



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE CIENCIAS DE LA
COMPUTACIÓN**

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

**TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO SISTEMAS E INFORMÁTICA**

**TEMA: ANALISIS, DISEÑO E IMPLEMENTACIÓN DE UNA
APLICACIÓN WEB DE CONTROL DE BODEGAS PARA LA
EMPRESA NOVAMUSIC, MEDIANTE LA UTILIZACION DE LA
PLATAFORMA JAVA ENTERPRISE EDITION JEE7 WEB**

AUTOR: CHANCHAY TITUAÑA, GABRIEL DARIO

DIRECTOR: ING. PRÓCEL, CARLOS

SANGOLQUÍ

2016

CERTIFICADO DEL DIRECTOR DEL TRABAJO DE TITULACIÓN



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

CERTIFICACIÓN

Certifico que el trabajo de titulación, ***“ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE CONTROL DE BODEGAS PARA LA EMPRESA NOVAMUSIC, MEDIANTE LA UTILIZACIÓN DE LA PLATAFORMA JAVA ENTERPRISE EDITION JEE7 WEB”*** realizado por el señor ***CHANCHAY TITUAÑA GABRIEL DARIO***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***CHANCHAY TITUAÑA GABRIEL DARIO*** para que lo sustente públicamente.

Sangolquí, 20 de Enero del 2016


CARLOS PROCEL

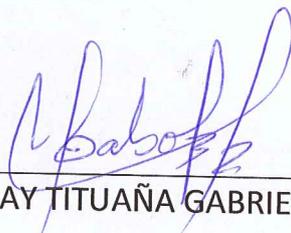
DIRECTOR

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

AUTORÍA DE RESPONSABILIDAD

Yo, **CHANCHAY TITUAÑA GABRIEL DARIO**, con cédula de identidad N° **1716261555** declaro que este trabajo de titulación "**ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE CONTROL DE BODEGAS PARA LA EMPRESA NOVAMUSIC, MEDIANTE LA UTILIZACIÓN DE LA PLATAFORMA JAVA ENTERPRISE EDITION JEE7 WEB**" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas. Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 29 de Enero del 2016



CHANCHAY TITUAÑA GABRIEL DARIO

CC: 1716261555



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

iv

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

AUTORIZACIÓN

Yo, **CHANCHAY TITUAÑA GABRIEL DARIO**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE CONTROL DE BODEGAS PARA LA EMPRESA NOVAMUSIC, MEDIANTE LA UTILIZACIÓN DE LA PLATAFORMA JAVA ENTERPRISE EDITION JEE7 WEB"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolquí, 29 de Enero del 2016

CHANCHAY TITUAÑA GABRIEL DARIO

CC: 1716261555

DEDICATORIA

Dedico este trabajo a Dios por ser mi luz y mi camino durante toda mi vida, porque sin él no hubiera podido concluir ninguno de mis logros tanto académicos como personales.

A mis Padres, quienes con su apoyo y valores han sido el principal apoyo para forjar mi espíritu de perseverancia y dedicación para poder alcanzar mis objetivos.

A mis amigos, compañeros de la universidad quienes compartieron conmigo sus conocimientos y me apoyaron día a día en mi vida estudiantil.

AGRADECIMIENTO

Le doy gracias a mi Dios por brindarme salud y sabiduría para poder afrontar mi vida estudiantil.

A mis padres por su apoyo incondicional, amor y dedicación en mí, sin ellos no hubiera podido ser la persona que soy.

A mi Director tesis quien con su conocimiento y experiencia supo guiarme para el desarrollo de esta tesis.

A Saviasoft la empresa que me ha dado la oportunidad de adquirir conocimientos y desarrollarme como profesional, a Daniel Cárdenas la persona que con su paciencia me ha transferido sus conocimientos y valores profesionales.

ÍNDICE DE CONTENIDO

CARATULA

CERTIFICACIÓN.....	ii
DECLARACIÓN	iii
AUTORIZACIÓN	iv
DEDICATORIA.....	v
AGRADECIMIENTO.....	vi
ÍNDICE DE CONTENIDO	vii
RESUMEN.....	xi
ABSTRACT	xii

CAPITULO I..... 1

GENERALIDADES

1.1. Planteamiento del Problema.....	1
1.2. Antecedentes.....	1
1.3. Justificación e importancia.....	2
1.4. Objetivos.....	3
1.4.1. Objetivo General	3
1.4.2. Objetivos Específicos.....	4
1.5. Alcance.....	4
1.6. Metodología.....	5
1.7. Factibilidad.....	5
1.7.1. Factibilidad Técnica.....	5
1.7.2. Factibilidad Operativa.....	6
1.7.3. Factibilidad Económica	6
1.8. Cronograma.....	8

CAPITULO II..... 9

MARCO TEÓRICO

2.1. PostgreSQL.....	9
2.1.1. Características	9

2.2. Birt	9
2.2.1. Arquitectura.....	10
2.2.2. Arquitectura.....	11
2.3. Lenguaje de Modelamiento Unificado (UML).....	12
2.3.1 Diagramas UML.....	12
2.4. Lenguaje de Programación JAVA.....	13
2.5. Metodologías de desarrollo Web	14
2.5.1. Metodología UWE	14
2.5.2. Metodología XP.....	16
2.5.3. Metodología HDM	17
2.5.4. Metodología RMM - RMDM.....	18
2.5.5. Metodología EORM.....	19
2.5.6. OOHDM	20
2.5.7. Metodología SCRUM	22
2.6. Análisis comparativo de Metodologías.....	27
Tabla 1: Comparación de Metodologías.....	27
2.7. Metodología Seleccionada.....	28
CAPITULO III.....	30
ANÁLISIS Y DISEÑO DE LA APLICACIÓN	
3.1. Captura de Requisitos	30
3.2. DEFINICION DE ACTORES	33
3.3. Modelo de Casos de Uso.....	34
3.4. CASOS DE USO DETALLADOS	37
3.4.1. Requerimientos No Funcionales	44
3.5. Diagrama de Clases	46
3.6. Artefactos propios de la metodología.....	47
CAPITULO IV	53
DESARROLLO.....	53
4.1. Arquitectura a Implementarse.....	53
4.1.1. Arquitectura Web.....	53

4.1.2. Modelo de N Capas.....	53
4.2. Estándares de implementación.....	54
4.2.1. Estandar J2EE.....	54
4.2.2. ESTANDARES DE CODIFICACION	55
4.3. Desarrollo de los módulos del sistema	57
4.4. INTEGRACION DEL CÓDIGO.....	57
4.5. PRUEBAS.....	58
CAPITULO V	59
CONCLUSIONES Y RECOMENDACIONES.....	59
5.1. Conclusiones	59
5.2. Recomendaciones	60
REFERENCIAS BIBLIOGRAFICAS	61

ÍNDICE DE TABLAS

Tabla 1 Presupuesto.....	7
Tabla 2 Comparación de Metodologías.....	27
Tabla 3 Crear Usuario.....	37
Tabla 4 Asignar Permisos	37
Tabla 5 Ingresar Proveedores.....	38
Tabla 6 Ingresar Secciones.....	38
Tabla 7 Ingresar Producto a la bodega.....	39
Tabla 8 Cambiar de sección un producto.....	39
Tabla 9 Devolución a proveedor.....	40
Tabla 10 Devolución de un producto.....	41
Tabla 11 Salida de un producto de bodega.....	41
Tabla 12 Reservar Producto.....	42
Tabla 13 Anular un registro de bodega.....	42
Tabla 14 Consultar le inventario	43
Tabla 15 Consultar Reportes.....	43
Tabla 16 Sprint 1	49
Tabla 17 Sprint 2	49
Tabla 18 Sprint 3.....	50
Tabla 19 Sprint 4.....	50
Tabla 20 Burn Down Chart.....	51

ÍNDICE DE FIGURAS

Figura 1 Cronograma de Tesis	8
Figura 2 Arquitectura Birt.....	11
Figura 3 Iteraciones SCRUM	23
Figura 4 Caso de Uso para Administrador.....	34
Figura 5 Casos de Uso Jefe de Bodega	35
Figura 6 Casos de Uso Documentos Bodega	35
Figura 7 Casos de Uso Gestor de Bodega	36
Figura 8 Diagrama de Clase	46
Figura 9 Proceso del Proyecto	52

RESUMEN

Este trabajo presenta el diseño, desarrollo e implementación de un software de gestión de bodegas SGBN, para la empresa NOVAMUSIC. Para llevarlo a cabo se ha empleado la Metodología SCRUM, ya que la empresa donde se implementó el software no tenía claro los requisitos, con esta metodología se puede realizar modelos incrementales modificando los requisitos en cada incremento, este trabajo se realizó utilizando java enterprise edition jee7 web, se utilizó la herramienta de Bussines Intelligence BIRT para el módulo de reportes a fin de dividir los reportes de la lógica de negocio, se utilizó PostgreSql para modelar e implementar la base de datos de este sistema. Con la implementación de este software en la empresa NOVAMUSIC los procesos de gestión de bodegas serán más rápidos, con estos procesos más ágiles los demás procesos de esta empresa también serán no solo más rápidos si no también proporcionarán información más veraz.

Palabras Clave:

- GESTIÓN DE BODEGAS
- METODOLOGÍA SCRUM
- BIRT
- JAVA WEB.

ABSTRACT

This paper presents the design, development and implementation of an management warehouse software SGBN, for NOVAMUSIC company. In order to implement it, the SCRUM methodology has been implemented, as the company didn't have clear the requirements with this methodology its possible to do incremental models modifying the requirements in each one, this work was made using java enterprise edition jee7 web, BIRT a business intelligence tool was used for the report module in order to divide the reports of the business logic, PostgreSQL was used to model and implement the data base of this software. With the implementation of this software in the NOVAMUSIC company the process of the warehouse management will be faster, with these faster process the other ones will be not only faster but also more effective.

Keywords:

- WAREHOUSE MANAGEMENT
- SCRUM METODOLOGY
- BIRT
- JAVA WEB.

CAPITULO I

GENERALIDADES

1.1. Planteamiento del Problema

El problema surge debido a la carencia de un sistema para controlar los productos que existen en la bodega de la empresa NOVAMUSIC de la ciudad de Quito.

Al no poseer la empresa registro de sus productos la empresa no puede saber con cuántos recursos cuenta, no puede agilizar los despachos de bodega, actualmente el sistema que posee para controlar la bodega es muy lento.

En la actualidad la empresa cuando quiere despachar o ingresar un producto en las bodegas los procesos son demasiado lentos sobre todo cuando existen grandes cantidades de productos.

Mediante esta aplicación se pretende solucionar las dificultades que tiene esta empresa para poder controlar los productos que entran y salen de la bodega, también poder manejar correctamente un stock de productos.

1.2. Antecedentes

La empresa NOVAMUSIC nació hace más de 12 años con el objetivo de ofrecer al mercado una variedad de productos musicales diferentes al resto de compañías que venden instrumentos musicales, ofreciendo accesorios varios para instrumentos musicales los cuales los mismos usuarios tenían que importan ellos mismos para poder usarlos.

Misión:

Somos una empresa que nace en el 2000 como distribuidora de accesorios e instrumentos musicales de marcas reconocidas a nivel mundial,

el ser músicos nos brinda la confianza de poder satisfacer las necesidades del músico bajo la práctica de una filosofía y valores para asegurar una relación permanente con nuestros clientes y proveedores, siempre partiendo de que: “los pequeños detalles hacen la mejor música”.

Visión:

Alcanzar y mantener una posición de liderazgo en la comercialización de accesorios e instrumentos musicales, fundamentado en la buena relación con nuestros proveedores para que las mejores marcas estén al alcance del músico a los mejores precios.

La empresa actualmente maneja la bodega manualmente debido a esto los procesos de toda la empresa se hacen más lentos, ya se han presentado problemas al momento de despachar una orden ya que hay veces que no solo hay un cliente esperando sino varios y en el momento de realizar el proceso estos tienen que esperar más tiempo del que estaba previsto

1.3. Justificación e importancia

Actualmente, la empresa NOVAMUSIC ubicada en la ciudad de Quito no cuenta con un sistema para poder apoyar a la gestión de bodegas, que cumpla con los requerimientos tecnológicos y de información para poder mantener actualizado su stock de productos.

Debido al crecimiento de la empresa es necesario un mejor control de toda la línea de productos musicales que la empresa ofrece a sus clientes.

En la actualidad la empresa maneja un stock de productos en Excel el cual no funciona con rapidez y eficacia, ya que cuando alguien quiere acceder al stock de bodegas cada empleado tiene un archivo diferente de Excel en sus computadores debido a esto la información que se maneja no es confiable y veras.

La administración de almacenes e inventarios es vital para la productividad de una organización.

Cuando se lleva a cabo una gestión de almacenes de manera tradicional, se incurre en muchos vicios y errores que impactan negativamente al cliente final.

Una empresa que pretenda ser competitiva en su mercado deberá calcular y valorar las ventajas que puede generarle la administración de almacenes. Así mismo, las empresas o instituciones cuya naturaleza no sea la fabricación y/o comercialización de mercancías, también deben hacer énfasis en lo mismo para obtener los beneficios que esto conlleva, y hacer del almacén un área estratégica para la consecución de objetivos.

Por lo contrario, las empresas que lleven a cabo una administración de almacenes empírica corren el riesgo de enfrentarse a serios problemas que pueden reflejarse cuantitativamente al momento de auditorías tanto contables como administrativas, ya sea internas o externas, o bien también pueden reflejarse en errores que pueden impactar en el cliente final, sin dejar de lado el famoso "robo hormiga" o pérdidas económicas por el deficiente manejo de materiales y/o producto terminado.

Mediante la automatización de los procesos busca también se busca agilizar los demás procesos de la empresa, ya que cuando el sistema esté en funcionamiento los demás procesos que la empresa maneje serán más rápidos ayudando así a una mejor atención al cliente.

1.4. Objetivos

1.4.1. Objetivo General

“Analizar, Diseñar e Implementar una aplicación web para la empresa NOVAMUSIC mediante la utilización de la Plataforma JAVA Enterprise Edition JEE7 Web.”

1.4.2. Objetivos Específicos

- Analizar los procesos de bodegas que se realizan en la empresa NOVAMUSIC.
- Analizar la metodología más eficaz para el desarrollo del sistema.
- Diseño de un ambiente Web en el que los miembros de la empresa NOVAMUSIC puedan disponer de información de forma rápida y confiable, de acuerdo a la metodología de desarrollo seleccionada.
- Implementar y probar el sistema con todas las funcionalidades que requiere la empresa.

1.5. Alcance

El tema “Análisis, Diseño e Implementación de una Aplicación Web para el control de bodegas en la empresa NOVAMUSIC mediante la utilización de la plataforma JAVA Enterprise Edition JEE7 web.” comprende de los siguientes módulos:

- Módulo de Inventario.- Este se encargara de gestionar todo el inventario de bodegas, de controlar los productos que entran y salen de la misma.
- Módulo de Administración.- Este módulo gestionara los usuarios del sistema y sus respectivos permisos.
- Módulo de Repotes.- Este módulo tendrá todos los reportes que el cliente solicite, se utilizara la herramienta de Bussines Intelligence BIRT.

El Sistema que se desarrollará llegará a su implementación en la empresa NOVAMUSIC, para el correcto desenvolvimiento de las actividades que tiene que desempeñar.

1.6. Metodología

El uso de una metodología para la creación de una aplicación es indispensable porque brinda las pautas para llevar de una manera organizada y sistemática las fases y artefactos necesarios para su desarrollo.

En las aplicaciones Web es muy importante un buen análisis y diseño del sistema para cumplir con todos los requerimientos del cliente, por lo cual se va analizar algunas metodologías para el desarrollo del sistema, una vez que se haya analizado las diferentes metodologías se escogerá una para el desarrollo de esta tesis, también se utilizará el lenguaje de modelamiento UML para la creación de los distintos artefactos a realizarse en cada una de las fases de desarrollo.

1.7. Factibilidad

1.7.1. Factibilidad Técnica

La empresa NOVAMUSIC otorgará la información necesaria para el desarrollo de las páginas web así como de la base de datos, también dará a conocer sus requerimientos.

- El Hardware con que se cuenta para la realización del proyecto es:
 - 1 PC de Escritorio, cuyas características principales son:
 - Marca: Clon
 - Procesador: Pentium 4
 - Velocidad: 3.36 Ghz
 - Memoria RAM: 1GB
 - Disco 60 GB.

- 1 Laptop, cuyas características principales son:
Marca: TOSHIBA
Procesador: Intel i5 64 bits
Memoria RAM: 6 GB
Disco Duro: 500 GB.
Requisitos en redes y equipos de comunicación
Acceso a internet.

1.7.2. Factibilidad Operativa

Las instalaciones en las que se va llevar a cabo el levantamiento y aprobación de los requisitos del proyecto serán en las oficinas de la empresa NOVAMUSIC, ubicadas en la calle Quisquis N2-38 y Atahualpa, Nayón. QUITO – ECUADOR.

La consultoría y seguimiento de este proyecto se realizará por parte de los coordinadores de tesis y el tesista.

1.7.3. Factibilidad Económica

Para la realización de esta aplicación web se utilizará software libre porque sus licencias son gratuitas, razón por la cual, la empresa NOVAMUSIC no tendrá que sobrellevar grandes gastos por la adquisición de licencias en herramientas de desarrollo.

Tanto las laptops, como los dispositivos de comunicación (tarjetas de red) serán de propiedad del desarrollador (tesista).

Tabla 1

Presupuesto

#	Descripción	Cantidad	V. Unitario	V. Total
1	HARDWARE			
	• Laptop	1	900	900
2	SOFTWARE			
	• Licencia Eclipse	1	0	0
	• Licencia PostgreSQL	1	0	0
	• Licencia Servidor de Aplicaciones Wildfly	1	0	0
4	CONSULTORÍA			
	• Desarrolladores	1	2160	2160
	Total USD			3060

1.8. Cronograma

Nombre de tarea	Duración	Comienzo	Fin
Elaboración y Aprobación del Plan de Tesis	18 días	jue 01/01/15	lun 26/01/15
[-] Analisis de los Requerimientos	52 días	vie 16/01/15	lun 30/03/15
[-] Captura de Requisitos	37 días	mar 27/01/15	mié 18/03/15
Análisis de los Requisitos de Contenido	6 días	mar 27/01/15	mar 03/02/15
Análisis de los Requerimientos de Interacción o de Usuario	5 días	mié 04/02/15	mar 10/02/15
Análisis de los requerimientos Navegacionales	5 días	mié 11/02/15	mar 17/02/15
[-] Levantamiento de Requerimientos Funcionales	16 días	mié 18/02/15	mié 11/03/15
Definir de Actores	2 días	mié 18/02/15	jue 19/02/15
Análisis de los procesos de la bodega	12 días	vie 20/02/15	lun 09/03/15
Establecimiento de las Limitaciones del Sistema	2 días	mar 10/03/15	mié 11/03/15
[-] Analisis de los requerimientos no funcionales	5 días	jue 12/03/15	mié 18/03/15
Análisis y propuesta de la infraestructura física sobre la cual el sistema deberá funcionar	3 días	jue 12/03/15	lun 16/03/15
Análisis de las dependencias que tendrá el sistema a nivel de Software	2 días	mar 17/03/15	mié 18/03/15
[-] Definición de los requisitos	47 días	vie 16/01/15	lun 23/03/15
Diseño de los modelos de casos de uso	4 días	vie 16/01/15	lun 23/03/15
Aprobación de los Requerimientos	5 días	mar 24/03/15	lun 30/03/15
[-] Diseño del sistema	18 días	lun 08/06/15	mié 01/07/15
[-] Diseño Conceptual	20 días	lun 08/06/15	vie 03/07/15
Elaboración de Diagramas de clases	5 días	lun 08/06/15	vie 12/06/15
Elaboración de artefactos de la metodología seleccionada	15 días	lun 15/06/15	vie 03/07/15
[-] Modelo Físico de Base de Datos	10 días	lun 08/06/15	vie 19/06/15
Elaboración del Modelo E-R	10 días	lun 08/06/15	vie 19/06/15
Codificación del Sistema	60 días	jue 02/07/15	mié 23/09/15
Pruebas del Sistema	10 días	jue 24/09/15	mié 07/10/15
Implementación de la infraestructura propuesta	10 días	jue 08/10/15	mié 21/10/15
Implementación del Sistema sobre la Infraestructura	7 días	jue 22/10/15	vie 30/10/15
Capacitación en el uso del Sistema	5 días	lun 02/11/15	vie 06/11/15

Figura 1 Cronograma de Tesis

CAPITULO II

MARCO TEÓRICO

2.1. PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y libre.

Como un proyecto de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada.

2.1.1. Características

Alta Concurrencia.- Mediante un sistema denominado MVCC (Acceso concurrente multiversión) permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*.

Funciones.- Bloques de código que se ejecutan en el servidor, se pueden ser escribir en varios lenguajes, con la potencia que cada uno de ellos da, desde las operaciones básicas de programación, tales como bifurcaciones y bucles, hasta las complejidades de la programación orientada a objetos o la programación funcional.

Triggers.- Los *triggers* son funciones enlazadas a operaciones sobre los datos. PostgreSQL soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser tratados igual a una fila retornada por una consulta.

2.2. Birt

BIRT es un sistema de información de código abierto basado en Eclipse para aplicaciones web, especialmente las basadas en Java y Java EE. BIRT posee dos componentes principales: un diseñador de informes basado en

Eclipse y un componente de tiempo de ejecución que se puede añadir a su servidor de aplicaciones. BIRT también ofrece un motor de gráficos que le permite agregar componentes a su propia aplicación, además ofrece las siguientes opciones:

- **Listas.-** Los reportes más simples son las listas de datos se puede agregar grupos para organizar los datos relacionados entre sí (pedidos agrupados por cliente, productos agrupados por proveedor). Si los datos son numéricos, se puede agregar fácilmente los totales, promedios y otros resúmenes.
- **Gráficos.-** BIRT proporciona gráficos circulares, líneas, barras y muchos más. Estos gráficos permiten la interacción del usuario.
- **Tablas de contingencia.-** Tablas de contingencia (también llamados una tabulación cruzada o matriz) muestra los datos en dos dimensiones: las ventas por trimestre o visitas por la página web.

2.2.1. Arquitectura

El proyecto BIRT proporciona muchos componentes los cuales son:

- **BIRT Report Designer.-** Es una perspectiva que se utiliza para crear diseños de informes BIRT. Estos diseños se guardan en un formato XML. Esta perspectiva se puede descargar como una aplicación Rich Client Platform (RCP), un conjunto de plug-ins para permitir la perspectiva Diseñador dentro del mismo Eclipse.
- **Diseño Motor.-** Este motor es el responsable de crear y modificar diseños de los reportes. Esta API de Diseño Motor contiene la funcionalidad del motor de diseño y está disponible para su uso dentro de cualquier proyecto Java / Java EE. El BIRT Report Designer utiliza esta API internamente para construir los diseños XML.

- **Motor de informes.-** El motor de informes utiliza los archivos de diseño de informes para generarlos. El Visualizador Web BIRT utiliza esta API para ejecutar y visualizar informes.
- **Trazando Motor.-** El motor de gráficos se utiliza para diseñar y generar todos los gráficos, ya sea en forma independiente o dentro de los informes. La API de gráficos Motor permite a los desarrolladores Java / Java EE para agregar capacidades de gráficos para sus aplicaciones. El diseño y el informe Motores hacen uso de esta API para entregar Gráficas.
- **BIRT Viewer.-** El proyecto BIRT proporciona una vista previa informes dentro de Eclipse. BIRT incluye un servidor Apache Tomcat invocado cada vez que se visualiza el informe. Además de ser empaquetado como un Eclipse Plug-in, el visor también está disponible como una aplicación Java EE independiente. Esta vista posee una salida de web como un único documento HTML, HTML paginado, PDF, XLS, DOC, PPT y Postscript. Además, el visor permite exportar los datos a CSV, la impresión y la tabla de la funcionalidad de contenidos.

2.2.2. Arquitectura

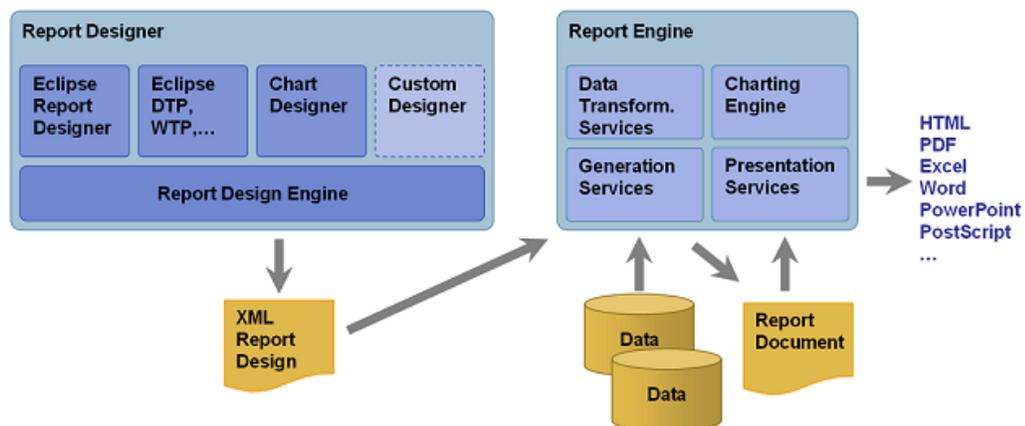


Figura 2 Arquitectura Birt.

Fuente: (ECLIPSE, 2014)

2.3. Lenguaje de Modelamiento Unificado (UML)

El lenguaje UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc.; hasta la implementación y configuración con los diagramas de despliegue.

UML es ante todo un lenguaje. Un lenguaje proporciona un vocabulario y una regla para permitir una comunicación. En este caso, este lenguaje se centra en la representación gráfica de un sistema.

Este lenguaje nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Un modelo UML está compuesto por tres clases de bloques de construcción:

- Elementos: Los elementos son abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
- Relaciones: relacionan los elementos entre sí.
- Diagramas: Son colecciones de elementos con sus relaciones.

2.3.1 Diagramas UML

Un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. Para poder representar correctamente un sistema, UML ofrece una

amplia variedad de diagramas para visualizar el sistema desde varias perspectivas. UML incluye los siguientes diagramas:

- Diagrama de casos de uso.
- Diagrama de clases.
- Diagrama de objetos.
- Diagrama de secuencia.
- Diagrama de colaboración.
- Diagrama de estados.
- Diagrama de actividades.
- Diagrama de componentes.
- Diagrama de despliegue.

2.4. Lenguaje de Programación JAVA

Java es un lenguaje de programación concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. La principal característica de Java es la de ser un lenguaje compilado e interpretado. Todo programa en Java ha de compilarse y el código que se genera bytecodes es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma.

Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de applets que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto.

Al ser orientado a objetos se refiere a un método de programación y al diseño del lenguaje. El objetivo de la orientación a objetos es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a

sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el “comportamiento” (el código) y el “estado” (datos). Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software.

2.5. Metodologías de desarrollo Web

La aparición de aplicaciones y sitios Web proporciona la explotación de otros mercados y servicios antes impensables como el comercio electrónico, la enseñanza virtual, etc y esto conlleva un importante crecimiento en el desarrollo del software sobre dicha tecnología. Desde el punto de vista de la ingeniería del software es importante dotar de los mecanismos adecuados, para que la realización de este tipo de aplicaciones satisfaga las necesidades tanto de los usuarios como de los clientes que contratan el desarrollo de este tipo de aplicaciones. Pero actualmente no existe una metodología universalmente aceptada, que guíe en el proceso de desarrollo de aplicaciones Web.

Existen criterios universalmente aceptados acerca del desarrollo software. Por ejemplo y según afirma Jacobson et al. (2000), el modelo de proceso más adecuado para el desarrollo de software es un proceso iterativo e incremental, puesto que a diferencia de otros modelos de proceso, como por ejemplo el modelo en cascada, permite la obtención de diversas versiones del producto software antes de la entrega final del mismo.

2.5.1. Metodología UWE

La propuesta de Ingeniería Web basada en UML (UWE (Koch, 2000)) es una metodología detallada para el proceso de autoría de aplicaciones con una definición exhaustiva del proceso de diseño que debe ser utilizado. Este proceso, iterativo e incremental, incluye flujos de trabajo y puntos de control

y sus fases coinciden con las propuestas en el Proceso Unificado de Modelado.

UWE está especializada en la especificación de aplicaciones adaptativas, y por tanto hace especial hincapié en características de personalización, como es la definición de un modelo de usuario o una etapa de definición de características adaptativas de la navegación en función de las preferencias, conocimiento o tareas de usuario.

Otras características relevantes del proceso y método de autoría de UWE son el uso del paradigma orientado a objetos, su orientación al usuario, la definición de un meta-modelo (modelo de referencia) que da soporte al método y el grado de formalismo que alcanza debido al soporte que proporciona para la definición de restricciones sobre los modelos.

Los principales aspectos en los que se fundamenta UWE son los siguientes:

- Uso de una notación estándar, para todos los modelos (UML: Lenguaje de modelado unificado).
- Definición de métodos: Definición de los pasos para la construcción de los diferentes modelos.
- Especificación de Restricciones: Se recomienda el uso de restricciones escritas (OCL: Lenguaje de restricciones de objetos) para aumentar la exactitud de los modelos.

Por lo que respecta al proceso de autoría de la aplicación, UWE hace un uso exclusivo de estándares reconocidos como UML y el lenguaje de especificación de restricciones asociado OCL. Para simplificar la captura de las necesidades de las aplicaciones web, UWE propone una extensión que se utiliza a lo largo del proceso de autoría. Este proceso de autoría está dividido en cuatro pasos o actividades:

- Análisis de Requisitos: Fija los requisitos funcionales de la aplicación Web para reflejarlos en un modelo de casos de uso.
- Diseño Conceptual: Materializado en un modelo de dominio, considerando los requisitos reflejados en los casos de uso.

- Diseño Navegacional: Lo podemos subdividir en: Modelo del Espacio de Navegacional.
- Modelo de la Estructura de navegación: Muestra la forma de navegar ante el espacio de navegación.
- Diseño de Presentación: Representa las vistas del interfaz del usuario mediante modelos estándares de interacción UML.

2.5.2. Metodología XP

XP es una metodología de desarrollo formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. XP puede ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Se puede considerar la programación extrema como la adopción de las mejores metodologías de desarrollo de acuerdo a lo que se pretende llevar a cabo con el proyecto y aplicarlo de manera dinámica durante el ciclo de vida del software. XP posee varias características:

Simplicidad.- La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hacen que la complejidad aumente exponencialmente. Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece.

Comunicación.- La comunicación se realiza de diferentes formas. Para los programadores el código comunica mejor cuanto más simple sea. Si el

código es complejo hay que esforzarse para hacerlo inteligible. El código autodocumentado es más fiable que los comentarios ya que éstos últimos pronto quedan desfasados con el código a medida que es modificado. Debe comentarse sólo aquello que no va a variar, por ejemplo el objetivo de una clase o la funcionalidad de un método.

Realimentación.- Considerando los problemas que derivan de tener ciclos muy largos. Meses de trabajo pueden tirarse por la borda debido a cambios en los criterios del cliente o malentendidos por parte del equipo de desarrollo. El código también es una fuente de retroalimentación gracias a las herramientas de desarrollo. Por ejemplo, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

2.5.3. Metodología HDM

El HDM o Hypertext Design Model, fue creado por Franca Garzotto, Paolo Paolini y Daniel Schwabe en 1991, dentro del marco del proyecto HYTEA de la Comunidad Europea. El objetivo era crear un modelo que fuera de utilidad para realizar el diseño de una aplicación de hipertexto. En 1993, F. Garzotto, L. Mainetti y P. Paolini introdujeron algunas mejoras al modelo HDM, al añadir estructuras de acceso como los índices y las visitas guiadas.

El modelo HDM divide el proceso de diseño de una aplicación de hipertexto en 2 partes: el *authoring-in-the-large*, que se refiere a la especificación y diseño de los aspectos globales y estructurales de la aplicación y el *authoring-in-the-small*, que se refiere al desarrollo del contenido de los nodos.

En HDM, el equivalente de un nodo es lo que denomina unidad. Las unidades se agrupan mediante una visita guiada o un índice, formando componentes, que a su vez se agrupan jerárquicamente en lo que denomina entidades. Una entidad es una jerarquía de componentes que "heredan" las propiedades de la entidad y que no pueden existir más que

como partes de la entidad. Las perspectivas permiten representar la multiplicidad de las presentaciones de un mismo contenido de información.

El principal objetivo de HDM fue crear un modelo basado en las llamadas primitivas antes de comenzar a desarrollarlo, lo que ayudará a conseguir una navegación más consistente y rica. Además HDM puede resultar útil también para evaluar aplicaciones ya desarrolladas, con el fin de detectar errores en la estructura navegacional.

Así pues, el modelo HDM establece una serie de nociones como: primitivas, entidades, componentes, perspectivas, unidades, enlaces y esquema HDM. Veamos con detalle en qué consisten estos conceptos:

Esquema HDM.- Una especificación de un hipertexto consta de una definición de esquema y un conjunto de definiciones de instancias. Un esquema especifica un conjunto de tipos de entidad y tipos de enlace. Las instancias se introducen en el sistema sólo si cumplen las restricciones establecidas en el esquema. El concepto de esquema en hipertexto es novedoso, pero ya ha sido ampliamente utilizado en el campo de las bases de datos.

El modelo HDM, ha sido la fuente de inspiración de los métodos RMM y OOHDM.

2.5.4. Metodología RMM - RMDM

La RMM o *Relationship Management Methodology* se define como un proceso de análisis, diseño y desarrollo de aplicaciones hipermedia. Los elementos principales de este método son el modelo E-R (Entidad-Relación) y el modelo RMDM (Relationship Management Data Model) basado en el modelo HDM. La metodología fue creada por Isakowitz, Stohr y Balasubramanian. Esta metodología es apropiada para dominios con estructuras regulares (es decir, con clases de objetos bien definidas y con claras relaciones entre esas clases).

El modelo propone un lenguaje que permite describir los objetos del dominio, sus interrelaciones y los mecanismos de navegación hipermedia de la aplicación. Los objetos del dominio se definen con la ayuda de entidades,

atributos y relaciones asociativas. La navegación se modeliza con la ayuda de primitivas de acceso, enlaces estructurales (unidireccional y bidireccional) que permiten especificar la navegación entre entidades. El esquema completo del dominio y de la navegación de la aplicación se denomina esquema RMDM y se obtiene como resultado de las tres primeras etapas del método. Las etapas son:

- Definición del protocolo de conversión de elementos del diagrama RMDM en objetos de la plataforma de desarrollo
- Concepción del interfaz usuario
- Concepción del comportamiento en ejecución
- Construcción del sistema y test

2.5.5. Metodología EORM

Es una Metodología de Relación de Objeto Mejorada (Enhanced Object Relationship Methodology), es definido por un proceso iterativo que se concentra en el modelado orientado a objetos por la representación de relaciones entre los objetos (acoplamientos) como objetos, es por ello que fue una de las primeras propuestas para Web centrada en el paradigma de la orientación a objetos. Se basa en muchas de las ideas que se definen en HDM, pero las traslada a la orientación a objetos. La adopción del enfoque orientado a objetos garantiza todas las ventajas reconocidas para esta técnica de modelado, como la flexibilidad (posible existencia de múltiples formas de relaciones entre nodos) y la reutilización, por la existencia de una librería de clases de enlaces que pueden ser reutilizados en diferentes proyectos de desarrollo hipermedial.

Podemos mencionar que esta metodología consta de las siguientes fases según el siguiente diagrama de flujo:

Fase de Análisis.- En esta fase se enfoca a los objetos del sistema, sin considerar los aspectos hipermediales del mismo, obteniéndose para ello un Modelo de Objetos con la misma notación utilizada en OMT, que refleje la estructura de la información (mediante clases de objetos con atributos y

relaciones entre las clases) y el comportamiento del sistema (a través de los métodos asociados a las clases de objetos)

Fase de Diseño.- Se procede a modificar el modelo de objetos obtenido durante el análisis añadiendo la semántica apropiada a las relaciones entre clases de objetos para convertirlas en enlaces hipermedia, obteniendo finalmente un modelo enriquecido, que su autor denomina EORM (Enhanced Object-Relationship Model), en el que se refleje tanto la estructura de la información (modelo abstracto hipermedial compuesto de nodos y enlaces) como las posibilidades de navegación ofrecidas por el sistema. sobre dicha estructura, para lo cual existirá un repositorio o librería de clases de enlaces, donde se especifican las posibles operaciones asociadas a cada enlace de un hiperdocumento, que serán de tipo crear, eliminar, atravesar, siguiente, previo etc., así como sus posibles atributos (fecha de creación del enlace, estilo de presentación en pantalla, restricciones de acceso, etc.)

Fase de Construcción.- Se transforma los esquemas en código una vez guardados en una Base de Datos Orientada a Objetos se elabora formularios de consulta de las clases con la ayuda de un editor gráfico de interfaces. Se genera el código fuente (por ejemplo en C#) correspondiente a cada clase y se prepara la Interfase Gráfica de Usuario.

2.5.6. OOHDM

Es un Método de Diseño de Desarrollo en Hipermedia Orientado a Objetos (Object-Oriented Hypermedia Design Method) y abarca las cuatro actividades: El modelado conceptual, diseño navegacional, diseño abstracto de interfaz y la puesta en práctica. Estas actividades se realizan en una mezcla de estilo incremental, iterativo y basado en prototipos de desarrollo.

Los modelos orientados a objetos se construyen en cada paso que mejora los modelos diseñados en iteraciones anteriores y consta de las siguientes fases:

Fase Conceptual.- Durante esta actividad se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. En las aplicaciones

hipermedia convencionales, cuyos componentes de hipermedia no son modificados durante la ejecución, se podría usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones). De este modo, en los casos en que la información base pueda cambiar dinámicamente o se intenten ejecutar cálculos complejos, se necesitará enriquecer el comportamiento del modelo de objetos. En OOHDM, el esquema conceptual está construido por clases, relaciones y subsistemas. Las clases son descritas como en los modelos orientados a objetos tradicionales. Sin embargo, los atributos pueden ser de múltiples tipos para representar perspectivas diferentes de las mismas entidades del mundo real.

Fase Navegacional.- Se debe tener en mente que la generación de aplicaciones Web fue pensada para realizar navegación a través del espacio de información, utilizando un simple modelo de datos de hipermedia. En OOHDM, la navegación es considerada un paso crítico en el diseño aplicaciones.

Fase de Interfaz Abstracta.- Con las estructuras navegacionales definidas, se deben especificar los aspectos de interfaz. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, de cómo los objetos de interfaz activarán la navegación y el resto de la funcionalidad de la aplicación, qué transformaciones de la interfaz son pertinentes y cuándo es necesario realizarlas.

El aspecto de la interfaz de usuario de aplicaciones interactivas (en particular las aplicaciones Web) es un punto crítico en el desarrollo que las modernas metodologías tienden a descuidar.

En OOHDM se utiliza el diseño de interfaz abstracta para describir la interfaz del usuario de la aplicación de hipermedia. El modelo de interfaz ADVs (Vista de Datos Abstracta) especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación

Fase Implementación.- Una vez implementado el diseño, se define los ítems de información que son parte del dominio del problema.

2.5.7. Metodología SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

El proceso

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

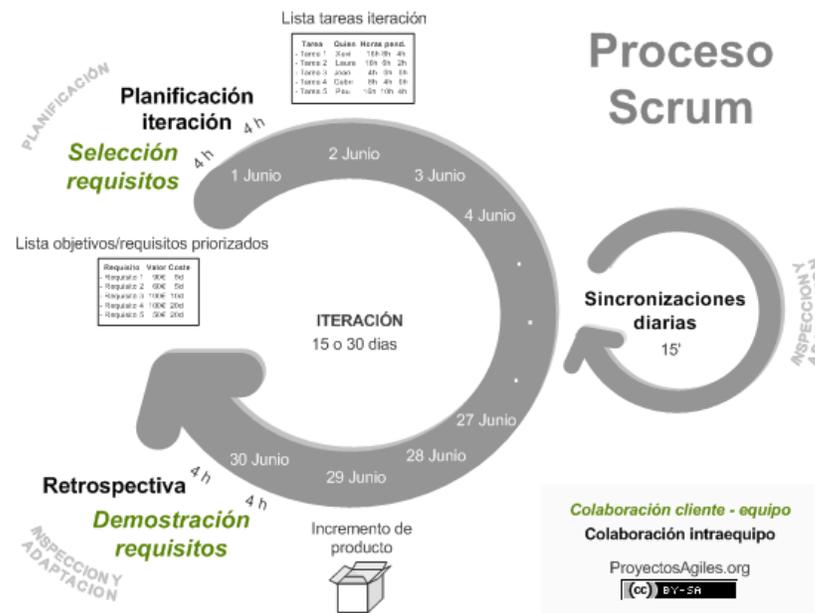


Figura 3 Iteraciones SCRUM

Fuente: (scrumespanol, 2016)

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la re planificación de objetivos del producto, que realiza durante la iteración con vista a las siguientes iteraciones. Las actividades que se llevan a cabo en Scrum son las siguientes:

Planificación de la iteración

Selección de requisitos (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Planificación de la iteración (4 horas máximo). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a

que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se auto asignan las tareas.

Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos máximo). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido.

En la reunión cada miembro del equipo responde a tres preguntas:

- ¿Qué he hecho desde la última reunión de sincronización?
- ¿Qué voy a hacer a partir de este momento?
- ¿Qué impedimentos tengo o voy a tener?

Durante la iteración el Facilitador (Scrum Master) se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad.

Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

Demostración (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, re planificando el proyecto.

Retrospectiva (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados.

Reunión de Planificación del Sprint (Sprint Planning Meeting)

Al inicio de cada ciclo de Sprint (cada 15 o 30 días), se lleva a cabo una *reunión de planificación del Sprint*. Se pretende:

- Seleccionar qué trabajo se hará.
- Preparar, con el equipo completo, el Sprint Backlog que detalla el tiempo que llevará hacer el trabajo.
- Identificar y comunicar cuánto del trabajo es probable que se realice durante el actual Sprint.
- Realizarse esta planificación en ocho horas como tiempo límite.
- Al final del ciclo Sprint se hacen dos reuniones más: la *reunión de revisión del Sprint* y la *retrospectiva del Sprint*.
- Reunión de Revisión del Sprint (Sprint Review Meeting)
- Revisar el trabajo que fue completado y no completado
- Presentar el trabajo completado a los interesados (alias “demo”)
- El trabajo incompleto no puede ser demostrado
- Cuatro horas como límite

Retrospectiva del Sprint (Sprint Retrospective)

Después de cada sprint, se lleva a cabo una retrospectiva del sprint, en la cual todos los miembros del equipo dejan sus impresiones sobre el sprint recién superado. El propósito de la retrospectiva es realizar una mejora continua del proceso. Esta reunión tiene un tiempo fijo de cuatro horas.

Sprint

El Sprint es el período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo con base en su propia experiencia. Se puede comenzar con una duración de sprint en particular (2 o 3 semanas) e ir ajustándolo con base en el ritmo del equipo, aunque sin relajarlo demasiado. Al final de cada sprint, el equipo deberá presentar los avances logrados y el resultado obtenido es un producto potencialmente entregable al cliente. Asimismo, se recomienda no agregar objetivos al sprint o *sprint backlog* a menos que la falta de estos objetivos amenace al éxito del proyecto. La constancia permite la concentración y mejora la productividad del equipo de trabajo.

Documentos

Product backlog

El *product backlog* se trata como un documento de alto nivel para todo el proyecto. Es el conjunto de todos los requisitos de proyecto, el cual contiene descripciones genéricas de funcionalidades deseables, priorizadas según su retorno sobre la inversión (ROI) . Representa el *qué* va a ser construido en su totalidad. Es abierto y solo puede ser modificado por el *product owner*. Contiene estimaciones realizadas a grandes rasgos, tanto del valor para el negocio, como del esfuerzo de desarrollo requerido. Esta estimación ayuda al *product owner* a ajustar la línea temporal (KEV) y de manera limitada, la prioridad de las diferentes tareas. Por ejemplo, si dos características tienen el mismo valor de negocio la que requiera menor tiempo de desarrollo tendrá probablemente más prioridad, debido a que su ROI será más alto.

Sprint backlog

El sprint backlog es el subconjunto de requisitos que serán desarrollados durante el siguiente sprint. Al definir el sprint backlog, se describe el *cómo* el equipo va a implementar los requisitos durante el sprint. Por lo general los requisitos se subdividen en tareas, a las cuales se asignan ciertas *horas de trabajo* pero ninguna tarea con una duración superior a 16 horas. Si una tarea es mayor de 16 horas, deberá ser dividida en otras menores. Las tareas en el *sprint backlog* nunca son asignadas, son tomadas por los miembros del equipo del modo que les parezca adecuado.

Burn down chart

La burn down chart es una gráfica mostrada públicamente que mide la cantidad de requisitos en el Backlog del proyecto pendientes al comienzo de cada Sprint. Dibujando una línea que conecte los puntos de todos los Sprints completados, podremos ver el progreso del proyecto. Lo normal es que esta línea sea descendente (en casos en que todo va bien en el sentido de que los requisitos están bien definidos desde el principio y no varían nunca) hasta llegar al eje horizontal, momento en el cual el proyecto se ha terminado (no hay más requisitos pendientes de ser completados en el Backlog). Si durante el proceso se añaden nuevos requisitos la recta tendrá

pendiente ascendente en determinados segmentos y si se modifican algunos requisitos la pendiente variará o incluso valdrá cero en algunos tramos.

2.6. Análisis comparativo de Metodologías

Tabla 2

Comparación de Metodologías.

Metodología	Proceso	Representación grafica
UWE	Su proceso se adapta a los diagramas uml	Elementos de UML
XP	Planificación Diseño Codificación Pruebas	Tarea de Ingeniería(Task card)
SCRUM	Product Backlog Sprint Planning Sprint Sprint Backlog Daily sprint meeting Demo y retrospectiva	Sprint Sprint backlog
HDM	1.Desarrollo a largo plazo 2.Desarrollo a corto plazo	Diagrama E-R
RMM	1.Diseño E-R 2.Diseño Slice ¹³ 3.Diseño de navegación 4.Diseño de protocolo de conversión 5.Diseño de UI ¹⁴ 6.Diseño de comportamiento en tiempo de ejecución	1.Diagrama E-R 2.Diagrama Slice 3.Diagrama RMDM

	7.Prueba y construcción	
EORM	1.Clases del entorno de desarrollo 2.Composición del entorno de desarrollo 3.Entorno de desarrollo de UI	1.Diagrama de clases 2.Diseño GUI
OOHDM	1.Diseño conceptual 2.Diseño navegacional 3.Diseño abstracto de la UI 4.Implementación	1.Diagrama de clases 2.Diagrama navegacional, clase + contexto 3.Diagrama de configuración de ADV + Diagrama ADV

2.7. Metodología Seleccionada

En base al cuadro comparativo y el análisis de cada una de las metodologías se seleccionó la metodología SCRUM debido a las siguientes razones:

- Adaptabilidad a los cambios entre iteraciones.

La empresa novamusic no tiene claro sus requerimientos ya que es la primera vez que van a tener un software a medida, debido a esto es riesgoso tener una metodología q no se adapte a los cambios.

- Blindaje de cada iteración con respecto al cambio.

Para un desarrollo más ágil los requerimientos que sean aceptados en cada iteración no se cambian en la siguiente

- Resultados anticipados.

Los resultados del proyecto se visualizan en cada iteración de esta manera el cliente puede visualizar el sistema y sus avances.

- Mitigación de Riesgos

Al tener un feedback en cada sprint se mitigan los riesgos que pueden presentarse cuando el proyecto esté en producción.

CAPITULO III

ANÁLISIS Y DISEÑO DE LA APLICACIÓN

3.1. Captura de Requisitos

La presente especificación de requerimientos pertenece al desarrollo del Caso de Estudio " Análisis, Diseño e Implementación de una Aplicación Web para el control de bodegas en la empresa NOVAMUSIC mediante la utilización de la plataforma JAVA Enterprise Edition JEE7 web".

Introducción

Propósito

El propósito del presente apartado es definir los requerimientos que debe tener la una Aplicación Web para el control de bodegas en la empresa NOVAMUSIC a partir de ahora. Con la especificación de requerimientos se formalizará las funcionalidades de la aplicación junto al cliente.

Definiciones, Acrónimos y abreviaturas

SGBN.- Abreviatura para definir el nombre del Sistema: Sistema de Gestión de Bodegas Novamusic.

Java.- Corresponde al lenguaje de programación con el cual está hecho el sistema.

Postgres.- Corresponde a la base de datos utilizada para el desarrollo del sistema.

Birt.- Corresponde a la herramienta de Business Intelligence utilizada para el módulo de reportes

Eclipse.- Corresponde al IDE que se utilizó para el desarrollo de la aplicación.

Identificación de Roles y Tareas

1. Roles

Administrador

Es el usuario que tiene acceso a todas las opciones que nos permite realizar la aplicación.

Jefe de Bodega

Es el usuario que gestiona todas las acciones que se pueden realizar en la bodega.

Documentos Bodega:

Usuario que puede consultar los comprobantes de ingresos y salidas de Bodega.

Gestor de Bodega:

Usuario que accede a realizar las salidas de Bodega por consumo, traspaso y devolución a proveedor.

2. Tareas

Administrador

- Crear Usuarios.
- Asignar permisos a Usuarios.
- Ingresar Proveedores.
- Ingresar Secciones.
- Cambiar de sección un Producto.
- Ingresar un producto a la Bodega.
- Registrar la salida de un producto de la Bodega.
- Reservar un producto para un cliente.
- Anular un ingreso, salida o reserva de la Bodega.
- Ingresar devoluciones a la Bodega.
- Devolución a Proveedor.
- Consultar el inventario de Bodega.
- Consultar cada uno de los reportes disponibles.

Jefe de Bodega

- Ingresar Proveedores.
- Ingresar Secciones.
- Cambiar de sección un Producto.
- Ingresar un Producto a la Bodega.
- Salida de un producto de la Bodega.
- Reservar un producto.
- Anular un registro de la Bodega.
- Ingresar devoluciones a la Bodega.
- Devolución a Proveedor.
- Consultar el inventario de Bodega.
- Consultar cada uno de los reportes disponibles.

Documentos de Bodega

- Consultar el inventario de Bodega.
- Consultar cada uno de los reportes disponibles.

Gestor de Bodega

- Ingresar un Producto a la Bodega.
- Salida de un producto de la Bodega.
- Reservar un producto.
- Anular un registro de la Bodega.
- Ingresar devoluciones a la Bodega.
- Devolución a Proveedor.
- Consultar el inventario de Bodega.
- Consultar cada uno de los reportes disponibles.

Especificación de Escenarios

Rol Administrador

- **Crear Usuario:** El usuario podrá crear usuarios para que tengan acceso al sistema.
- **Asignar Permisos a Usuarios:** El usuario podrá asignar los permisos respectivos dependiendo del perfil de usuario.

- **Ingresar Proveedores:** El usuario podrá registrar los proveedores de la empresa.
- **Ingresar Secciones:** *El usuario podrá ingresar secciones donde se van a guardar los productos.*
- **Cambiar de sección un Producto:** El usuario podrá cambiar de sección (ubicación) a un producto.
- **Ingresar un producto a la bodega:** *El usuario podrá registrar un producto en la bodega.*
- **Salida de un producto de la bodega:** El usuario podrá registrar la salida de un producto de la bodega.
- **Reservar un producto:** El usuario podrá reservar un producto para venderlo a cliente.
- **Anular un registro de la Bodega:** El usuario podrá anular un ingreso, salida o reserva registrados.
- **Ingresar devoluciones a la Bodega:** El usuario podrá registrar cuando un producto es devuelto por un cliente.
- **Devolución a Proveedor:** El usuario podrá registrar cuando se realizan devoluciones de productos a los proveedores.
- **Consultar el inventario de Bodega:** El usuario podrá consultar todo el inventario que posee la bodega.
- **Consultar cada uno de los reportes disponibles:** EL usuario podrá consultar cada uno de los reportes realizados.

3.2. DEFINICION DE ACTORES

Los Actores que este sistema posee son únicamente los roles que tiene el sistema debido a que este software no se conecta con otros sistemas o subsistemas.

Los actores son los siguientes:

Administrador

Es el usuario que tiene acceso a todas las opciones que nos permite realizar la aplicación.

Jefe de Bodega

Es el usuario que gestiona todas las acciones que se pueden realizar en la bodega.

Documentos Bodega:

Usuario que puede consultar los comprobantes de ingresos y salidas de Bodega.

Gestor de Bodega:

Usuario que accede a realizar las salidas de Bodega por consumo, traspaso y devolución a proveedor.

3.3. Modelo de Casos de Uso

Especificación de casos de uso por Actor

ACTOR: ADMINISTRADOR

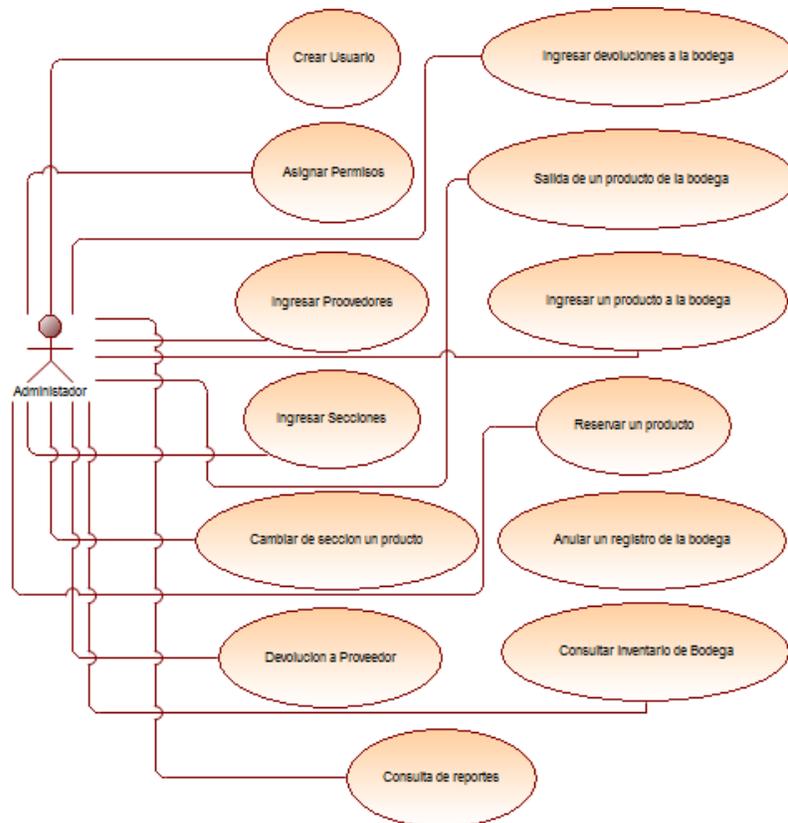


Figura 4 Caso de Uso para Administrador.

ACTOR: JEFE DE BODEGA

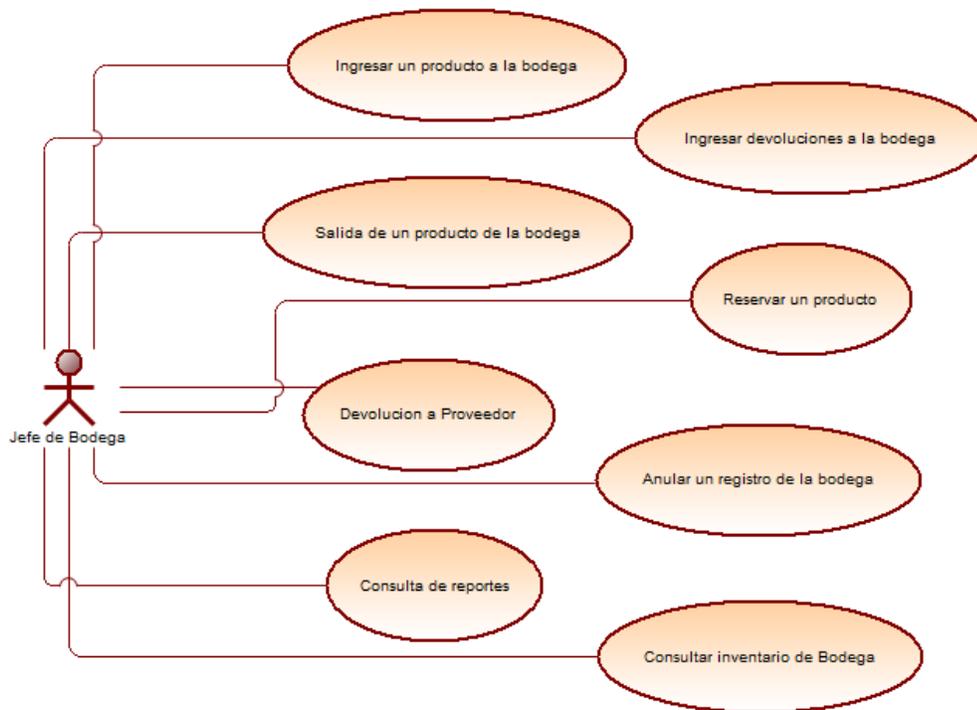


Figura 5 Casos de Uso Jefe de Bodega

ACTOR: DOCUMENTOS BODEGA

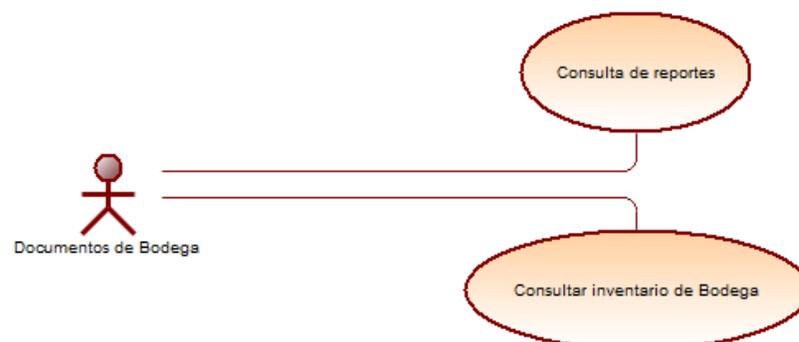
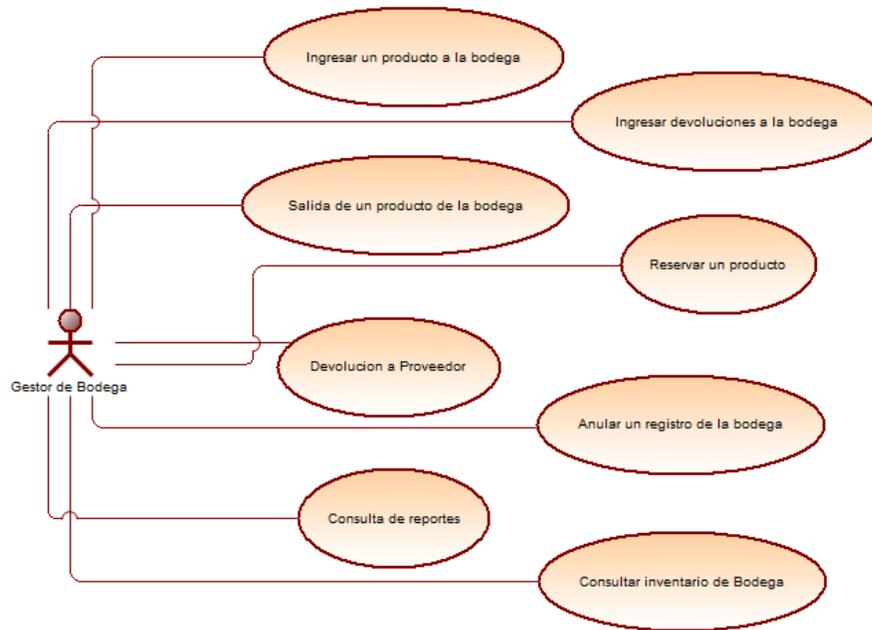


Figura 6 Casos de Uso Documentos Bodega

ACTOR: GESTOR DE BODEGA**Figura 7** Casos de Uso Gestor de Bodega

3.4. CASOS DE USO DETALLADOS

En la tablas siguientes de detallan las especificaciones de caso de uso del actor Administrador.

Tabla 3

Crear Usuario.

USR-ADM-CU-01: Crear Usuario	
Resumen:	Proceso en el cual el usuario administrador crea usuarios para que estos puedan acceder al sistema.
Prioridad:	Esencial
Actores Directos:	Administrador
Escenarios	
Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario administrador ingresar nuevos usuarios para q puedan tener acceso al sistema: el usuario debe llenar todos los datos del usuario para que el sistema pueda signarle una clave.
Pre-condiciones	
Ninguna	

Tabla 4

Asignar Permisos

USR-ADM-CU-02: Asignar permisos	
Resumen:	Proceso en el cual el usuario administrador asignar los permisos a determinado usuario.
Prioridad:	Esencial
Actores Directos:	Jugador
Escenarios	
Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario asignar los permisos a determinado usuario, dependiendo las funciones del mismo para de esta manera asignar roles a un usuario.

Continúa la Tabla

Pre-condiciones Continúa la Tabla

1. El usuario debe ingresar al sistema autenticado como administrador.
2. Deben existir los usuarios a los cuales se les va asignar permisos.

Tabla 5

Ingresar Proveedores.

USR-ADM-CU-03: Ingresar Proveedores

Resumen: **Proceso en el cual el usuario administrador crea proveedores**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario crear proveedores, los cuales abastecerán de productos a la empresa

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 6

Ingresar Secciones

USR-ADM-CU-04: Ingresar Secciones

Resumen: **Proceso en el cual el usuario administrador crea secciones**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Continúa la Tabla

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario crear secciones a las cuales se les van a signar productos, para de esta manera saber dónde se encuentran los productos.
Pre-condiciones	
1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.	

Tabla 7

Ingresar Producto a la bodega.

USR-ADM-CU-05: Ingresar un producto a la bodega

Resumen:	Proceso en el cual el usuario administrador registra productos en la bodega
Prioridad:	Esencial
Actores Directos:	Jugador
Escenarios	
Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario ingresar los productos a la bodega, asignando cada uno de sus características y donde se encuentra el mismo.
Pre-condiciones	
1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.	

Tabla 8

Cambiar de sección un producto.

 USR-ADM-CU-06: Cambiar de sección un producto

Resumen: **Proceso en el cual el usuario administrador puede cambiar la sección a la que está asignado un producto**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario cambiar la sección a la cual está asignado un producto, la sección es el lugar donde se guarda el producto.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 9

Devolución a proveedor

 USR-ADM-CU-07: Devolución a Proveedor

Resumen: **Proceso en el cual el usuario administrador puede registrar la devolución de un producto al proveedor**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario registrar cuando un producto se devolvió al proveedor.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 10

Devolución de un producto.

USR-ADM-CU-08: Devolución de un Producto

Resumen:	Proceso en el cual el usuario administrador puede registrar la devolución de un producto.
Prioridad:	Esencial
Actores Directos:	Jugador

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario registrar cuando un producto es devuelto por un cliente y regresa a la bodega.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 11

Salida de un producto de bodega.

USR-ADM-CU-09: Salida de un Producto de Bodega

Resumen:	Proceso en el cual el usuario administrador puede registrar la salida de un producto de la bodega.
Prioridad:	Esencial
Actores Directos:	Jugador

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario registrar cuando se vende un producto y este sale de la bodega.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 12

Reservar Producto.

USR-ADM-CU-10 Reservar Producto

Resumen: **Proceso en el cual el usuario administrador puede registrar la reserva de un producto.**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario registrar cuando se reserva un producto para un cliente

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 13

Anular un registro de bodega.

USR-ADM-CU-11 Anular un registro de bodega

Resumen: **Proceso en el cual el usuario administrador puede anular un registro de bodega.**

Prioridad: **Esencial**

Actores Directos: **Jugador**

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario puede anular una reserva, salida o ingreso de un producto a la bodega.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 14

Consultar le inventario

USR-ADM-CU-12 Consultar el inventario

Resumen:	Proceso en el cual el usuario administrador puede consultar el inventario de la bodega
Prioridad:	Esencial
Actores Directos:	Jugador

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario puede consultar todos los producto que se encuentran en la bodega.

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

Tabla 15

Consultar Reportes.

USR-ADM-CU-13 Consultar reportes

Resumen:	Proceso en el cual el usuario administrador puede consultar los reportes
Prioridad:	Esencial
Actores Directos:	Administrador

Escenarios

Tipo de Escenario	Descripción
Principal	1. La aplicación permite al usuario puede consultar todos los reportes que tiene el sistema

Pre-condiciones

1. El usuario debe ingresar al sistema autenticado como administrador o Jefe de Bodega.

3.4.1.Requerimientos No Funcionales

- El sistema debe poseer un esquema de perfiles de usuario, que garantice el acceso al sistema a cada uno, de tal manera que un usuario no registrado no tendrá acceso a todos los datos u operaciones disponibles.
- El sistema estará disponible en la red de la empresa, si los gerentes de la empresa lo disponen se subirá en un hosting en el internet.
- El sistema será un sistema web, por lo tanto será accesible para todo empleado en la red de la empresa, con usuario y permisos previamente asignados.

3.6. Artefactos propios de la metodología

Product Backlog

El siguiente documento desarrollado con la metodología scrum especifica los requerimientos del sistema, sus prioridades y la estimación en horas.

Los siguientes requerimientos son funcionalidades específicas del sistema

Prioridad: Normal – Estas funcionalidades se presentaran cada una en los diferentes sprints del proyecto.

- **Crear Usuario:** El usuario podrá crear usuarios para que tengan acceso al sistema. Horas: 8
- **Asignar Permisos a Usuarios:** El usuario podrá asignar los permisos respectivos dependiendo del perfil de usuario. Horas: 6
- **Ingresar Proveedores:** El usuario podrá registrar los proveedores de la empresa. Horas: 6
- **Ingresar Secciones:** *El usuario podrá ingresar secciones donde se van a guardar los productos.* Horas: 6
- **Cambiar de sección un Producto:** El usuario podrá cambiar de sección (ubicación) a un producto. Horas: 6
- **Ingresar un producto a la bodega:** *El usuario podrá registrar un producto en la bodega.* Horas: 8
- **Salida de un producto de la bodega:** El usuario podrá registrar la salida de un producto de la bodega. Horas: 6
- **Reservar un producto:** El usuario podrá reservar un producto para venderlo a cliente. Horas: 6
- **Anular un registro de la Bodega:** El usuario podrá anular un ingreso, salida o reserva registrados. Horas: 6
- **Ingresar devoluciones a la Bodega:** El usuario podrá registrar cuando un producto es devuelto por un cliente. Horas: 6

- **Devolución a Proveedor:** El usuario podrá registrar cuando se realizan devoluciones de productos a los proveedores. Horas: 8
- **Consultar el inventario de Bodega:** El usuario podrá consultar todo el inventario que posee la bodega. Horas: 8
- **Consultar cada uno de los reportes disponibles:** EL usuario podrá consultar cada uno de los reportes realizados. Horas: 12

Los siguientes requerimientos son generales y están enfocados en la gestión del sistema:

Prioridad: Alta – Deben estar listos para la implementación del sistema

- El sistema debe poseer un esquema de perfiles de usuario, que garantice el acceso al sistema a cada uno, de tal manera que un usuario no registrado no tendrá acceso a todos los datos u operaciones disponibles. Horas: 48
- El sistema estará disponible en la red de la empresa, si los gerentes de la empresa lo disponen se subirá en un hosting en el internet. Horas: 24
- El sistema será un sistema web, por lo tanto será accesible para todo empleado en la red de la empresa, con usuario y permisos previamente asignados. Horas: 24

Sprint Backlog

Tabla 16

Sprint 1

Numero:	Sprint 1
Tareas:	Obtención de requerimientos Análisis de requerimientos Diseño de la base de datos Diseño de la arquitectura Especificación de Requerimientos
Tiempo:	2 semanas
Recurso:	Tesista
Porcentaje a realizarse:	100%

Tabla 17

Sprint 2

Numero:	Sprint 2
Tareas:	Autenticación de usuarios en el sistema Formulario de creación de personas Formulario de creación de usuarios Formulario de asignación de permisos
Tiempo:	2 semanas
Recurso:	Tesista
Porcentaje a realizarse:	90%

Tabla 18

Sprint 3

Numero: Sprint 3

Tareas:	Formulario de ingreso de marcas Formulario de ingreso de secciones Formulario de ingreso de productos y eliminación de productos de la bodega.
---------	--

Tiempo: 2 semanas

Recurso: Tesista

Porcentaje a realizarse: 100%

Tabla 19

Sprint 4

Numero: Sprint 4

Tareas:	Formulario de administración de productos, reservas, anulación de reservas, ventas, anulación de ventas. Elaboración de reportes
---------	---

Tiempo: 2 semanas

Recurso: Tesista

Porcentaje a realizarse: 100%

BURNDOWN CHART

Tabla 20

Burn Down Chart

Sprints		Sprint 1				Sprint 2				Sprint 3 / Semana				Sprint 4		
Tarea / Semana	Horas	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Obtención de requerimientos	24	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Análisis de requerimientos	12	4	8	0	0	0	0	0	0	0	0	0	0	0	0	0
Diseño de la base de datos	14	0	5	9	0	0	0	0	0	0	0	0	0	0	0	0
Diseño de la arquitectura	12	0	0	4	8	0	0	0	0	0	0	0	0	0	0	0
Especificación de Requerimientos	10	0	0	0	2	8	0	0	0	0	0	0	0	0	0	0
Autenticación de usuarios en el sistema	10	0	0	0	0	2	8	0	0	0	0	0	0	0	0	0
Formulario de creación de personas	9	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0
Formulario de creación de usuarios	8	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
Formulario de asignación de permisos	9	0	0	0	0	0	0	0	4	5	0	0	0	0	0	0
Formulario de ingreso de marcas	8	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0
Formulario de ingreso de secciones	9	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0
Formulario de ingreso de productos y eliminación de productos de la bodega.	16	0	0	0	0	0	0	0	3	0	0	12	4	0	0	0
Formulario de administración de productos.	18	0	0	0	0	0	0	0	0	0	0	0	6	14	4	0
Elaboración de reportes	7	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0
Entrega de Proyecto	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
Horas estimadas	190	11	22	33	44	55	66	77	88	99	110	121	132	143	154	165
Horas Realizadas	190	12	13	13	10	10	8	9	15	13	9	12	10	14	11	24

GRAFICO DE PROGRESO DEL PROYECTO

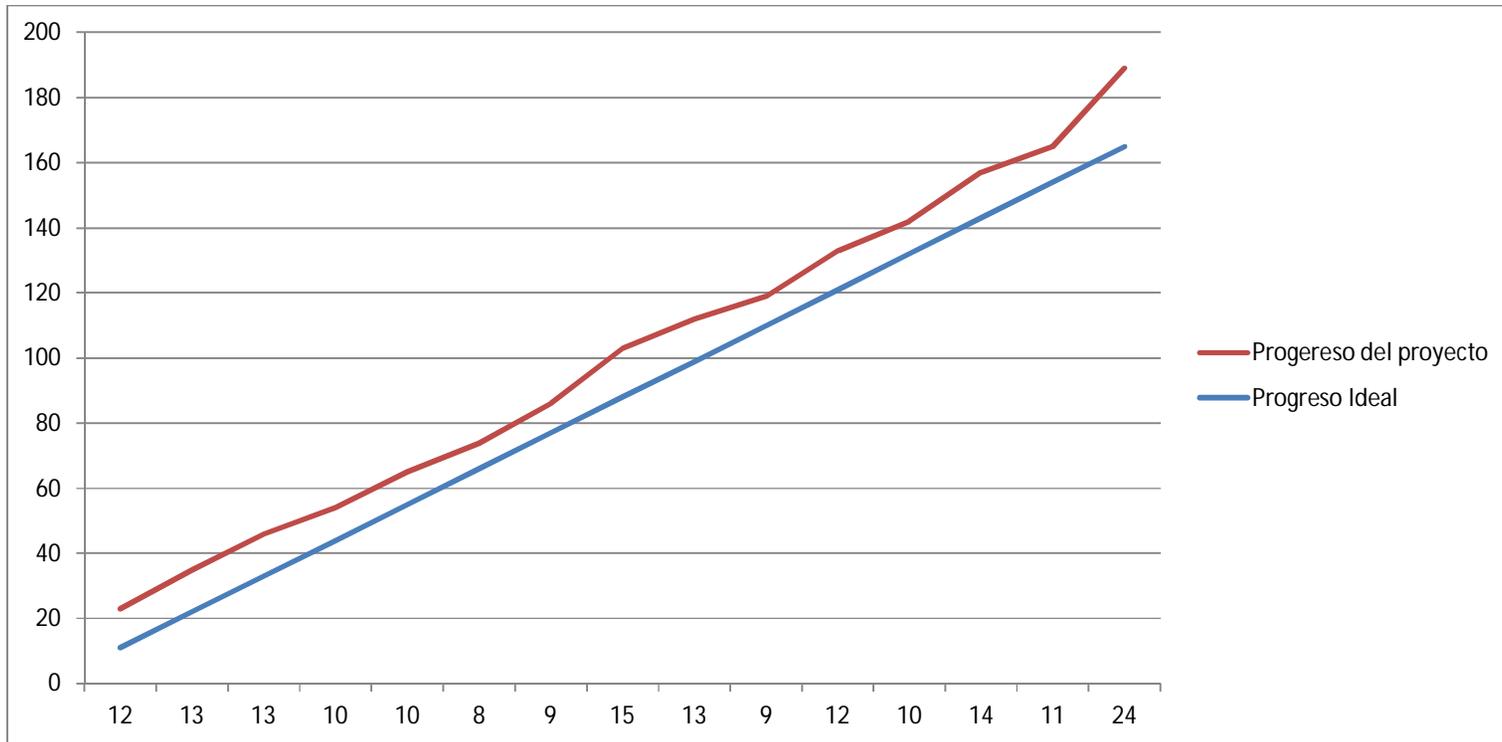


Figura 9 Proceso del Proyecto

CAPITULO IV

DESARROLLO

4.1. Arquitectura a Implementarse

4.1.1.Arquitectura Web

Una aplicación web es proporcionada por un servidor web, en nuestro caso WildFly el cual se pueden conectar varios clientes mediante un navegador. La arquitectura de una página web tiene tres componentes principales:

- Un servidor Web

- Una conexión de red

- Uno o más clientes

El servidor Web distribuye páginas de información formateada a los clientes que las solicitan. Los requerimientos son hechos a través de una conexión de red y para ello se usa el protocolo HTTP. Una vez que se solicita esta petición mediante el protocolo HTTP y la recibe el servidor Web, éste localiza la página Web en su sistema de archivos y la envía de vuelta al navegador que la solicitó.

4.1.2.Modelo de N Capas

Este modelo está diseñado para superar las limitaciones de las arquitecturas ajustadas a los modelos de menos capas, introduce una capas intermedias (en esa aplicación de bodegas se utilizara capas de proceso y validación) entre presentación y los datos, los procesos pueden ser manejados de forma separada a la interfaz de usuario y a los datos, esta capa intermedia centraliza la lógica de negocio, haciendo la administración más sencilla, las capas del modelo que se va a utilizar en el sistema de gestión de bodegas son:

Capa de presentación.-parte en el cliente y parte en el servidor; los procesos de esta capa los realiza el navegador

Recoge la información del usuario y la envía al servidor (cliente)

Manda información a la capa de proceso para su procesamiento

Recibe los resultados de la capa de proceso

Generan la presentación

Visualizan la presentación al usuario (cliente)

Capa de proceso.-Estos procesos son netamente del servidor web, en esta aplicación WildFly

Recibe la entrada de datos de la capa de presentación

Interactúa con la capa de datos para realizar operaciones

Manda los resultados procesados a la capa de presentación

Capa de Validaciones, Excepciones.- Esta capa está en java, se encarga de comunicarse entre el lenguaje de programación y la base de Datos.

Se comunica con la base de datos para poder obtener datos que son validados o procesados en java.

Comunicación entre el reporteador (Birt) y la base de datos.

Capa de datos.- (servidor de datos)

Almacena los datos

Recupera datos

Mantiene los datos

Asegura la integridad de los datos

4.2. Estándares de implementación

4.2.1. Estandar J2EE.

La plataforma de java, Enterprise Edition(J2EE) define el estándar para desarrollar componentes empresariales orientados a objetos, este estándar

simplifica la construcción de aplicaciones empresariales estables, escalables y que se integran fácilmente datos y aplicaciones heredadas.

En contexto un ambiente de servicios de aplicaciones distribuidas se compone de:

- Runtime hosting Applications(Servidor de aplicaciones.)
- Java Database Connectivity (jdbc)
- Remote Method Invocation(RMI)
- Enterprise Java Beans (EJB)
- Java Server Pages (JSP)
- Java Message Service(JMS)
- Java Naming Directory Interface(JNDI)
- Java Transaction API
- Java Mail
- Estándar Java API

4.2.2. ESTANDARES DE CODIFICACION

Los estándares utilizados en este proyecto de tesis son los estándares internacionales de desarrollo de software, buenas prácticas, etc.

A continuación se va a detallar los estándares utilizados:

NOMENCLATURA –Paquetes

- El paquete base está definido como com.novamusic, los nombres de todos los paquetes están escritos en minúsculas y sin caracteres especiales

NOMENCLATURA –Interfaces

- Todos los nombres de los interfaces están escritos en formato CamelCase
- No se usan abreviaciones que dificultan la comprensión del código

NOMENCLATURA –Clases

- Los nombres están escritos en formato CamelCase
- Los nombres son simples y descriptivos.
- Se usan palabras completas sin acrónimos y abreviaturas.

NOMENCLATURA –Métodos

- Los métodos son verbos en infinitivo.
- Están en formato lowerCamelCase.
- No contienen caracteres especiales.
- Los nombres son suficientemente descriptivos.

NOMENCLATURA –Variables

- Están en formato lowerCamelCase (no aplicable a variables finales)
- No contienen caracteres especiales
- Los nombres son suficientemente descriptivos.

ESTILO DE CODIFICACIÓN –Comentarios

- No se hace un uso abusivo de ellos
- Se evita el uso de caracteres especiales y “gráficos” en ASCII

ESTILO DE CODIFICACIÓN –Declaraciones

- Se evita la duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase.

ESTILO DE CODIFICACIÓN –Sentencias

- Todo bloque de sentencias se encuentra entre llaves.
- La variable de avance del bucle nunca es modificada dentro del propio bucle.

BUENAS PRÁCTICAS –Propiedades

- El acceso/modificación de las propiedades de una clase (no constantes) siempre mediante métodos de acceso get/set.

BUENAS PRÁCTICAS –Métodos

- No se accede a un método estático desde una instancia de una clase.

4.3. Desarrollo de los módulos del sistema

Módulo de Inventario

Este módulo se encarga de gestionar el flujo de la bodega, posee las siguientes funcionalidades:

- Cambiar de sección un Producto.
- Ingresar un producto a la Bodega.
- Registrar la salida de un producto de la Bodega.
- Reservar un producto para un cliente.
- Anular un ingreso, salida o reserva de la Bodega.
- Ingresar devoluciones a la Bodega.
- Devolución a Proveedor.
- Consultar el inventario de Bodega.

Módulo de Administración

Este módulo se encarga de gestionar permisos y administrar el sistema

- Crear Usuarios.
- Asignar permisos a Usuarios.

Módulo de Reportes

- Este módulo contiene reportes solicitados por el usuario final.

4.4. INTEGRACION DEL CÓDIGO

La integración del código en este proyecto se realiza en la puesta a producción del sistema de gestión de bodegas.

- Creación de los documentos y archivos entregables
- Creación de la base de datos.
- Creación del archivo ear del proyecto.
- Creación de manuales del usuario y manual técnico.\

4.5. PRUEBAS

Actividades realizadas luego de terminar las iteraciones.

- Se levantó un ambiente de pruebas de la aplicación.
- Se valida que el sistema realice todo lo que está especificado en el levantamiento de requerimientos.
- Se realizan los procesos de cada perfil de usuario, haciendo que estos prueben la aplicación por un periodo de 3 días.
- Se analiza cuáles son los reportes necesarios para que cada perfil de usuario, debido a que se eligió desarrollar este sistema con la metodología scrum se puede cambiar los requerimientos en cada iteración.
- Se realiza la corrección de los errores (bugs) en el sistema.
- Se verifica la corrección de los errores volviendo a levantar un ambiente de pruebas.
- Se firma una carta de aceptación del sistema por parte de la empresa NOVAMUSIC.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- El estándar J2EE facilitó los componentes necesarios para poder realizar una aplicación Web Java que gestiona todo el proceso de Bodegas en la empresa NOVAMUSIC.
- El uso de la metodología SCRUM proyecto agilizo el proceso de desarrollo del sistema, ya que la empresa NOVAMUSIC no tenia en claro los requerimiento, las iteraciones de SCRUM permitieron modificar los requerimientos para poder cumplir con las necesidades del cliente.
- La herramienta de Bussiness Intelligence BIRT permitió realizar el módulo de reportes, mediante la utilización de POJO Data Sources se modelo cada uno de los reportes del proyecto, que utiliza una clase intermedia para poder conectar el reporte y los servicios expuestos.
- A través del uso de EJB una API del estándar J2EE se realizó las transacciones, control de concurrencia y seguridades en el sistema de gestión de bodegas, esta API permitió simplicidad, portabilidad, reutilización de componentes.
- A través del uso de JPA e HIBERNATE parte del estándar J2EE se configuro la comunicación entre la base de datos y los servicios locales, estos estándares permiten conectar la aplicación a la base de datos mediante un solo dialecto.

5.2. Recomendaciones

- Para el adecuado funcionamiento y desarrollo de un software se recomienda, respetar los estándares y buenas prácticas del estándar J2EE.
- Utilizar una metodología incremental para el desarrollo de un proyecto en el cual el cliente no tenga claro sus requerimientos.
- Seleccionar de forma adecuada una herramienta de reportes según la necesidad del cliente.
- Escoger adecuadamente como se va a exponer servicios y consumir los mismos, según la complejidad de la aplicación.
- Se recomienda modelar la base de datos escalable, siempre pensando en que los datos y la complejidad van a aumentar.

REFERENCIAS BIBLIOGRAFICAS

Bibliografía

- Albahari, B., Drayton, P., & Merrill, B. (2002). *C# Essentials*. O'Reilly Media, Inc.
- Aleixos Borrás, N., Piquer Vicent, A., Galmes Gual, V., & Company Calleja, P. (Junio de 2002). ESTUDIO COMPARATIVO DE APLICACIONES CAD DE MODELADO.
- Alfonso Galipienso, M. I., Cazorla Quevedo, M. A., Colomina Pardo, O., Escolano Ruiz, F., & Lozano Ortega, M. A. (2003). *Inteligencia artificial: modelos, técnicas y áreas de aplicación*. Paraninfo.
- Autodesk, Inc. (2013). *Autodesk, Inc.* Recuperado el 17 de Junio de 2013, de 2013 Autodesk, Inc.: <http://www.autodesk.es/products/autodesk-maya/overview>
- Balerdi, F. E. (2009). *Gredos, Universidad de Salamanca*. Recuperado el 2 de Agosto de 2013, de http://gredos.usal.es/jspui/bitstream/10366/56438/1/TEE2001_V2_videojuegoseducacionpdf.pdf
- Caccuri, V. (2013). *Educación con TICS*. USERSHOP.
- Creighton, R. H. (2010). *Unity 3D Game Development by Example: A Seat-of-Your-Pants Manual for Building Fun, Groovy Little Games Quickly*. Packt Publishing Ltd.
- Cruz Zeballos, M. A. (2012). Las APP en la Nube. *Revista de Información, Tecnología y Sociedad*(7), 53-56.
- ECLIPSE. (1 de 1 de 2014). *BIRT ECLIPSE*. Obtenido de <http://www.eclipse.org/birt/img/about/birtarch.png>
- Exit Games. (2013). *Exit Games®*. Recuperado el 17 de Junio de 2013, de Exit Games®: <http://cloud.exitgames.com/Unity>
- García Fernandez, F. (2005). Videojuegos: un análisis desde el punto de vista educativo.
- García Fernández, F., & Bringué Sala, X. (2010). Educar hij@s Interactiv@s: Una reflexión práctica sobre las pantallas. *Foro Generaciones Interactivas*, 68.
- Gargenta, M. (2011). *Learning Android*. O'Reilly Media, Inc.
- González-Calero, P. A., & Gómez-Martin, M. A. (2011). *Artificial Intelligence for Computer Games*. Springer.
- Greenfield, P. M. (1996). Video games as cultural artifacts. *P.M. Greenfield & R. R. Cocking (Eds)*, 35-46.

- Inergis. (2012). *Inergis*. Recuperado el 18 de Febrero de 2014, de http://www.inergis.com/contenidos/general.action?idsupersection=2&idselectedsection=24&selectedsection=%BFPor%20qu%E9%20el%20Cloud%20Computing?&idparentmenu=146&idpage=44&idcomission=0&typetable=oferta_estudios
- Inteligencia Computacional. (2013). *Inteligencia Computacional*. Recuperado el 15 de Agosto de 2013, de Inteligencia Computacional: <http://www.smartcomputing.com.ar/capitulo6.aspx>
- JuegosMMO.net. (18 de Junio de 2013). *JuegosMMO.net*. Obtenido de JuegosMMO.net: <http://juegosmmo.net/juegos-mmo/>
- Mcfarlane , A., Sparrowhawk, A., & Heald , Y. (2002). Report on the educational use of games.
- Mcfarlane, A., Sparrowhawk, A., & Heald, Y. (2002). Report on the educational use of games.
- Mendoza Barros, P., & Galvis Panqueva, A. (1998). JUEGOS MULTIPLAYER: JUEGOS COLABORATIVOS .
- Mendoza Barros, P., & Galvis Panqueva, A. (1998). *JUEGOS MULTIPLAYER: JUEGOS COLABORATIVOS PARA LA EDUCACIÓN*.
- Millington, I., & Funge, J. (2009). *Artificial Intelligence for Games*. CRC Press.
- Orchilles, J. (2010). *Microsoft Windows 7 Administrator's Reference: Dive Deeper: Upgrading, Deploying, Managing, and Securing*. Syngress.
- Padilla Zea, N., González Sánchez, J. L., Gutiérrez, F. L., Cabrera, M. J., & Paderewski, P. (2008). Diseño de Videojuegos Educativos Multijugador. Una Visión desde el Aprendizaje Colaborativo. España. Recuperado el 5 de Diciembre de 2013, de Diseño de videojuegos multijugador. Una visión desde el aprendizaje colaborativo: <http://lsi.ugr.es/juegos/articulos/interaccion08-colaboracion.pdf>
- Pino Díez, R., Gómez Gómez, A., & de Abajo Martínez, N. (2001). *Introducción a la inteligencia artificial: sistemas expertos, redes neuronales artificiales y computación evolutiva*. Universidad de Oviedo.
- Pogue, D. (2011). *Mac OS X Lion: The Missing Manual*. O'Reilly Media, Inc.
- Prensky, M. (2001). *Digital Natives, Digital Immigrants*.
- Pressman, R. S. (2006). *Ingeniería del Software un enfoque práctico* (Sexta ed.). Madrid, España: McGraw Hill.
- Rollings, A., & Morris, D. (2003). *Game Architecture and Design*.
- Schwabe, D., & Rossi, G. (1998). Developing Hypermedia Applications using OOHDM.

scrumespanol. (01 de 09 de 2016). Obtenido de *scrumespanol*:

<https://scrumespanol.files.wordpress.com/2015/09/diagrama-proceso-scrum.gif?w=553&h=414>

Singer, R. N., & Williams, A. M. (1998). *An exploratory study in live tennis* .

Soto De Giorgis, R., Palma Muñoz, W., & Roncagliolo De La Horra, S. (2011). *Propuesta de un modelo navegacional para el desarrollo de aplicaciones basadas en OOHDM*.

Recuperado el 30 de Junio de 2013, de

<http://tallerinf281.wikispaces.com/file/view/Aplicacion-OOHDM.pdf>

Star UML. (2013). *Star UML*. Recuperado el 17 de Junio de 2013, de Star UML:

<http://staruml.sourceforge.net/en/about.php>