad, con tal de mejorar las futuras estimaciones. También se encarga de realizar el seguimiento de las iteraciones y valorar si los objetivos se pueden alcanzar con los recursos disponibles (tiempo, personal) o si es necesario hacer algún cambio. |
| **Coach o tutor** | Es la persona responsable de todo el proceso. Es importante que tenga conocimientos y experiencia en otros proyectos de XP y de este modo guiar y ayudar al equipo a adaptarse a XP. |
| **Consultor** | Es un miembro externo que tiene los conocimientos técnicos necesarios. El consultor guía al equipo para solucionar los problemas específicos. |
| **Gestor o Manager** | Más conocido como el jefe. Es el que toma las decisiones críticas y está en permanente contacto con el equipo para poder discernir las diferentes situaciones críticas. |

Tabla 6 Roles y responsabilidades de XP  
Fuente: (*Elaboración propia)*

### Prácticas.

Xtreme Programming dispone de un gran abanico de técnicas y prácticas, muchas de ellas que han sido escogidas de diferentes metodologías existentes que tenían suficientemente probada su eficiencia.

En la Tabla 7 se lista las principales técnicas y prácticas de XP.

|  |  |
| --- | --- |
| **PRÁCTICAS** | **DESCRIPCIÓN** |
| **El juego de la planificación** | El equipo de desarrolladores estima el esfuerzo necesario para implementar las historias y los clientes determinan los objetivos y tiempos de entrega. |
| **Historias de usuario** | Son los requisitos del sistema formulados en una o dos sentencias, en el lenguaje común del cliente.24 |
| **Cortas y pequeñas iteraciones** | Un sistema simple se libera cada dos o tres meses y las diferentes versiones del mismo se suceden en periodos no superiores al mes. |
| **Metáforas** | El sistema se define utilizando un conjunto de metáforas acordadas entre el cliente y los programadores. Esta historia compartida guiará todo el proceso describiendo cómo funciona el sistema. |
| **Prácticas** | Descripción |
| **Diseño simple** | Se da gran importancia a la obtención de diseños simples que se puedan implementar rápidamente, evitando diseños complejos y código extra. |
| **Pruebas** | El desarrollo del software es orientado a pruebas (Test Driven Development). Las pruebas unitarias se escriben antes que el código y están funcionando continuamente. Las pruebas funcionales las escriben los clientes.25 |
| **Refactorizar** | Restructurar el sistema eliminando información duplicada, mejorando la comunicación, simplificando y añadiendo flexibilidad. |
| **Programación por pares:** | Es la técnica que promulga que dos personas escriban código en el mismo ordenador. |
| **Propiedad colectiva** | Cualquiera puede compartir cualquier parte de código con cualquier otro componente del equipo. |
| **Integración continúa** | Una nueva porción de código es integrada en el código fuente, tan pronto como este lista. El sistema es integrado y construido muchas veces a lo largo del día, todas las pruebas son ejecutadas y deben ser pasadas para aceptar la nueva porción de código. |
| **40 horas a la semana** | Se establece un máximo de 40 horas de trabajo  semanales. |
| **Prácticas** | Descripción |
| **Disponibilidad del cliente** | Los clientes deben estar disponibles y presentes cuando sean requeridos por el equipo de desarrollo. |
| **Estándares de codificación** | Las reglas de codificación son establecidas y seguidas por los programadores. Se enfatiza la comunicación a través del código. |
| **Espacios de trabajo abiertos** | Un gran espacio, con cubículos es lo recomendable. 27 |

Tabla 7 Técnicas y prácticas de XP  
Fuente: (*Elaboración propia)*

## COMBINACIÓN DE XTREME PROGRAMMING Y SCRUM.

La elección de las metodologías ágiles a utilizar en este proyecto se ha sustentado fundamental y principalmente, en la popularidad y la documentación, siendo SCRUM y Xtreme Programming las dos metodologías que cumplen estos requisitos.

## MÉTODOS Y TÉCNICAS DE SCRUM.

Se ha seguido con las indicaciones de Henrik Kniberg para la realización de proyectos con SCRUM y Xtreme Programming, que presenta en su libro Scrum y Xp desde las Trincheras. De tal manera que los métodos y técnicas que se utilizará para SCRUM son:

• **La pila del producto o product backlog:** Es el documento a través del cual se recogen los requisitos de los clientes.

• **La pila de tareas o sprint backlog:** Es el conjunto de historias, que pertenecen a la pila del producto, que se va a realizar en el sprint.

• **Estimación del esfuerzo:** Cada historia de la pila del producto es estimada con lo que se llama puntos de historia y que se corresponde a días-persona ideales.

• **Gráfico Burn-down:** Este gráfico ayuda a medir la productividad y detectar riesgos de mala distribución del trabajo y desviaciones temporales.

* + - Reuniones para cada sprint.
    - Reuniones de seguimiento diarias.

• **Planning Poker:** Esta técnica se utiliza a la hora de realizar la estimación de las historias. Simula el juego de Poker para obtener una estimación colectiva y cooperativa de cada historia.

## MÉTODOS Y TÉCNICAS DE XTREME PROGRAMMING.

Los métodos y técnicas que se tomará de Xtreme Programming son las siguientes:

• **Test Driven Development:** Diseñar las pruebas antes que implementar las funcionalidades. Todo código que pasa a producción tiene sus pruebas asociadas.

• **Refactorizar:** Rescribir aquellos métodos y clases, para mejorar la legibilidad del código. Esta es una técnica fundamental cuando se genera software de manera incremental y a partir de las pruebas.

• **Diseño incremental:** Mantener un diseño simple desde el principio e ir mejorándolo continuamente.

• **Integración continúa:** Esta técnica permite compilar el código cada vez que se sube al repositorio de código, ejecutar las pruebas y desplegarlo en un entorno de producción. A su vez, puede realizar este proceso de forma automática y cada cierto tiempo, de tal manera que se puede observar los diferentes resultados de las pruebas en diferentes condiciones y horas.

# CAPÍTULO II: REQUERIMIENTOS

# REQUERIMIENTOS DE SOFTWARE

## REQUERIMIENTOS FUNCIONALES (PRODUCT BACKLOG)

|  |  |
| --- | --- |
| Identificación del documento | |
| Código | MTRB-SCRUM-REQ-SRS-001 |
| Título | Gestión y seguimiento de los productos de maxi kiosco |
| Fecha creación | 10/04/2014 |
| Elaborado por | Ernesto Soto Roca |

|  |  |
| --- | --- |
| Lista de distribución | |
| DESARROLLO | Ernesto Soto Roca |
| {CLIENTE} | MAXKIOSCO |

|  |  |
| --- | --- |
| Revisión del documento | |
| Revisado por | Ernesto Soto Roca |
| En fecha | 12/04/2014 |

|  |  |
| --- | --- |
| Aprobación del documento | |
| Aprobado por | Ernesto Soto Roca |
| En fecha | 13/04/2014 |

|  |  |  |  |
| --- | --- | --- | --- |
| Control de versiones | | | |
| Versión | **Causa del Cambio** | **Responsable** | Fecha |
| V001 | **Afinamiento RNF** | **Ernesto** | 30/04/2014 |
|  |  |  |  |
|  |  |  |  |
| 01 | Versión Inicial | Ernesto Soto Roca | 18/04/2014 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Nombre del requerimiento** | **Como Probarlo**  **(Atributos de calidad)** | **Departamento seccion** | **Estimacion**  **(History point)** |
| RF001 | *Página de inicio* | * *Inicio de sesión de usuario.* * *Al ingresar que la pagina muestre el logo de la empresa, con una pequeña descripción de ella.* * *Un menú con las siguientes opciones:*   *Administrador:*   * *Inicio* * *Proveedores* * *Clientes* * *Inventario* * *Salir.*   *Cliente:*   * *Catálogo de productos* * *Carrito de compras* * *Salir* | *Depto. Inventario y administración* | *24* |
| RF002 | *Gestionar proveedores* | * *Permitir registrar, modificar y dar de baja a un proveedor y la búsqueda del mismo.* * *El código es autogenerado.* * *Se debe considerar que el proveedor tiene más de un teléfono.* | *Depto. Inventario y administración* | *16* |
| RF003 | *Gestionar clasificación de artículos* | * *Permitir registrar, modificar y dar de baja a una clasificación y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *La descripción de la clasificación es única.* | *Depto. Inventario y administración* | *8* |
| RF004 | *Gestionar Marca de artículos* | * *Permitir registrar, modificar y dar de baja a una marca y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *La descripción de la marca es única* | *Depto. Inventario y administración* | *8* |
| RF005 | *Gestionar artículos* | * *Permitir registrar, modificar y dar de baja a un artículo y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *La descripción del artículo es único* | *Depto. Inventario y administración* | *12* |
| RF006 | *Gestionar Unidades de los artículos* | * *Permitir registrar, modificar y dar de baja a una unidad del artículo y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *El código del artículo del proveedor, es aquel código con que tiene registrado el artículo en su propio sistema. Se lo utiliza para hacer la orden de compra. Se lo debe solicitar al proveedor en un inicio.* * *La descripción la unidad es única.* | *Depto. Inventario y administración* | *6* |
| RF007 | *Gestionar listas de precios de los proveedores* | * *Permitir registrar, modificar y dar de baja a una lista de precios y la búsqueda de la misma.* * *La lista es por producto, proveedor y unidad.* | *Depto. Inventario y administración* | *16* |
| RF008 | *Gestionar listas de precios al público* | * *Permitir registrar, modificar y dar de baja a una lista de precios y la búsqueda de la misma.* * *La lista de venta es por producto y unidad del artículo.* | *Depto. Inventario y administración* | *8* |
| RF009 | *Autenticación del usuario en el sistema* | * *Permitir el ingreso de usuarios al sistema web, mediante el login (nombre de usuario) y password.* * *La contraseña de usuario: longitud mínima 6 caracteres, máximo 12, por lo menos un carácter especial).* * *Los permisos dependerán del rol que tenga.* * *Se debe crear un usuario invitado, para aquel cliente que quiera ver el sistema sin registrarse.* | *Depto. Inventario y administración* | *16* |
| RF010 | *Gestionar empleados y roles* | * *Permitir registrar, modificar y dar de baja a un empleado ó un rol y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *La descripción del rol es único* | *Depto. Inventario y administración* | *24* |
| RF011 | *Gestionar clientes* | * *Permitir registrar, modificar y dar de baja a un cliente y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *Los campos CI y NIT son únicos.* | *Depto. Inventario y administración* | *16* |
| RF012 | *Gestionar carrito de compra* | * *Permitir al usuario añadir los productos en el carrito de compras, mientras navega por el catálogo de artículos. Permitiendo modificar la cantidad del artículo añadido ó quitar el artículo del carrito, hasta que no haya confirmado la compra.* * *En el momento de confirmación se creará en la base de datos un pedido de venta, al nombre del cliente y detallando los artículos del carrito de compras.* | *Depto. de comercialización* | *24* |
| RF013 | *Gestionar órdenes de compra* | * *Permitir registrar, modificar y dar de baja a una orden de compra y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *Se debe emitir una orden de recepción, a la orden de compra, cuando la mercadería llegue.* * *Se debe permitir guardar la factura, cuando el proveedor facilite el documento. Siendo el nro. de factura único.* * *Así mismo se debe permitir anular una recepción o factura.* | *Depto. Inventario y administración* | *24* |
| RF014 | *Gestionar pedidos de venta* | * *Permitir registrar, modificar y dar de baja a un pedido de venta y la búsqueda del mismo.* * *El código es autogenerado del sistema.* * *Se debe emitir una orden de entrega del producto (s), por pedido de venta, cuando se entregue los artículo(s) al cliente* * *Se debe permitir generar la factura por pedido de venta, cuando el cliente cancele el monto del pedido de venta. Siendo el nro. de factura único.* * *Así mismo se debe permitir anular una entrega o factura.* | *Depto. de comercialización* | *24* |
| RF015 | *Reporte: Listado de proveedores.* | *Permite al usuario ver un listado de proveedores, con la información básica para contactarlos.* | *Depto. Inventario y administración* | *8* |
| RF016 | *Reporte: Lista de precios por proveedor.* | *Permite al usuario obtener la lista de precios por artículo, por proveedor a una fecha determinada por el usuario.* | *Depto. Inventario y administración* | *8* |
| RF017 | *Reporte: Lista de precios.* | *Permite al usuario obtener la lista de precios por artículo para la venta al público, a una fecha (por defecto la fecha del día) determinada por el usuario.* | *Depto. Inventario y administración* | *8* |
| RF018 | *Reporte: Listado de stock por artículos* | *Permite al usuario ver un listado con el stock actual, y stock mínimo por artículo, clasificación y marca.* | *Depto. Inventario y administración* | *8* |
| RF019 | *Reporte: Orden de compra* | *Permite al usuario visualizar el documento de orden de compra.* | *Depto. administración y comercialización* | *8* |
| RF020 | *Reporte: Recepción de mercadería* | *Permite al usuario visualizar el documento de recepción de mercadería.* | *Depto. administración y comercialización* | *8* |
| RF021 | *Reporte: Pedido de venta* | *Permite al usuario visualizar el documento de pedido de venta.* | *Depto. administración y comercialización* | *8* |
| RF022 | *Reporte: Entrega de mercadería* | *Permite al usuario visualizar el documento de entrega de mercadería al cliente.* | *Depto. administración y comercialización* | *8* |
| RF022 | *Reporte: Factura* | *Permite al usuario visualizar el documento de facturación de un pedido de venta.* | *Depto. administración y comercialización* | *8* |

Tabla 8 Product Backlog  
Fuente: (Elaboración propia)

## REQUERIMIENTOS NO FUNCIONALES.

### Tiempo de aprendizaje.

La capacitación del personal, en el manejo de software y mantenimiento del sistema deberá considerar un tiempo considerable de por lo menos 40 horas laborales de capacitación (por día mínimo 2 horas.), hasta que los lineamientos de manejo del sistema y otros hayan sido debidamente adquiridos por el personal previamente asignado.

### Identificación del usuario propio de la aplicación.

El usuario ingresará al Sistema con su clave y contraseña, que será validada por el sistema, y los permisos al sistema según el rol que tiene asignado.

#### Contraseña de usuario.

Para la registración de la contraseña de usuario deberán asegurarse las siguientes prácticas:

* + La contraseña debe contener letras y números, y al menos un carácter especial.
  + La longitud de la contraseña debe ser de 7 caracteres mínimo, máximo 12.

### Confiabilidad.

#### Tiempo de disponibilidad del sistema.

La aplicación debe estar disponible las 24 horas todos los días del año, para los clientes que hagan su compra vía online y administradores de la empresa, y para los empleados de la empresa en su debido turno.

Debido a que cualquier usuario con usuario y contraseña (registrados en el sistema) puede tener acceso al sitio, es necesario hacer notar que para cumplir con la meta de la disponibilidad del sitio, se deberá analizar ciertos parámetros como:

* Rendimiento del servidor.
  + - * Calidad del enlace.
      * Calidad y velocidad de conexión del usuario final.

#### Tiempo fuera de servicio.

El tiempo máximo de fuera de operación depende del funcionamiento de los, servidores: web, base de datos y la infraestructura de red. El mismo debe ser:

* + Fallas comunes 40 minutos (aprox.)
  + Fallas no comunes 1hora. (aprox.)

### Performance.

#### Acceso de clientes en línea.

Los usuarios clientes deben poder acceder a los datos en línea, en tiempo real.

#### Tiempo de respuesta.

El tiempo de respuesta al acceso del usuario debe ir de 7 segundos, la primera vez que ingresa al sistema, luego menos de 5 segundos.

El tiempo de respuesta para una transacción promedio también debe ser de 10 segundos.

#### Cantidad de atención a usuarios.

El sistema debe poder atender normalmente a 40 o más usuarios a la vez.

### Soportabilidad o Facilidad de Mantenimiento.

#### Actualización transparente al usuario.

Debe aprovecharse la característica de la tecnología Web para que las actualizaciones y patchs de la aplicación se instalen y esta operación sea transparente al usuario. Caso contrario si es un cambio que necesite que el sistema esté fuera de línea debe realizarse en un horario dónde no exista mucho tráfico de datos.

#### Estándares de codificación.

Debe utilizarse los padrones entregados por IT “MTRB-SCRUM-CODIGO-FUENTE-S001”.

### Restricciones de Diseño.

#### Estándares de diseño.

* Para el diseño se deberá seguir el estándar establecido por el World Wide Web Consortium (W3C). El W3C produce recomendaciones para el diseño y desarrollo de páginas Web, en este caso XHTML y CSS.

### Estándares de arquitectura.

* Se debe usar para el desarrollo del software la arquitectura MVC, en 3 capas.
* Para la infraestructura de red, se debe implementar la arquitectura Cliente/Servidor.

### Motor de base de datos.

Se debe utilizar el motor de base de datos SQL Server 2008 de Microsoft o superior.

Se utilizaran los servicios de:

* Motor de base de datos
* Agent SQL
* Reporting Services

### Cliente del navegador.

La aplicación deberá ser accesible utilizando el siguiente tipo de navegador: Internet Explorer 10 o superior, Chrome y Firefox.

### Servidor Web.

El servidor Web debe ser Internet Información Server versión 7 o superior.

#### Lenguaje de programación.

La aplicación debe desarrollarse en .NET utilizando ASP.NET, Entity framework, Visual C#.Net, y derivados de SQL con T/SQL para el motor de base de datos SQL Server 2008 Service pack 3ª como mínimo.

### Requerimientos de documentación, ayuda en línea y manuales, asistencia técnica.

#### Ayuda en línea

Ayuda básica para guiarse con cada proceso. Habrá en cada ventana de usuario un botón con la imagen de (?)

### Interfaces.

#### Interfaz de usuario.

* EL sistema debe poseer una interfaz gráfica basada en el patrón “F” de distribución de componentes de manera intuitiva que permita la fácil interacción con el usuario.
* Mantener una combinación de colores de la interfaz que no afecte la utilización del software con el color Rojo como color corporativo y la gama de colores magenta gray.
* Colocar los botones ó íconos de forma ergonómica que no se le haga difícil al usuario utilizarlos.
* No deben existir presentación de imágenes no distorsionadas o difíciles de entender.
* La presentación de mensajes de error o de información al usuario deberá ser lo más específico posible y comunicarse con el administrador del sistema.

#### Interfaz de Hardware.

* Los usuarios necesitarán un dispositivo que mediante una interfaz de red (Ethernet, modem ADLS, etc.) les permita acceder al sistema, vía internet o red interna.
* Este dispositivo puede estar basado en cualquier arquitectura pero deberá disponer al menos de un procesador con capacidad suficiente para ejecutar un navegador Web moderno.

#### Interfaz de Comunicaciones.

* Existen dos tipos de conexión entre los usuarios y el servidor dónde está alojado el sistema web:
* Los usuarios clientes se conectarán al sistema web mediante Internet. Esta conexión la realizaran desde su casa o desde cualquier otro lugar con acceso a Internet pero en cualquier caso, desde fuera de la red local.
* Los empleados, en cambio, además de tener acceso al igual que los clientes desde su hogar a través de Internet, también podrán conectarse a través de la red interna de la empresa.

Cabe mencionar que en cualquiera de los dos casos mencionados anteriormente el mecanismo de comunicación será el mismo. Los clientes y el sistema web tendrán conexiones TCP/IP y el protocolo para comunicar a los navegadores con el servidor Web será el HTTP.

* Para el **Entorno de Red**: la aplicación deberá tener la capacidad de funcionar en un entorno de red LAN y WAN.

#### Interfaces de Software.

* + Cualquier usuario que desee conectarse al sistema web no necesitará ningún sistema operativo en concreto para poder acceder a ella. Únicamente bastará con un navegador moderno (IE10 o superior, Chrome y Firefox) para poder visualizarla.
  + Por otra parte el servidor de base de datos (SQL server 2008 o superior) y el servidor web (IIS7), se implementará bajo el sistema operativo Windows Server 2003 o superior.

# CAPÍTULO III: ANÁLISIS

# ANÁLISIS.

## ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE.

A diferencia de la metodología tradicional, XP utiliza las historias de usuario para la especificación de requisitos, permitiendo disminuir la documentación. XP presenta 4 valores que al seguirlos y utilizarlos facilita la especificación de requerimientos:

* **La comunicación:** permite que el cliente y el programador lleguen a un acuerdo en la especificación de requerimientos evitando los malos entendidos.
* **La sencillez**: es lo que diferencia a XP con las demás metodologías tradicionales las cuales utilizan estándares para la especificación de requerimientos que hacen del sistema muy complejo. La sencillez evita la documentación extensa centrándose en lo básico, en lo que se utiliza en este momento y no en lo que se podrá utilizar.
* **La realimentación**: permite que la especificación de requerimientos se comprenda mejor con el pasar del tiempo, permitiendo que los usuarios aprendan a describir mejor las Historias.
* **Las Historias de usuario**: es una pequeña descripción del programa con el fin de estimar tiempos y costos. Para obtener mayor detalle de las historias de usuario en el momento de la implementación, el programador preguntará al cliente, aumentando el detalle de cada historia.

### Historias de usuario.

Las historias de usuario permiten obtener los requerimientos del sistema a implementar los primeros requerimientos por parte del usuario, es importante no detallar las historias de usuario porque son utilizadas solo para dar una pequeña visión de lo que se quiere obtener. Cuando se inicie la fase de desarrollo el investigador con la ayuda del usuario detallarán las historias de usuario.

#### Historia de usuario para registro de un pedido de venta.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nro. de historia** | | **Titulo** | | | |
| 001 | *Registro de un pedido de venta* | | | | |
| **Descripción** | | | | | |
| Permite el registro de los datos necesarios para realizar un pedido de venta en el sistema, y al estar confirmado el pedido imprime un documento que acredita la transacción realizada. | | | | | |
| **Estimación (HP)** | | | **Prioridad** | | **Dependiente de** |
| 20 | | | Alta | |  |
| **Criterios de aceptación** | | | | | |
| * La interfaz de usuario debe seguir el patrón F para la distribución de controles y deducible para usar por el usuario. * El código del pedido de venta se debe generar de manera automática, y debe ser único para cada pedido de venta. * El sistema deberá informar si el artículo que se va a añadir al pedido de venta aún tiene stock disponible. No se puede vender un artículo que no tenga stock. * Se debe verificar que el stock del artículo satisfaga la cantidad solicitada del pedido. * El cliente que solicita el pedido de venta, puede o no estar registrado en el sistema. En caso de que no esté registrado, se le asignará un usuario de cliente por defecto, previamente creado como: cliente mostrador. * Si el cliente no está en sistema, pero quiere ser registrado, se hará el registro en la misma interfaz del pedido de venta. * Debe permitir buscar a un cliente y a un artículo, mediante una ventana de búsqueda. * El código del empleado no es obligatorio en caso de que el pedido de venta lo haga directamente el cliente mediante el carrito de compras online. * La fecha y hora del pedido de venta por defecto debe ser la fecha actual del servidor. * No se puede eliminar un pedido de venta de la base de datos, sólo se le puede dar de baja. * Al confirmarse el pedido se debe imprimir un documento comprobante. * Se debe crear una interfaz de búsqueda de pedido de ventas, para su posterior modificación. * Un pedido de venta no se podrá modificar, si esta anulado, entregado o facturado. | | | | | |
| Responsable | | | | Janeth | |
| **Adjunto** | | | | | |
| **C:\Documents and Settings\Nemesis\Escritorio\inventariodoc\RegistroPedidoVenta.bmpC:\Documents and Settings\Nemesis\Escritorio\inventariodoc\pedidosVenta.bmp** | | | | | |

Tabla 9 Historia de usuario: Registro de un pedido de venta  
Fuente: (Elaboración propia)

#### Historia de usuario para registro de un proveedor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nro. de historia** | **Titulo** | | | |
| 002 | *Registro de un proveedor* | | | |
| **Descripción** | | | | |
| Permite el registro de los datos necesarios de un proveedor, ya sea este natural o jurídico en el sistema.  Impresión de un reporte de: catálogo de proveedores. | | | | |
| **Estimación (HP)** | | **Prioridad** | | **Dependiente de** |
| 16 | | Alta | |  |
| **Criterios de aceptación** | | | | |
| * La interfaz de usuario debe seguir el patrón F para la distribución de controles y deducible para usar por el usuario. * El código del proveedor se genera de manera automática, y debe ser único para cada proveedor. * Un proveedor sólo puede tener un tipo de proveedor. * El nombre del proveedor jurídico es único. * Un proveedor puede tener más de un teléfono de contacto. * El sistema debe validar que los datos ingresados sean del tipo de datos requeridos por el sistema. * Se debe poder imprimir un catálogo de proveedores con datos básicos para contactarse con él. * Sólo se podrá eliminar un proveedor si no tiene asociado a él ninguna transacción, caso contrario sólo se le podrá dar de baja. | | | | |
| Responsable | | | Janeth | |
| **Adjunto** | | | | |
| **C:\Documents and Settings\Nemesis\Escritorio\inventariodoc\RegistroProveedor.bmp**  **C:\Documents and Settings\Nemesis\Escritorio\inventariodoc\catalogoprov.bmp** | | | | |

Tabla 10 Historia de usuario: Registro de un proveedor  
Fuente: (Elaboración propia)

#### Historia de usuario para registro de un artículo.

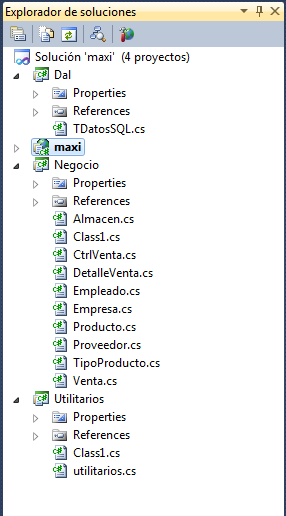
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nro. de historia** | **Titulo** | | | |
| 003 | *Registro de un artículo* | | | |
| **Descripción** | | | | |
| Permite el registro de los datos necesarios de un artículo en el sistema.  Impresión de un reporte de: catálogo de artículos. | | | | |
| **Estimación (HP)** | | **Prioridad** | | **Dependiente de** |
| 16 | | Alta | |  |
| **Criterios de aceptación** | | | | |
| * La interfaz de usuario debe seguir el patrón F para la distribución de controles y deducible para usar por el usuario. * El código del artículo se genera de manera automática, y es único para artículo. * El código de artículo del proveedor es un campo único. * Se debe especificar, la marca, la clasificación que tiene, y la unidad de medida para la venta. * El stock mínimo y el stock actual no puede ser negativos. * La descripción del producto es único, no debe repetirse. * Se debe poder imprimir un catálogo de artículos con datos básicos. * Sólo se podrá eliminar un artículo si no tiene asociado a él ninguna transacción, caso contrario sólo se le podrá dar de baja. | | | | |
| Responsable | | | Janeth | |
| **Adjunto** | | | | |
| **C:\Documents and Settings\Nemesis\Escritorio\inventariodoc\registroArticulo.bmp**  **C:\Documents and Settings\Nemesis\Escritorio\inventariodoc\catalogoprod.bmp** | | | | |

Tabla 11 Historia de usuario: Registro de un artículo  
Fuente: (Elaboración propia)

# CAPÍTULO IV: DISEÑO

# DISEÑO.

## Arquitectura de la solución para modelo en capas:



**Ilustración 2: Arquitectura de la solución.**

**Fuente: Elaboración propia.**

1. **Descripción de la arquitectura de la solución:**

A continuación se hace una breve descripción de los distintos componentes que forman parte de la solución en general.

* Biblioteca de clases Dal:

Biblioteca de clases que permite hacer la conexión al motor de base de datos de SQL Server, dentro de esta biblioteca se encuentra alojada una clase llamada TDatosSQL.cs, en esta clase están definidos los métodos principales para las operaciones al momento de manipular datos en la base de datos, estos métodos son guardar, modificar, eliminar y buscar.

* Biblioteca de clases Negocio:

Dentro de la clase Negocio se encuentra cada una de las clases que componen el cuerpo del sistema, estas clases se caracterizan por tener la definición de atributos, propiedades, constructores y métodos que permiten dar forma a la solución para cada uno de los requerimientos de negocios del sistema.

* Biblioteca de clases Utilitarios:

Se utiliza para definir variables que permitan al método realizar una acción, por ejemplo el método guardar, eliminar y modificar.

* Aplicación web de asp.net (Maxi Quiosco):

Dentro de la aplicación de asp.net se encuentran todos los elementos de diseño para el sistema, como ser la Master Page, la Página index.html, que son la base de diseño para poder invocar a los distintos formularios que forman parte del sistema, por ejemplo el formularios Proveedor (FormProveedor),Producto (FormProducto), Venta (FormVenta), etc.

## Arquitectura de la solución usando modelo mvc.

La arquitectura base para las soluciones desarrolladas en la empresa está reflejada en el siguiente gráfico.

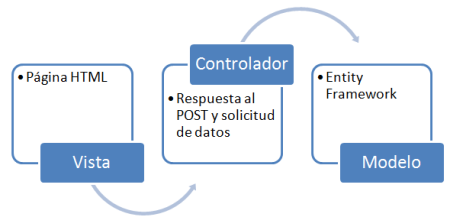


Ilustración 5 Arquitectura y distribución de componentes de software  
Fuente: compilando.es

La arquitectura base para el desarrollo de aplicaciones es MVC

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento

### Vista.

Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

### Modelo.

Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

### Controlador.

Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta de 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.

### Acceso a Datos.

Son los encargados de comunicarse con los repositorios de datos (normalmente bases de datos relacionales). Incluyen código específico de acceso a datos, como ser consultas, sintaxis o nomenclatura específica relacionada al repositorio de datos que se desea acceder.

## DISEÑO DE BASE DE DATOS.

### Diagrama conceptual de la base de datos.

Ilustración 6 Diseño conceptual de la base de datos   
Fuente: (Elaboración propia)

### Modelo Lógico.

#### Modelo Lógico relacional de la base de datos

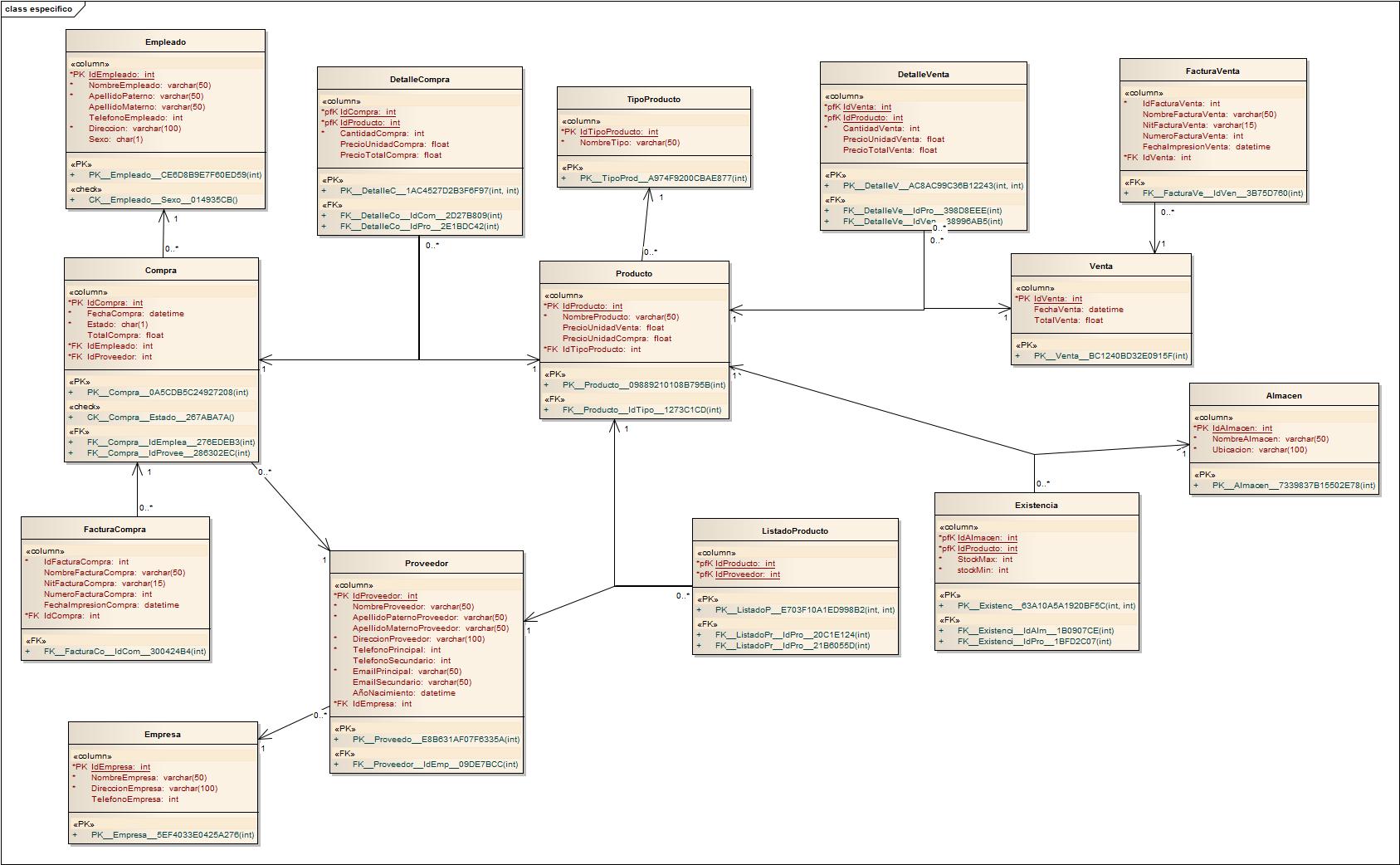
**

Ilustración 7 Diseño lógico relacional de la base de datos  
Fuente: (Elaboración propia)

#### Diseño lógico específico de la base de datos.

|  |
| --- |
| DISENO LOGICO ESPECIFICO PARA SQL.SERVER 2012 R2 |
| create database dbMaxiKiosco  use MAXIQUIOSCO  create table TipoProducto  (  Id int not null primary key identity,  NombreTipoProducto varchar(20) not null  )  create table Empleado  (  Id int not null primary key identity,  NombreEmpleado varchar(20) not null,  ApellidoPaterno varchar(20),  ApellidoMaterno varchar(20),  Telefono varchar(8) not null,  Usuario varchar(20) not null,  Contraseña varchar(20) not null  )  create table Empresa  (  Id int not null primary key identity,  NombreEmpresa varchar(20) not null,  Direccion varchar(50) not null,  Telefono varchar(8) not null  )  create table Proveedor  (  Id int not null primary key identity,  NombreProveedor varchar(20) not null,  ApellidoPaterno varchar(20),  ApellidoMaterno varchar(20),  Direccion varchar(50) not null,  Telefono varchar(30) not null,  Email varchar(50) not null,  FechaNacimiento datetime not null,  IdEmpresa int not null foreign key references Empresa(Id)  )  create table Producto  (  Id int not null primary key identity,  NombreProducto varchar(20) not null,  PrecioCompra float not null,  PrecioVenta float not null,  IdTipoProducto int not null foreign key references TipoProducto(Id)  )  create table ProductoProveedor  (  IdProducto int not null foreign key references Producto(Id),  IdProveedor int not null foreign key references Proveedor(Id),  primary key (IdProducto,IdProveedor)  )  create table Almacen  (  Id int not null primary key identity,  NombreAlmacen varchar(20) not null,  Direccion varchar(50) not null,  Capacidad int not null  )  create table Existencia  (  IdProducto int not null foreign key references Producto(Id),  IdAlmacen int not null foreign key references Almacen(Id),  primary key(IdProducto,IdAlmacen),  StockCritico int not null,  stockReal int not null  )  create table Venta  (  Id int not null primary key identity,  FechaVenta datetime not null,  PrecioTotal float  )  create table DetalleVenta  (  IdVenta int not null foreign key references Venta(Id),  IdProducto int not null foreign key references Producto(Id),  primary key (IdVenta,IdProducto),  Cantidad int not null,  Precio float not null,  SubTotal as (Cantidad\*Precio)  )  create table Compra  (  Id int not null primary key identity,  FechaCompra datetime not null,  PrecioTotal float,  IdEmpleado int not null foreign key references Empleado(Id),  IdProveedor int not null foreign key references Proveedor(Id)  )  create table DetalleCompra  (  IdProducto int not null foreign key references Producto(Id),  IdCompra int not null foreign key references Compra(Id),  primary key (IdProducto,IdCompra),  Cantidad int not null,  Precio float not null,  SubTotal as (Cantidad\*Precio)  ) |

Tabla 12 Diseño lógico específico de la base de datos  
Fuente: (Elaboración propia)

#### Definiendo Triggers.

|  |
| --- |
| Actualización de stock actual del producto al insertar un detalle de venta. |
| GO  CREATE TRIGGER tg\_detallePedidoVenta\_insert ON DetallePedidoVenta FOR INSERT  AS  BEGIN  DECLARE @stock INT  DECLARE @cantidad INT  SELECT @stock = a.stockActual ,@cantidad = I.cantidad FROM Articulo a INNER JOIN INSERTED I ON I.codigoArticulo = a.codigo  IF (@stock >= @cantidad)  BEGIN  UPDATE a SET a.stockActual = a.stockActual - I.cantidad  FROM Articulo a  INNER JOIN INSERTED I  ON I.codigoArticulo = a.codigo  END  ELSE  BEGIN  RAISERROR ('NO HAY stock SUFICIENTE PARA SATISFACER LA cantidad SOLICITADA PARA LA VENTA.VERIFIQUE E INTENTE NUEVAMENTE.', 16, 1)  ROLLBACK TRANSACTION  END  END  GO |

Tabla 29 Trigger para actualizar stock del artículo al guardar un detalle de venta.  
Fuente: (Elaboración propia)

|  |
| --- |
| Actualización de stock actual del producto al actualizar un detalle de venta. |
| CREATE TRIGGER tg\_detallePedidoVenta\_update ON DetallePedidoVenta FOR UPDATE  AS  IF (UPDATE(codigoPedidoVenta) OR UPDATE(codigoArticulo) OR UPDATE(precioUnidadMN))  BEGIN  RAISERROR('SÓLO SE PUEDE ACTUALIZAR LA cantidad.', 10, 1)  ROLLBACK TRANSACTION  END  ELSE  BEGIN  DECLARE @stock INT  DECLARE @cantidad INT    UPDATE a SET a.stockActual = a.stockActual + D.cantidad  FROM Articulo a INNER JOIN DELETED D ON D.codigoArticulo = a.codigo    SELECT @stock = a.stockActual ,@cantidad = I.cantidad FROM Articulo a INNER JOIN INSERTED I ON I.codigoArticulo = a.codigo  IF (@stock >= @cantidad)  BEGIN  UPDATE a SET a.stockActual = a.stockActual - I.cantidad  FROM Articulo a INNER JOIN INSERTED I ON I.codigoArticulo = a.codigo  END  ELSE  BEGIN  RAISERROR ('NO HAY stock SUFICIENTE PARA SATISFACER LA cantidad A ACTUALIZAR. VERIFIQUE E INTENTE NUEVAMENTE.', 16, 1)  ROLLBACK TRANSACTION  END  END  GO |

Tabla 30 Triggers para actualizar el stock del producto al actualizar un detalle de venta.  
Fuente: (Elaboración propiaMaxi1---

### DISEÑO FISICO DE LA BASE DE DATOS.

#### Creación de la base de datos

|  |
| --- |
| Scripts de creación de la base de datos |
| CREATE DATABASE dbMaxiKiosco  ON PRIMARY  ( NAME='MiBD\_Primaria',  FILENAME= 'c:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\data\MiBD\_Primaria.mdf',  SIZE=4MB,  MAXSIZE=10MB,  FILEGROWTH=1MB), -- Especifica el incremento de crecimiento automático del archivo  FILEGROUP dbMaxiKiosco\_GrupoArch1  ( NAME = 'dbMaxiKiosco\_GrupoArch1\_Datos1',  FILENAME = 'c:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\data\dbMaxiKiosco\_GrupoArch\_1.ndf',  SIZE = 1MB,  MAXSIZE=10MB,  FILEGROWTH=1MB),  ( NAME = 'dbMaxiKiosco\_GrupoArch1\_Datos2',  FILENAME = 'c:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\data\dbMaxiKiosco\_GrupoArch\_2.ndf',  SIZE = 1MB,  MAXSIZE=10MB,  FILEGROWTH=1MB)  LOG ON  ( NAME='dbMaxiKiosco\_log',  FILENAME = 'c:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\data\dbMaxiKiosco.ldf',  SIZE=1MB,  MAXSIZE=10MB,  FILEGROWTH=1MB);  GO  46  ALTER DATABASE dbMaxiKiosco  MODIFY FILEGROUP dbMaxiKiosco\_GrupoArch1 DEFAULT; --grupo de archivo por defecto  GO |

#### Definiendo jobs.

|  |
| --- |
| Script para sacar backups de la base de datos |
| USE dbMaxiKiosco;  GO  declare@fechavarchar(MAX)  declare@archivovarchar(MAX)  set@fecha=CONVERT(Varchar(max),GETDATE(),102)+'\_'+SUBSTRING(CONVERT(varchar(10),getdate(),108),1,2)+SUBSTRING(CONVERT(varchar(10),getdate(),108),4,2)+'horas'  set@archivo='D:\BACKUPSEMANAL\exgramaxi'+@fecha+'.bak'  BACKUPDATABASEexgrabiblio  TODISK=@archivo  WITHFORMAT,  MEDIANAME='D\_SQLServerBackups',  NAME='Full Backup of exgramaxi';  GO |
| Programación del trabajo |
|  |

Tabla 31 Implementación de un job para backup   
Fuente: (Elaboración propia)

Este job ejecuta un backup incremental cada día de la semana, a las 11:00 pm.

#### Plan de Backup

Se ha investigado y se optó por la Secuencia de Respaldo GFS (Grandfather-Father-Son).

El backup se ejecutara durante la noche, a menudo, aunque irá en función del hardware utilizado, podría no dar tiempo suficiente para la realización de todo el backup completo, por este motivo, es habitual dejar las copias completas para los fines de semana.

De esta forma, una planificación tipo podría ser la siguiente:

Sábados 11:00 PM: Copia completa. Medios: S1, S2, S3, S4, S5(\*)

* Lunes 11:00 PM: Copia diferencial. Medio: L
* Martes 11:00 PM: Copia diferencial. Medio: M
* Miércoles 11:00 PM: Copia diferencial. Medio: X
* Jueves 11:00 PM: Copia diferencial. Medio: J
* Viernes 11:00 PM: Copia diferencial. Medio: V

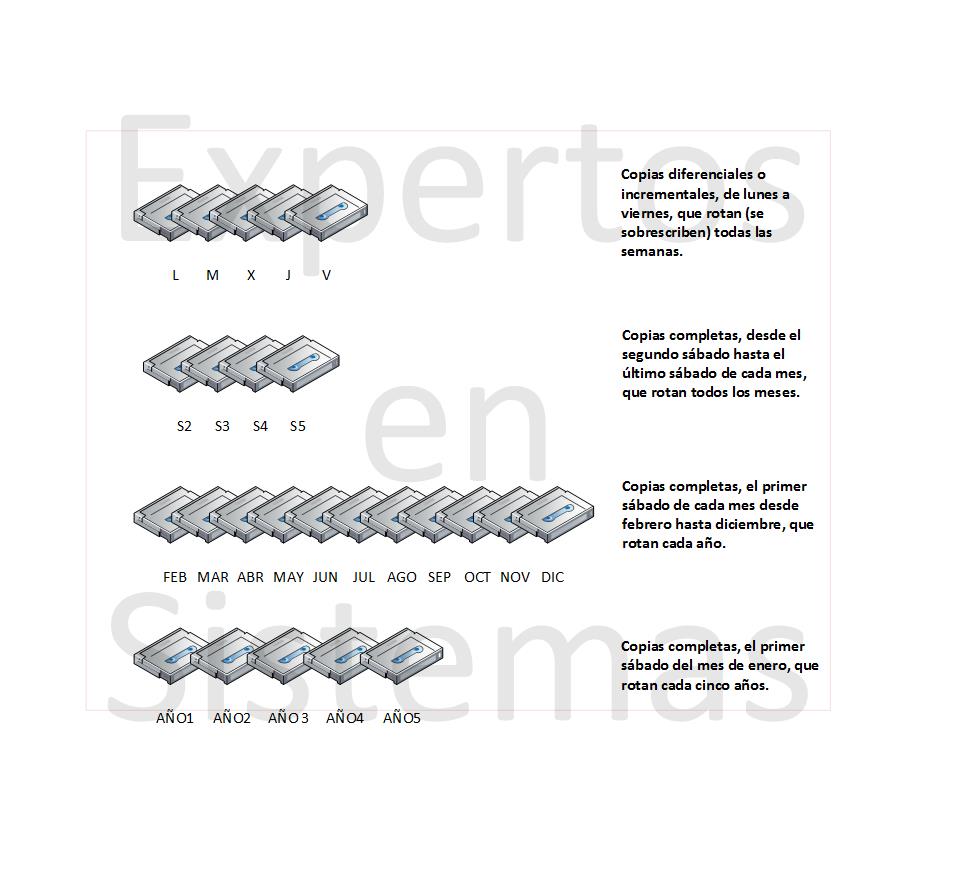
*(\*) S5 solo se utilizará en los meses que coincidan 5 sábados.*

Cada medio de copia completa Sn, en función del volumen de datos, podría estar formado por dos o más cintas LTO, por ejemplo, aunque debido a su alta capacidad, podría ser también de una sola cinta LTO. Sin embargo, lo más normal será que los medios diferenciales e incrementales sí que tengan suficiente con una única cinta LTO, ya que normalmente ocupan poco volumen. En tal caso, podríamos estar hablando de un total de 10 cintas LTO para empezar.

De esta forma, con tan solo 10 cintas podremos tener un plan de copias que permitirá restaurar todo el contenido hasta un mes hacia atrás. Como esto es poco, deberemos recurrir al archivado de medios por más tiempo, de forma que si sustituimos el medio S1 por uno que lleve el nombre del mes, estaremos guardando el contenido del primer sábado de cada mes durante un año:

ENE FEB MAR ABR MAY JUN JUL AGO SEP OCT NOV DIC

Con esta modificación, podremos restaurar toda la información hasta un año hacia atrás, pero si además sustituimos el medio ENE (enero) por cinco nuevos medios de rotación que se sobrescriban anualmente, podremos recurrir a un backup de la información hasta 5 años hacia atrás. Con este último cambio, el conjunto de medios resultante sería el siguiente:



Conjunto de medios para el Plan de Copias de Seguridad

Ilustración 8 Conjunto de medios para el plan de backup  
Fuente: expertosensistemas.com

 En definitiva, con un **conjunto de medios finito** (25 cintas LTO), hemos podido definir un plan de copias de seguridad que nos permite recuperar la siguiente información:

* Datos perdidos cualquier día de la semana anterior.
* En caso de necesitar datos más antiguos de una semana, se podrán recuperar de sábado en sábado, ya que los medios diarios habrán sido sobrescritos.
* En caso de necesitar datos más antiguos de un mes, se podrán recuperar de mes en mes (primer sábado de cada mes, que corresponde a las cintas etiquetadas con el nombre de cada mes).
* En caso de necesitar datos más antiguos de un año, se podrá recurrir a las cintas LTO etiquetadas como AÑO, que contendrán un backup completo del primer sábado de cada año.

Una opción bastante recomendable es adquirir una cinta LTO cada año, o bien, el número de cintas LTO que conformen el medio completo, y almacenar estas copias indefinidamente, etiquetándolas con el año en curso. De esta forma se obtiene un plan de copias de seguridad bastante completo con un número de medios moderadamente reducido

# CAPÍTULO V: IMPLEMENTACIÓN

# IMPLEMENTACIÓN.

## APLICACIÓN WEB.

### Implementación del catálogo de artículos.

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Web;

usingSystem.Web.Mvc;

namespaceMvcMaxiQuiosco.Controllers

{

publicclassCatalogoController : Controller

{

Models.exgramaxiEntities1CatalogoDB = newModels.exgramaxiEntities1();

publicActionResult Index()

{

varclasificacion = CatalogoDB.Clasificacion.ToList();

return View(clasificacion);

}

publicActionResult Buscar(stringclasificacion)

{

// Recuperar la clasificación asociada a los artículos desde la base de datos

varclasificacionModel = CatalogoDB.Clasificacion.Include("Articulos")

.Single(g =>g.descripcion == clasificacion);

return View(clasificacionModel);

}

publicActionResult Detalle(intcodigo)

{

var artículo = CatalogoDB.Articulo.Find(codigo);

return View(artículo);

}

[ChildActionOnly]

publicActionResultMenuClasificacion()

{

var clasificaciones= CatalogoDB.Clasificacion.ToList();

returnPartialView(clasificaciones);

}

}

}

namespaceMvcMaxiQuiosco.Models

{

using System;

usingSystem.Collections.Generic;

publicpartialclassClasificacion

{

publicClasificacion()

{

this.Articulo = newHashSet<Articulo>();

}

publicintcodigo { get; set; }

publicstringdescripcion { get; set; }

publicstringestado { get; set; }

publicvirtualICollection<Articulo>Articulo { get; set; }

}

}

namespaceMvcMaxiQuiosco.Models

{

using System;

using System.Collections.Generic;

publicpartialclassArticulo

{

public Articulo()

{

this.DetalleOrdenCompra = newHashSet<DetalleOrdenCompra>();

this.DetallePedidoVenta = newHashSet<DetallePedidoVenta>();

this.PrecioCompra = newHashSet<PrecioCompra>();

}

publicint codigo { get; set; }

publicstring descripcion { get; set; }

publicint stockCritico { get; set; }

publicint stockActual { get; set; }

publicint idMarca { get; set; }

publicint idClasificacion { get; set; }

publicint idUnidadVenta { get; set; }

publicint idUnidadCompra { get; set; }

publicstring rutaImagen { get; set; }

publicstring estado { get; set; }

publicNullable<decimal> precioVentaMN { get; set; }

publicvirtualClasificacion Clasificacion { get; set; }

publicvirtualMarca Marca { get; set; }

publicvirtualUnidadMedida UnidadMedida { get; set; }

publicvirtualUnidadMedida UnidadMedida1 { get; set; }

publicvirtualICollection<DetalleOrdenCompra> DetalleOrdenCompra { get; set; }

publicvirtualICollection<DetallePedidoVenta>DetallePedidoVenta { get; set; }

publicvirtualICollection<PrecioCompra>PrecioCompra { get; set; }

}

}

### Implementación del carrito de compras

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using ExGraMaxi.Models;

using ExGraMaxi.ViewModels;

namespace ExGraMaxi.Controllers

{

public class ShoppingCartController : Controller

{

Models.exgramaxiEntities storeDB = new Models.exgramaxiEntities();

//

// GET: /ShoppingCart/

public ActionResult Index()

{

var cart = ShoppingCart.GetCart(this.HttpContext);

// establecer el ViewModel

var viewModel = new ShoppingCartViewModel

{

CartItems = cart.GetCartItems(),

CartTotal = cart.GetTotal()

};

// Retornar la vista

return View(viewModel);

}

//

// GET: /Store/AddToCart/5

public ActionResult AddToCart(int id)

{

// Recuperar el articulo desde la base de datos

var addedArticulo = storeDB.Articulo

.Single(articulo => articulo.codigo == id);

// Anadirlo al carrito de compras

var cart = ShoppingCart.GetCart(this.HttpContext);

cart.AddToCart(addedArticulo);

// Retornar a la pagina principal para mas info.

return RedirectToAction("Index");

}

//

// AJAX: /ShoppingCart/RemoveFromCart/5

[HttpPost]

public ActionResult RemoveFromCart(int id)

{

// Quitar el item del carrito de compras

var cart = ShoppingCart.GetCart(this.HttpContext);

// Obtener el nombre del articulo para mostrar la confirmacion

string articuloNombre = storeDB.Cart

.Single(item => item.RecordId == id).Articulo.descripcion;

// Quitar del carrito de compras

int itemCount = cart.RemoveFromCart(id);

// Mostrar el mensaje de confirmacion

var results = new ShoppingCartRemoveViewModel

{

Message = Server.HtmlEncode(articuloNombre) +

" ha sido removido de su carrito de compras.",

CartTotal = cart.GetTotal(),

CartCount = cart.GetCount(),

ItemCount = itemCount,

DeleteId = id

};

return Json(results);

}

//

// GET: /ShoppingCart/CartSummary

[ChildActionOnly]

public ActionResult CartSummary()

{

var cart = ShoppingCart.GetCart(this.HttpContext);

ViewData["CartCount"] = cart.GetCount();

return PartialView("CartSummary");

}

//

// POST: /ShoppingCart/PedidoVenta

public ActionResult PedidoVenta()

{

var pedido = new PedidoVenta();

TryUpdateModel(pedido);

try

{

//Procesar pedido de venta

var cart = ShoppingCart.GetCart(this.HttpContext);

pedido.codigoCliente = 1;

pedido.fecha = DateTime.Now;

pedido.iva = 13;

pedido.estado = "V";

pedido.totalMN = cart.GetTotal();

//Guardar pedido de venta

storeDB.PedidoVenta.Add(pedido);

//storeDB.SaveChanges();

cart.CrearPedido(pedido);

return RedirectToAction("Complete",

new { id = pedido.codigo});

}

catch

{

//Error.

ModelState.AddModelError("", "Error al guardar el pedido de venta.");

return View("Error");

}

}

//

// GET: /Checkout/Complete

public ActionResult Complete(int id)

{

// Validar que el pedido sea del cliente.

bool isValid = storeDB.PedidoVenta.Any(

o => o.codigo == id &&

o.Cliente.codigo == 1);

if (isValid)

{

return View(id);

}

else

{

return View("Error");

}

}

}

}

### Modelo de despliegue para la solución.



## 

Ilustración 9 Modelo de despliegue en una arquitectura web.  
Fuente: (Elaboración propia)

# CAPÍTULO VI: PRUEBAS

# Pruebas de software

|  |
| --- |
| TEST CASE |
| Identificador: 0001  Título: Probar que no coloque un producto en otra fila si ya existe en el carrito de compras.  Descripción:  Seleccionar en el catálogo de articulos un producto que ya esta en el carrito y verificar que aumente la cantidad y no lo coloque como una fila en el detalle. |
| Pasos |
| 1. Ejecutar la aplicación  2. Buscar en el catálogo de artículos leche sin nata (seleccionar).  4. Agregar al carrito de compras (deberá mostrar el detalle del carrito.)  5. Seleccionar el mismo producto y agregar al carrito.  6. Deberá aumentar la cantidad del producto. Y actualizar el detalle del carrito de compras  7. Si es que no aumenta la cantidad de dicho artículo. No pasa la prueba. |
| Resultado Esperado |
| * Deberá indicar aumentar la cantidad del producto en el detalle del carrito y actualizar los demás datos del mismo. |
| ¿Automatizado?  no  Autor Janeth  Cuando se ejecutan  Resultado Actual  Comentarios/Observaciones |
|  |

Tabla 32 Test Case probar si agrega un mismo ítem en otra fila en el detalle del carrito de compras  
Fuente: Elaboración propia

|  |
| --- |
| TEST CASE |
| Identificador: 0002  Título: Probar que guarde el pedido de venta y muestre una confirmación  Descripción:  Debera confirmar el pedido de venta, y el sistema debera guardar el pedido de venta en la base de datos. |
| Pasos |
| 1. Ejecutar la aplicación  2. Agregar al carrito todos los artículos a comprar.  4. Confirmar el pedido de venta.  5. El sistema debe guardar, la cabecera del pedido de venta y los ítems que están en el carrito de compras como detalle del pedido.  6. Si todo sale bien, mostrar un mensaje de confirmación al usuario, con su número de pedido.  7. Si sale error. No pasa la prueba. |
| Resultado Esperado |
| * Deberá verificar si guarda un estudiante y confirmar la transacción. |
| Automatizado no |
| Autor Janeth  Cuando se ejecutan  Resultado Actual  Comentarios/Observaciones |
|  |

Tabla 33 Test Case Probar si guarda un pedido de venta controlando la integridad transaccional.  
Fuente: Elaboración propia

|  |
| --- |
| TEST CASE |
| Identificador: R003  Título: Probar si guarda un empleado correctamente  Descripción:  Introducir los datos de un empleado que ya este registrado, debera mostrar un error indicando que ya existe ese registro Requerimientos. |
| Pasos |
| 1. Abrir la aplicación  2. Ejecutar la opción registrar empleado (datos personales).  3. Cargar un empleado (por primera vez)  Código : 1  Nombre : juan  App : Sánchez.  Apm : López  Teléfono :72342232  Dirección :Barrio senac  Sexo :M  4. Presionar guardar (deberá mostrar un mensaje que guardo el registro.)  5. Ejecutar la opción registrar empleado.  6.Cargar los datos del mismo empleado (por segunda vez)  Código : 1  Nombre : juan  App : Sánchez.  Apm : López  Teléfono :72342232  Dirección :Barrio senac  Sexo :M  Deberá indicar que ya existe dicho proveedor y pedirá confirma la transacción.  7. Si es que no indica la existencia de dicho empleado. No pasa la prueba. |
| Resultado Esperado |
| * Deberá indicar que ya existe dicho empleado y pedirá confirma la transacción. |
| * Otros campos:   si |
| ¿Automatizado?  no  Autor Cinthia  Cuando se ejecutan |
|  |

Tabla 12: Prueba de software para guardar empleado

Fuente: Elaboración propia

|  |
| --- |
| TEST CASE |
| Identificador: R004  Título: Probar que el código es único del proveedor  Descripción:  Introducir el codigo del proveedor |
| Pasos |
| 1. Abrir la aplicación  2. Ejecutar la opción registrar proveedor (datos personales).  3. Cargar un proveedor (por primera vez)  Código : 1  … ….  4. Presionar guardar (deberá mostrar un mensaje que guardo el registro.)  5. Ejecutar la opción registrar proveedor.  Deberá indicar que ya existe dicho código proveedor  6. Si es que no indica la existencia de dicho proveedor. No pasa la prueba. |
| Resultado Esperado |
| * Deberá indicar que no existe dicho código proveedor. |
| * Otros campos:   si |
| ¿Automatizado?  no  Autor Cinthia  Cuando se ejecutan  Resultado Actual  Comentarios/Observaciones |
|  |

Tabla 13: Prueba de software para guardar proveedor

Fuente: Elaboración propia

|  |
| --- |
| TEST CASE |
| Identificador: R005  Título: Probar si guarda la venta correctamente  Descripción:  Introducir los datos de la venta, que ya este registrado, debera mostrar un error indicando que ya existe ese registro Requerimientos. |
| Pasos |
| 1. Abrir la aplicación  2. Ejecutar la opción registrar venta (datos).  3. Cargar una venta(por primera vez)  Código : 1  Fecha venta : 2014/04/12  Total venta : 10  Producto : coca-cola  4. Presionar guardar (deberá mostrar un mensaje que guardo el registro.)  5. Ejecutar la opción registrar venta.  6.Cargar los datos de la misma venta (por segunda vez)  Código : 1  Fecha venta : 2014/04/12  Total venta : 10  Producto : coca-cola  Deberá indicar que ya existe dicho venta y pedirá confirma la transacción.  7. Si es que no indica la existencia de dicha venta. No pasa la prueba. |
| Resultado Esperado |
| * Deberá indicar que ya existe dicha venta y pedirá confirma la transacción. |
| * Otros campos:   si |
| ¿Automatizado?  no  Autor Cinthia  Cuando se ejecutan  Resultado Actual  Comentarios/Observaciones |
|  |

Tabla 14: Prueba de software para guardar venta

Fuente: Elaboración propia

# REFERENCIAS.

## BIBLIOGRAFÍA.

**Schwaber, Ken;**[**Sutherland, Jeff**](http://en.wikipedia.org/wiki/Jeff_Sutherland)

*2012* - *Software in 30 Days: How Agile Managers Beat the Odds, Delight Their Customers, And Leave Competitors In the Dust.*

**Takeuchi, H y Nonaka, I**

*1986 - The New Product Development Game.*

**Kent Beck**

*1999 - Extreme Programming explained: Embrace Change.*

***Fernández Escribano, Gerardo***

*2002- Ingeniería de Software II. Introducción a Extreme Programming.*

***Palacio, Juan***

*2007 - Flexibilidad con Scrum, principios de diseño e implantación en campos Scrum. SafeCreative.*

Kaplan ‘Cu, D. (2001). Cuadro de Mando Integral. 2da Edición-2001.

Booch, G., Jacobson, I., & James, R. (2000.). El Lenguaje Unificado de Modelado. español año 2000.

Henrik Kniberg, & Prólogos de Jeff Sutherland y Mike Cohn. (2007 C4Media Inc). SCRUM Y XP DESDE LAS TRINCHERAS. Estados Unidos de América: Proyectalis.

IEEE-STD. (1998). IEEE-STD-830-1998 : ESPECIFICACIONES DE LOS REQUISITOS DEL. eeuu.

Joskowicz, J. (2008). Metodología de desarrollo de software: Reglas y Prácticas en eXtreme Programming. EEUU: Impreso en NOVATICA-EEUU.

Kniberg Pr, H., & SUTHERLAND-M. (s.f.). SCRUM y XP desde las trincheras.

Schwaber, k., & Sutherland, j. (2012). SCRUM\_GUIDE. UUEE: SCRUM Aliance.

## ENLACES DE INTERNET.

*http://upcommons.upc.edu/pfc/bitstream/2099.1/5608/1/50015.pdf*

*http://sedici.unlp.edu.ar/bitstream/handle/10915/5519/Documento%20completo.pdf?sequence=1*[*http://es.wikipedia.org/wiki/Scrum*](http://es.wikipedia.org/wiki/Scrum)[*http://assets.scrumfoundation.com/downloads/3/scrumprimer\_es.pdf*](http://assets.scrumfoundation.com/downloads/3/scrumprimer_es.pdf)*?*

*http://www.navegapolis.net/files/presentaciones/scrum.ppt*

*http://www.dosideas.com/wiki/Backlog\_Del\_Producto*

*http://www.navegapolis.net/files/Flexibilidad\_con\_Scrum.pdf*

*http://www.ia.uned.es/ia/asignaturas/adms/GuiaDidADMS/node61.html*

*http://www.slideshare.net/JoseYZF/metodologas-giles-xp-y-scrum*

*http://tallerinf281.wikispaces.com/file/view/Tabla+comparativa+de+Metodolog%C3%ADas.pdf  
 http://es.scribd.com/doc/83342499/practica2PPDF   
 http://www.cyta.com.ar/ta0502/v5n2a1.htm   
 http://infoq.com/minibooks/scrum-xp-fromthetrenches* [*http://www.intercambiosvirtuales.org/software/microsoft-sql-server-2008-r2-enterprise-edition-dvd-espanol*](http://www.intercambiosvirtuales.org/software/microsoft-sql-server-2008-r2-enterprise-edition-dvd-espanol)[*http://www.compilando.es/*](http://www.compilando.es/)[*http://repositorio.espe.edu.ec/bitstream/21000/5719/1/T-ESPE-033705.pdf*](http://repositorio.espe.edu.ec/bitstream/21000/5719/1/T-ESPE-033705.pdf) *http://www.expertosensistemas.com/estrategias-de-copias-de-seguridad-iii/*