Universidad de las Ciencias Informáticas

FACULTAD 7

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Título: Desarrollo del componente de predicción de consumo de productos para el Sistema de Información Hospitalaria alas HIS.

Autores: Zahily Peña García

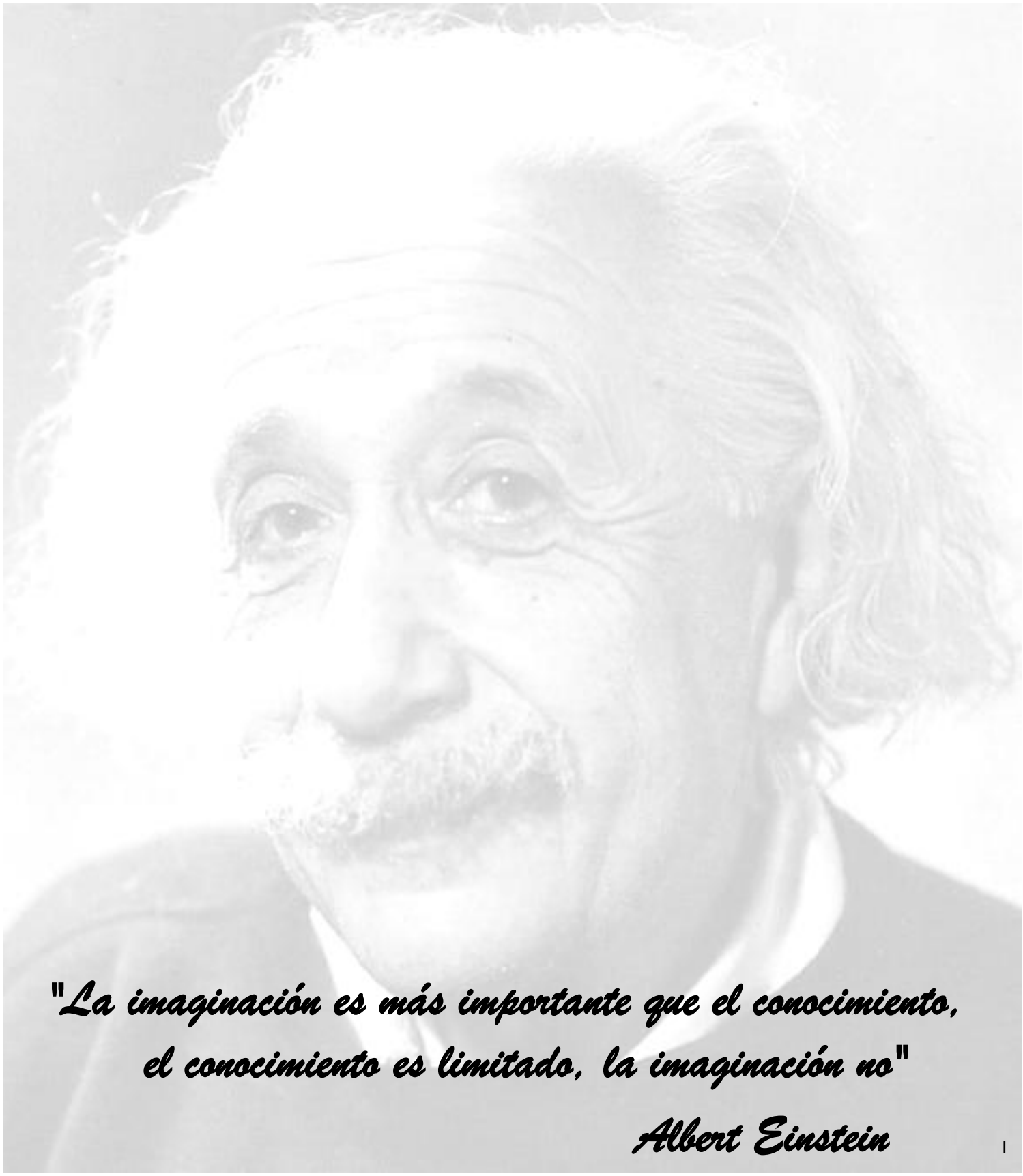
Geidar Soler Izquierdo

Tutor: Ing. Yeinier Ferrás Cecilio

Co-tutor: Ing. Leitniz Pérez Buján

La Habana, Junio de 2013

"Año 55 de la Revolución"



"La imaginación es más importante que el conocimiento, el conocimiento es limitado, la imaginación no"

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Zahily Peña García	Geidar Soler Izquierdo
Autora	Autor
_____	_____
Ing. Yeinier Ferrás Cecilio	Ing. Leitniz Pérez Buján
Tutor	Co-tutor

DATOS DE CONTACTO

Síntesis de los tutores:

Ing. Yeinier Ferrás Cecilio: Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2007. Posee la categoría docente de Profesor Instructor. Ha impartido asignaturas de Práctica Profesional, Programación Web, Gráficos por Computadora y Arquitectura de Computadoras. Ha participado en varios proyectos de desarrollo vinculados al perfil de salud. Actualmente labora en el Departamento de Programación como profesor.

Correo Electrónico: yferras@uci.cu

Ing. Leitniz Pérez Buján: Ingeniero en Ciencias Informáticas, graduado en el año 2012 en la Universidad de las Ciencias Informáticas. Es recién Graduado en Adiestramiento, ha impartido la asignatura de Gráficos por Computadora como alumno ayudante. Actualmente labora en el Departamento de Sistemas de Gestión Hospitalaria del Centro de Informática Médica (CESIM).

Correo Electrónico: lbujan@uci.cu

DEDICATORIA

De Zahily:

A mi hermano.

A mi esposo.

A mis padres.

A mis suegros.

A mis abuelos.

De Geidar:

A mis padres.

A mi hermana.

AGRADECIMIENTOS

Queremos agradecer a los profesores que en algún momento nos dieron clases, por ser parte imprescindible de nuestra formación. A los profes del proyecto, en especial a Dunia, por ayudarnos tanto. A nuestros tutores y a nuestros compañeros del grupo 7501.

De Zahily:

Quiero agradecer a todas las personas que de una forma u otra han estado presentes en el transcurso de estos cinco años, que tantas alegrías y tristezas me han dado.

A mi esposo y amigo Fidelky, por ser el mejor, por quererme y permitirme estar en su vida, por estar conmigo en las buenas y malas. Por confiar en mí y darme fuerzas para seguir adelante.

A mi mamá Elvira, por ser la mejor del mundo, por haber hecho de mí una mujer de bien, por enseñarme lo bueno y malo de la vida. Por poner el bienestar de sus hijos antes que nada.

A mi papá Osmany, por quererme y apoyarme en cada decisión que he tomado en mi vida.

A mi hermano Yosvany, por quererme, por su alegría, por ser tal y como es.

A mis abuelos maravillosos Bertha, Betty y Peña, por todo su amor y confianza.

A mis queridos abuelos Orlando y Marina, por estar presentes en cada momento de mi vida, por darme fuerzas para seguir adelante, pase lo que pase.

A mis suegros Loyda y Jose, por permitirme ser parte de su familia y acogerme como su propia hija.

A mis primas, primos, tías y tíos, en especial a Jorge, por ayudarme tanto y ser parte de este sueño.

A mis amigas y amigos de Holguín: María, Olianka, Ismary, Dianelis, Lorena y Yunior, por no dejar que el estar lejos rompa lo que nos une.

A los nuevos amigos y a mis compañeros del viejo grupo.

A mi compañero de tesis Geidar, por soportar mis regaños y a veces mi mal humor.

De Geidar:

Quiero agradecer a las personas que me han dado su apoyo incondicional durante estos cinco años.

A mis padres Jose y Maritza, por el apoyo que siempre me han brindado.

A mi hermana Sarisleidys.

A todos mis familiares, principalmente a mis primos, en especial a Nelson.

A todos mis amigos, que se han convertido en hermanos, especialmente a Alexei, Noel, Jorge y Adrián.

Al piquete molador, gracias a ustedes pasé un mejor año, en especial a Elier, Dayron, Roy, Russo, Carlos, Humberto, Nilier, siempre los tendré presentes.

A mi compañera de tesis Zahily, por soportarme.

RESUMEN

El Centro de Informática Médica (CESIM) desarrolló el Sistema de Información Hospitalaria alas HIS. El mismo posee un conjunto de módulos para la gestión de la información, entre los que se encuentra Almacén. Dicho módulo permite realizar actividades que son de gran importancia para las instituciones hospitalarias. Registra la existencia de los productos almacenados, envía solicitudes de productos a entidades legales, registra los despachos realizados, entre otros. Sin embargo, actualmente en los almacenes hospitalarios estos procesos se realizan manualmente, corriéndose el riesgo de cometer errores, lo que hace deficiente su desarrollo. No se emplean métodos matemáticos o estadísticos para encontrar patrones de comportamiento en información histórica almacenada, por lo que no puede conocerse con anterioridad qué cantidad de productos podrían consumirse en el futuro dentro del hospital.

En este trabajo el objetivo fue desarrollar un componente dentro del módulo Almacén del sistema alas HIS, que permita realizar la predicción del consumo de los productos existentes en los almacenes de las instituciones hospitalarias. Para efectuar la predicción se utilizó la técnica Series Temporales, de Minería de Datos. El desarrollo del componente estuvo guiado por el Proceso Unificado de Desarrollo de Software (RUP). Se utilizó Java como lenguaje de programación, Visual Paradigm for UML para el modelado, Eclipse como herramienta de desarrollo, JBoss Server como servidor de aplicaciones, PostgreSQL como Sistema Gestor de Base de Datos, iReport para el diseño visual de los reportes y JasperReports para la generación de los mismos. El componente obtenido facilita el proceso de toma de decisiones, dentro del área de almacén de las instituciones hospitalarias. Permite conocer un aproximado del presupuesto necesario para pagar los pedidos a realizar y planificar correctamente las solicitudes a los proveedores, de forma tal que puedan satisfacerse todas las necesidades de la institución.

PALABRAS CLAVE: consumo, minería de datos, predicción, productos, series temporales.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

El desarrollo científico y tecnológico que se ha desatado durante las últimas décadas ha permitido que la humanidad progrese rápidamente. Se han creado Sistemas de Gestión de la Información, que facilitan la toma de decisiones y que pueden utilizarse en diversas ramas de la sociedad, en dependencia de la información que manejen.

En el sector de la salud, los primeros Sistemas de Gestión de la Información aparecieron en la década de los setenta, permitiendo perfeccionar la calidad de los servicios prestados en aquel entonces. Con el tiempo, estos sistemas evolucionaron y dieron lugar a otros con características superiores. Un claro ejemplo de ello son los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés). Estos están orientados a satisfacer las necesidades de generación de información, para almacenar, procesar y reinterpretar datos médicos-administrativos de cualquier institución hospitalaria, (1).

El Sistema Nacional de Salud (SNS) en Cuba apuesta por los cambios y las ventajas que sugiere el uso de estos y otros sistemas. Por tal motivo, han surgido disímiles centros de innovación y desarrollo, uno de ellos es el Centro de Informática Médica (CESIM), que se encuentra en la Universidad de las Ciencias Informáticas (UCI). La misión de este centro es desarrollar productos informáticos para la optimización del trabajo y el mejoramiento de la calidad de la atención médica. (2)

Uno de los productos desarrollados por el CESIM es alas HIS, el cual está concebido para llevar el control de las actividades de salud orientadas a los pacientes, permitiendo además gestionar y controlar los recursos en cada una de las áreas de las instituciones hospitalarias, (3). Pone a disposición del usuario un conjunto de módulos para la gestión de la información, entre los que se encuentra Almacén.

El módulo Almacén gestiona las actividades en las diferentes áreas del hospital en las cuales se controla la entrada y salida de los productos existentes. Posibilita el registro de existencia de los productos almacenados, así como el envío de solicitudes de productos a otros almacenes del hospital y a entidades legales, esta última con el fin de buscar proveedores externos. Posibilita además, el despacho de productos, llevando de forma minuciosa la información referente a ello. Registra, mediante la información almacenada de los despachos realizados, el consumo de medicamentos y material quirúrgico, así como el de los equipos distribuidos hacia las diversas áreas de la institución. El sistema cuenta con la flexibilidad

necesaria para configurar el flujo de los procesos mencionados, permitiendo que se adecue a las necesidades de cada entidad, (4).

Para cualquier institución hospitalaria, la correcta realización de estos procesos es de gran importancia, ya que puede conocerse la cantidad de productos almacenados, logrando hacer frente a la demanda de estos. Permite el envío de solicitudes con suficiente tiempo a los proveedores, posibilitando que el almacén sea resurtido antes de que se agoten las existencias. Se puede además, conocer la cantidad de productos que se consumen a diario dentro de la institución, con lo cual puede facilitarse la planificación de los pedidos a realizar.

Sin embargo, la mayoría de estos procesos se realizan de forma manual, corriéndose el riesgo de cometer errores, lo que hace deficiente su desarrollo. No se emplean métodos matemáticos o estadísticos, para encontrar patrones de comportamiento en información histórica almacenada. Por tal motivo, no puede conocerse con anterioridad qué cantidad de productos podrían consumirse en el futuro dentro del hospital.

Esto trae como consecuencia que se dificulte el proceso de toma de decisiones por parte de la gerencia o directivos del área de almacén en las instituciones hospitalarias. No se planifican adecuadamente los pedidos a realizar a los proveedores y no puede conocerse el presupuesto necesario para costear dichos pedidos.

De no planificarse correctamente los pedidos a los proveedores, es probable que no se reciban de estos la cantidad adecuada de productos, corriéndose el riesgo de no poder satisfacer todas las solicitudes realizadas al almacén, desde otras áreas del hospital. Así mismo, puede existir incapacidad a la hora de proveer en situaciones de urgencia. De esta forma, el funcionamiento de la atención médica es deficiente.

Basado en lo antes expuesto se define como **problema a resolver** ¿Cómo predecir el consumo de productos dentro del módulo Almacén del Sistema de Información Hospitalaria alas HIS?

En correspondencia con el problema el **objeto de estudio** trazado lo constituye el proceso de predicción de consumo de productos en los almacenes de las instituciones hospitalarias. El **campo de acción** se centra en el proceso de predicción de consumo de productos en el módulo Almacén del sistema alas HIS.

Para la solución del problema se plantea como **objetivo general:** Desarrollar un componente dentro del módulo Almacén del sistema alas HIS, que permita la predicción del consumo de los productos existentes en los almacenes de las instituciones hospitalarias.

Para dar cumplimiento al objetivo se proponen las siguientes **tareas de la investigación:**

- Analizar sistemas informáticos vinculados al área de almacén de las instituciones hospitalarias, determinando que métodos de predicción de consumo de productos utilizan, de modo que pueda elegirse el adecuado para el desarrollo del componente.
- Definir los conceptos básicos relacionados con el ámbito del componente, así como las funcionalidades asociadas al mismo.
- Elaborar los artefactos correspondientes al flujo de trabajo Análisis y diseño, mediante el Proceso Unificado de Desarrollo de Software.
- Implementar las funcionalidades requeridas en el componente de predicción de consumo de productos para el sistema alas HIS.

Con el desarrollo de un componente, dentro del módulo Almacén del sistema alas HIS, que permita predecir el consumo de los productos existentes en los almacenes de las instituciones hospitalarias, se espera obtener los siguientes beneficios:

- Facilitar el proceso de toma de decisiones, dentro del área de almacén de las instituciones hospitalarias.
- Conocer un aproximado del presupuesto necesario para pagar los pedidos a realizar.
- Planificar correctamente las solicitudes a los proveedores, de forma tal que puedan satisfacerse todas las necesidades de la institución.

El presente trabajo está estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica: se analizan varios sistemas informáticos que están vinculados al área de almacén de las instituciones hospitalarias. Se exponen los conceptos asociados a la investigación, como: predicción y almacén hospitalario. Además se incluyen métodos y procedimientos para llevar a cabo la predicción. Se realiza un estudio de la metodología, las herramientas y tecnologías a utilizar para el desarrollo del componente.

Capítulo 2. Características del sistema: se describen los conceptos básicos relacionados con el ámbito del componente, con los cuales se elabora el modelo de dominio. Se realiza la especificación de los requisitos de software y se definen los casos de uso.

Capítulo3. Análisis y diseño del sistema: se describe la arquitectura. Se elaboran los diagramas de paquetes, clases web e interacción. Se describen algunas de las clases identificadas en el diseño para su posterior implementación y se exponen las estrategias de reutilización empleadas en el desarrollo del componente de predicción.

Capítulo 4. Implementación: se elabora el modelo de datos y se describen cada una de las tablas que lo integran. Se obtienen los diagramas de componentes y despliegue. Se reflejan las estrategias de codificación, estándares y estilos de código a utilizar.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo contiene los aspectos principales que fundamentan de forma teórica el componente de predicción propuesto. Se realiza un análisis de diversos sistemas informáticos vinculados al área de almacén de las instituciones de salud. Se exponen los conceptos asociados a la investigación, incluyendo las técnicas y procedimientos para llevar a cabo la predicción. Además se describen la metodología, las tecnologías y herramientas de software definidas por el departamento Sistemas de Gestión Hospitalaria, para el proceso de desarrollo de sus productos.

1.1 Sistemas informáticos vinculados al área de almacén de las instituciones hospitalarias

Actualmente existe a nivel mundial un gran número de productos informáticos destinados a las instituciones hospitalarias, con el objetivo de mejorar la calidad de la atención médica y la administración de los recursos que allí se encuentran. Como parte de la investigación realizada para el desarrollo del componente en cuestión se han estudiado varios sistemas, los cuales están vinculados al área de almacén de las instituciones hospitalarias.

CNT PACIENTES

Sistema de Gestión Hospitalaria desarrollado por la empresa CNT Sistemas de Información S.A., tiene como objetivo satisfacer las necesidades de monitoreo y control de los servicios de clínicas y hospitales. Es una aplicación de escritorio confiable, escalable y flexible, aunque no es multiplataforma. Administra e integra eficazmente la totalidad de la información asistencial, administrativa y financiera de la institución hospitalaria y se compone de 23 módulos. (5)

Uno de los módulos se encarga de las actividades realizadas en la farmacia y en el almacén. Este permite registrar órdenes de compras, entradas y salidas, crear diferentes niveles de sub stock o dependencias, administrar el stock del depósito, realizar ajustes e informar que ítems se encuentran por debajo del nivel mínimo.

ASSIST

Sistema de Información desarrollado por TCA Software Solutions, permite manejar la operación, administración y planeación estratégica de una institución hospitalaria pública o privada. Presenta en todo momento información confiable y oportuna para la toma de decisiones y el control de las operaciones. (6)

Contempla la gestión y control en diferentes áreas, entre las que se encuentra la de abastecimientos. Esta es la encargada de la administración y surtido de medicamentos y materiales. Permite localizar cualquier artículo o producto dentro del almacén, manteniendo varios atributos de identificación para estos. Se pueden realizar reportes de los movimientos realizados en un período determinado, niveles de existencia del inventario, qué productos se deben resurtir y en qué cantidad; además muestra la información necesaria para la elaboración de las órdenes de compra.

Tiene como ventaja que desde el momento en que se realizan las compras se mantiene un control de la fecha de caducidad de todos los medicamentos, de manera que siempre estén vigentes, tanto en el almacén como en las farmacias. Además, genera solicitudes de compra en forma automática para reponer el inventario a partir de información, como es el caso de análisis de inventario, tasa de uso, ciclo de revisión, manejo de inventario de seguridad, costos de manejos y de órdenes.

El sistema también permite la consulta de información estadística de cada artículo, incluyendo costo de venta promedio y última entrada, costo de reposición, rotación de inventario y estadísticas de movimiento en el almacén, así como el comportamiento de cada proveedor. Calcula el tiempo promedio de entrega, número de devoluciones e importes, número de pedidos recibidos y porcentaje de pedidos completos. (7)

X-HIS

Sistema de Información Hospitalaria desarrollado por la compañía australiana iSOFT, permite a las organizaciones sanitarias conseguir la máxima eficiencia y calidad en sus servicios. Es adaptable y extensible a cualquier centro hospitalario, ya sea público o privado. Posee una interfaz altamente intuitiva con ventajas como: visión general del conjunto de funcionalidades básicas, versatilidad en el acceso a distintas opciones, reducción del tiempo de formación, etc.

Facilita la gestión de la información clínica y administrativa de un hospital o conjunto de hospitales (multicentro) con las áreas de gestión, hospitales de referencia, centros de especialidades, laboratorios, y

otros organismos o corporaciones que se desee, proporcionando una Historia Clínica única y completa del paciente. (8)

Posee funcionalidades para las diferentes áreas del hospital, una de ellas es la encargada de la logística, en esta se facilita el control y la gestión de almacenes de cualquier organización sanitaria, independientemente de su tamaño y estructura. Permite hacer un seguimiento de todo el material que maneja un centro sanitario: entradas, salidas, préstamos, compras e inventarios. Además gestiona la recepción y distribución interna de los productos, así como la contratación administrativa y el registro y comprobación de facturas.

Es un sistema con una amplia gama de funcionalidades que aunque tiene la ventaja de ser una aplicación web y multiplataforma no permite realizar predicción ni llevar el control de las fechas de vencimiento de los productos.

MEDINOUS

Sistema de Gestión Hospitalaria de gran alcance, flexible y fácil de usar. Es una aplicación de escritorio diseñada y desarrollada con el propósito de ofrecer disímiles beneficios en hospitales de varias especialidades. Proporciona información relevante en el hospital para apoyar la toma eficaz de decisiones en la atención al paciente, la administración del hospital y la contabilidad financiera, en un flujo continuo.

Está constituido por varios módulos, uno de ellos está destinado para llevar el control del almacén general e inventario, entre sus funcionalidades se encuentran: transferencia de acciones en línea, mantenimiento de existencias en diferentes sub-almacenes, verificación de existencias físicas y ajuste, citas y solicitudes de compra, compra de artículos, gestión de proveedores y la facilidad de pago a los mismos, devolución de artículos a proveedores, gestión de stocks y verificación de facturas. (9)

Estos sistemas ofrecen casi todas las funcionalidades deseables por cualquier institución de salud para el área de almacén. Sin embargo, ninguno permite realizar predicciones del consumo de los productos almacenados, siendo este el principal motivo por el cual no pueden ser usados directamente. Por esta razón se hace necesario el desarrollo de un componente que sí sea capaz de predecir y que pueda utilizarse en el Sistema de Información Hospitalaria alas HIS.

Una de las vías existentes para realizar predicciones es la Minería de Datos (MD). Para la creación del componente de predicción de consumo de productos para el sistema alas HIS, se analizarán sus técnicas, de modo que pueda seleccionarse alguna de ellas para dar solución al problema planteado.

1.2 Predicción

La predicción es una de las formas que tienen las instituciones para mejorar la toma las decisiones. Al predecir, se trata de calcular algún hecho futuro en general, como resultado de un análisis racional o de un estudio de los datos existentes, (10). Este término se ha utilizado con éxito en varias ramas de la sociedad: en la economía, para predecir ingresos, gastos, precios del mercado, evolución futura del salario, demanda eléctrica, consumo de agua, incendios forestales y para planificar estrategias de venta; en la educación, para predecir el rendimiento académico de los estudiantes; en el deporte, para la selección de talentos y la predicción de resultados en partidos, y en la medicina, para emitir diagnósticos y tratamientos, así como para predecir la actividad biológica de compuestos orgánicos y el consumo de productos en los hospitales.

En la actualidad se le concede gran importancia a la predicción de consumo de productos en las instituciones de salud, resultando esta, una tarea que permite planificar adecuadamente la distribución de medicamentos, material quirúrgico y otros recursos útiles que deben estar disponibles en todo momento.

1.3 Almacén Hospitalario

Un almacén hospitalario es un espacio o lugar físico perteneciente a una entidad hospitalaria, que está destinado para alojar medicamentos, materiales, equipamientos y productos sanitarios. Según la cantidad de artículos que se almacenen, puede clasificarse en:

- Almacenes centrales: sirven a toda una comunidad autónoma o a una provincia. Es el almacén adecuado para el material de oficina.
- Almacenes completos en el hospital: se almacena en el hospital la mayor parte de los artículos que se necesitan. La ventaja es que se dispone de muchas cosas rápidamente. Los inconvenientes son que se requiere invertir mucho dinero para efectuar la compra y para el almacenamiento, además si se almacenan muchas unidades de un determinado producto se pueden producir muchas caducidades si este no sale del almacén en gran cantidad.

- Almacenes pequeños con control garantizado por parte de los proveedores: se llega a un acuerdo con los proveedores para que estos garanticen que no se van a producir desabastecimientos. (11)

1.4 Descubrimiento de Conocimiento en Bases de Datos

El Descubrimiento de Conocimiento en Bases de Datos (KDD, por sus siglas en inglés), se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información. (12)

El KDD persigue varias metas, estas son:

- Procesar automáticamente grandes cantidades de datos crudos.
- Identificar los patrones más significativos y relevantes.
- Presentarlos como conocimiento apropiado para satisfacer las metas del usuario. (13)

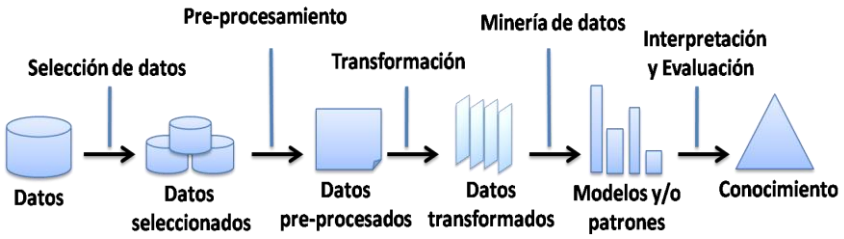


Figura 1.1 Proceso de KDD.

Como muestra la Figura 1.1, las etapas del proceso KDD se dividen en cinco fases y son:

- **Selección de datos:** en esta etapa se determinan las fuentes de datos y el tipo de información a utilizar. Es la etapa donde los datos relevantes para el análisis son extraídos desde la o las fuentes de datos.
- **Pre-procesamiento:** esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes de datos en una forma manejable, necesaria para las fases posteriores. En esta etapa se utilizan diversas estrategias para manejar datos faltantes o en blanco, datos inconsistentes o que están fuera de rango, obteniéndose al final una estructura de datos adecuada para su posterior transformación.

- **Transformación:** consiste en el tratamiento preliminar de los datos, transformación y generación de nuevas variables a partir de las ya existentes con una estructura de datos apropiada.
- **Minería de Datos:** es la fase de modelamiento, en donde métodos inteligentes son aplicados con el objetivo de extraer patrones previamente desconocidos que están contenidos u "ocultos" en los datos.
- **Interpretación y evaluación:** se identifican los patrones obtenidos que son realmente interesantes, basándose en algunas medidas y se realiza una evaluación de los resultados obtenidos. (14)

Además de las fases descritas, frecuentemente se incluye una fase previa de análisis de las necesidades de la organización y definición del problema, en la que se establecen los objetivos de la MD, (15). Señalar que las fases Selección de datos y Pre-procesamiento se engloban bajo el nombre de Preparación de datos y que aproximadamente el sesenta por ciento del esfuerzo total para realizar este proceso, se emplea durante la misma, esto se evidencia en la Figura 1.2.

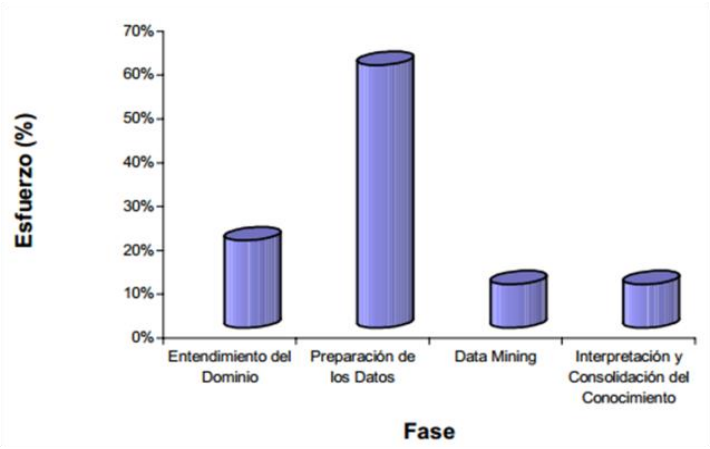


Figura 1.2 Esfuerzo en las fases del proceso KDD.

Si bien el término KDD se emplea para describir el proceso completo, y dentro de este existe una fase denominada MD, que engloba la aplicación de técnicas y herramientas, ambos términos son utilizados indistintamente con frecuencia para referirse a todo el proceso.

1.5 Minería de Datos

La Minería de Datos es el empleo de algoritmos y procedimientos para sacar a la luz asociaciones, correlaciones, reglas, patrones e incluso excepciones interesantes o potencialmente útiles, desconocidos y escondidos en bases de datos (o almacenes de datos), (16). Cualquier problema para el que existan datos históricos almacenados es un problema susceptible de ser tratado mediante técnicas de Minería de Datos, (17).

La MD tiene sus raíces básicamente en dos áreas del conocimiento: la primera y más grande en la cual tiene sus cimientos, es la estadística clásica, la cual cuenta con diversos conceptos como la distribución estándar, la varianza, entre muchos otros, que juegan un papel muy importante en el proceso de la misma, ya que brindan gran parte de la fundamentación bajo la cual muchos de sus modelos han sido construidos. (18) La segunda área de conocimiento que hace parte de la fundamentación de la MD es la Inteligencia Artificial (IA), que procura aplicar procesamiento lógico a diversos problemas estadísticos.

Principales **características y objetivos** de la Minería de Datos:

- Explorar los datos que se encuentran en las profundidades de las bases de datos, como los almacenes de datos, que algunas veces contienen información almacenada durante varios años.
- En algunos casos, los datos se consolidan en un almacén de datos y en mercados de datos; en otros, se mantienen en servidores de Internet e Intranet.
- El entorno de la Minería de Datos suele tener una arquitectura Cliente-Servidor.
- Las herramientas de la Minería de Datos ayudan a extraer el mineral de la información enterrado en archivos corporativos o en registros públicos, archivados.
- El minero es, muchas veces, un usuario final con poca o ninguna habilidad de programación.
- Hurgar y sacudir a menudo implica el descubrimiento de resultados valiosos e inesperados.
- Las herramientas de la Minería de Datos se combinan fácilmente y pueden analizarse y procesarse rápidamente.
- Debido a la gran cantidad de datos, algunas veces resulta necesario usar procesamiento en paralelo para la Minería de Datos.
- La Minería de Datos produce cinco tipos de información: asociaciones, secuencias, clasificaciones, agrupamientos y pronósticos. (19)

Metodología de Minería de Datos

Para guiar el proceso de extracción de conocimiento se utiliza la metodología CRoss-Industry Standard Process for Data Mining (CRISP-DM). Esta metodología está descrita en términos de un modelo de proceso jerárquico, consistente en un conjunto de tareas descritas en cuatro niveles de abstracción (de lo general a lo específico): fases, tareas genéricas, tareas especializadas, e instancias de procesos. (20)

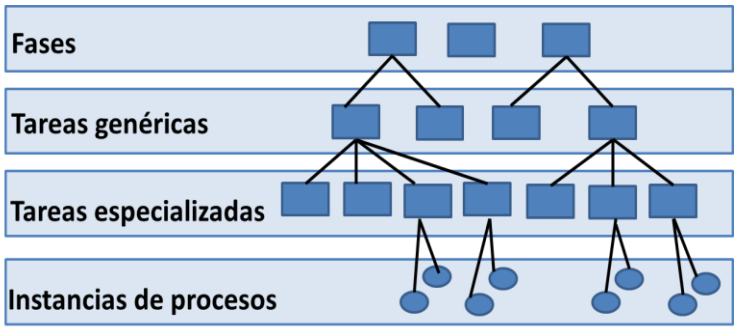


Figura 1.3 Niveles de la metodología CRISP-DM.

En el nivel superior, el proceso de Minería de Datos es organizado en un número de fases; cada fase consiste de varias tareas genéricas de segundo nivel. Este segundo nivel lo llaman genérico porque está destinado a ser bastante general para cubrir todas las situaciones posibles de MD. Las tareas genéricas están destinadas a ser tan completas y estables como sea posible. El tercer nivel, es el lugar para describir como las acciones en las tareas genéricas deberían ser realizadas en ciertas situaciones específicas. El cuarto nivel, instancias de procesos, es un registro de las acciones, decisiones y resultados de la MD. (21)

CRISP-DM incluye un modelo y una guía, estructurados en seis fases, algunas de estas son bidireccionales, lo que significa que permitirán revisar parcial o totalmente las fases anteriores.

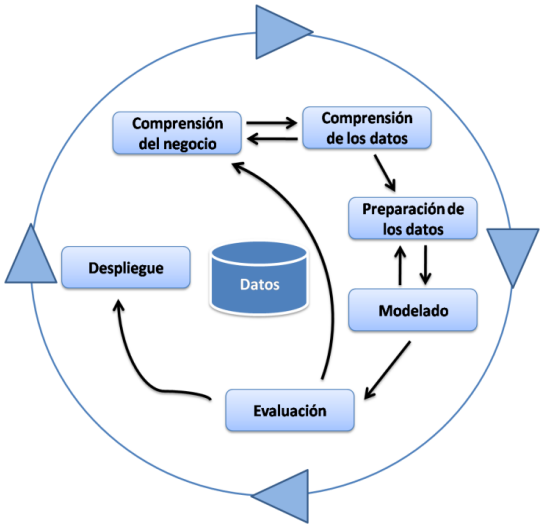


Figura 1.4 Fases de la metodología CRISP-DM.

Comprensión del negocio (Objetivos y requerimientos desde una perspectiva no técnica)

- Establecimiento de los objetivos del negocio.
- Evaluación de la situación.
- Establecimiento de los objetivos de la Minería de Datos.
- Generación del plan del proyecto.

Comprensión de los datos (Familiarizarse con los datos teniendo presente los objetivos del negocio)

- Recopilación inicial de datos.
- Descripción de los datos.
- Exploración de los datos.
- Verificación de calidad de datos.

Preparación de los datos (Obtener la vista minable o dataset)

- Selección de los datos.
- Limpieza de datos.
- Construcción de datos.

- Integración de datos.
- Formateo de datos.

Modelado (Aplicar las técnicas de Minería de Datos a los dataset)

- Selección de la técnica de modelado.
- Diseño de la evaluación.
- Construcción del modelo.
- Evaluación del modelo.

Evaluación (De los modelos de las fases anteriores, para determinar si son útiles a las necesidades del negocio)

- Evaluación de resultados.
- Revisar el proceso.
- Establecimiento de los siguientes pasos o acciones.

Despliegue (Explotar utilidad de los modelos, integrándolos en las tareas de toma de decisiones de la organización)

- Planificación de despliegue.
- Planificación de la monitorización y del mantenimiento.
- Generación de informe final.
- Revisión del proyecto. (22)

Áreas en las que se ha aplicado Minería de Datos satisfactoriamente:

- Astronomía: clasificación de cuerpos celestes.
- Meteorología: predicción de tormentas, entre otros.
- Medicina: caracterización y predicción de enfermedades, probabilidad de respuesta satisfactoria a tratamiento médico.
- Industria y manufactura: diagnóstico de fallas.
- Mercadotecnia: fidelidad de clientes, selección de sitios de tiendas, entre otros. (23)

1.6 Técnicas de Minería de Datos

Las técnicas de MD provienen de la IA y de la Estadística, dichas técnicas, no son más que algoritmos, más o menos sofisticados que se aplican sobre un conjunto de datos para obtener resultados. Cualquiera que sea el problema a resolver, no existe una única técnica para solucionarlo, sino que puede ser abordados siguiendo aproximaciones distintas, (24). Estos algoritmos se clasifican en dos grandes categorías:

Supervisados o Predictivos: predicen el valor de un atributo (etiqueta) de un conjunto de datos, conocidos otros atributos (atributos descriptivos). A partir de datos cuya etiqueta se conoce, se induce una relación entre dicha etiqueta y otra serie de atributos. Esas relaciones sirven para realizar la predicción en datos cuya etiqueta es desconocida. Esta forma de trabajar se conoce como aprendizaje supervisado y se desarrolla en dos fases:

- Entrenamiento: se construye un modelo usando un subconjunto de datos con una etiqueta conocida.
- Prueba: se prueba el modelo sobre el resto de los datos.

No Supervisados o de Descubrimiento del Conocimiento: descubren patrones y tendencias en los datos actuales (no utilizan datos históricos). El descubrimiento de esa información sirve para llevar a cabo acciones y obtener un beneficio (científico o de negocio) de ellas. (25)

Para predecir el consumo de productos debe utilizarse uno de los algoritmos de tipo supervisado. Por esta razón, se hará énfasis en ellos, de forma que pueda seleccionarse alguno para llevar a cabo la predicción.

Técnicas o algoritmos no supervisados:

- Técnicas de «clustering»: son técnicas que parten de una medida de proximidad entre individuos y a partir de ahí, se buscan los grupos de individuos más parecidos entre sí, según una serie de variables mesuradas, (26). Cuando se representa la información obtenida a través de clusters se pierden algunos detalles de los datos, pero a la vez se simplifica dicha información.
- Reglas de asociación: se utilizan para descubrir hechos que ocurren en común dentro de un determinado conjunto de datos. Las reglas de asociación tienen diversas aplicaciones como:

soporte para la toma de decisiones, diagnóstico y predicción de alarmas en telecomunicaciones y análisis de información de ventas.

Técnicas o algoritmos supervisados:

- Árboles de decisión: diagramas que representan en forma secuencial condiciones y acciones; muestran qué condiciones se consideran en primer lugar, en segundo lugar y así sucesivamente. Estos árboles permiten mostrar la relación que existe entre cada condición y el grupo de acciones permisibles asociado con ella. Un árbol de decisión sirve para modelar funciones discretas, en las que el objetivo es determinar el valor combinado de un conjunto de variables, y basándose en el valor de cada una de ellas, determinar la acción a ser tomada.

  Los árboles de decisión son normalmente construidos a partir de la descripción de la narrativa de un problema. Ellos proveen una visión gráfica de la toma de decisión necesaria, especifican las variables que son evaluadas y qué acciones deben ser tomadas. Cada vez que se ejecuta un árbol de decisión, solo un camino será seguido dependiendo del valor actual de la variable evaluada. Se recomienda el uso del árbol de decisión cuando el número de acciones es pequeño y no son posibles todas las combinaciones. (27)
- Redes neuronales: están inspiradas en el modelo biológico, son generalizaciones de modelos estadísticos clásicos. Su novedad radica en el aprendizaje secuencial, el hecho de utilizar transformaciones de las variables originales para la predicción y la no linealidad del modelo. Permiten aprender en contextos difíciles, sin precisar la formulación de un modelo concreto. Su principal inconveniente es que para el usuario son una caja negra. (28)

  Las redes de neuronas están siendo utilizadas en distintos sectores, como la industria, el gobierno, el ejército, las comunicaciones, la investigación aeroespacial, la banca y las finanzas, los seguros, la medicina, la distribución, la robótica, el marketing, etc. (29)
- Algoritmos genéticos: simulan el modelo biológico de la evolución de las especies. En principio, cualquier problema que se plantee, como la optimización de una combinación entre distintas componentes, estando estas sujetas a restricciones, puede resolverse mediante algoritmos genéticos.
- Series temporales: secuencia de observaciones o valores, medidos en determinados momentos del tiempo, ordenados cronológicamente y, normalmente, espaciados entre sí de manera uniforme.

Esta técnica permite modelar las componentes básicas de la serie: tendencia, ciclo y estacionalidad y así poder hacer predicciones para el futuro, tales como cifra de ventas, previsión de consumo de un producto o servicio, etc. (30)

1.7 Selección de la técnica de Minería de Datos adecuada. Justificación

La técnica que mejor se ajusta para cumplir el objetivo de la presente investigación es la basada en el uso de series temporales, ya que para realizar la predicción de los valores futuros de una o varias variables se utiliza como única información, la contenida en los valores pasados de la serie que mide la evolución de la(s) variable(s) en cuestión. Es decir, se tratan de identificar los patrones históricos y luego, suponiendo que se mantendrán en el futuro, se extrapolan. Existen importantes razones para utilizar métodos de predicción basados en el análisis de series temporales, algunas de estas son:

- La información histórica se encuentra almacenada de forma cronológica.
- Puede ser que se desee sólo predecir qué es lo que va a pasar y no el por qué pasa.
- Puede ser muy útil a la hora de realizar una buena planificación de recursos sanitarios, en función de la demanda que se espera en el futuro, prevista por el modelo.

Además, se tuvo en cuenta que: los árboles de decisión son muy similares a los sistemas de predicción basados en reglas, y que a pesar de utilizar datos históricos, solo se recomienda su uso cuando el número de acciones es pequeño. En las redes neuronales, la adquisición del conocimiento incluye la selección de ejemplos y aunque es utilizada para realizar predicción, su aplicación es más complicada, siendo para el usuario una caja negra. Por último, los algoritmos genéticos se utilizan mayormente para solucionar problemas donde se necesite optimizar, estos escogen una determinada población y de la misma ofrecen la solución más óptima.

1.8 Series Temporales

Una serie temporal es una secuencia de N observaciones recogidas a intervalos regulares de tiempo. Se utilizan en numerosos campos para llevar el control de actividades como: las temperaturas horarias, los precios diarios de acciones al cierre de la bolsa, la tasa de desempleo mensual, la tasa de mortalidad infantil por año, etc.

Clasificación de las series temporales

- Forma de observación de la serie:
  - Discreta: las observaciones se recogen solo en momentos determinados de tiempo, generalmente a intervalos iguales.
  - Continua: la serie se genera y se observa de forma continua, como, por ejemplo, la temperatura, que se observa de forma continua en el tiempo por medio de aparatos físicos.
- Tipo de predicción de la serie:
  - Determinista: los futuros valores de la serie pueden predecirse con exactitud.
  - Estocástica: el futuro sólo se puede determinar de modo parcial por las observaciones pasadas, se considera que la distribución de probabilidad de los futuros valores está condicionada a los valores pasados.

Modelos de series temporales

Los modelos de series temporales pueden clasificarse de dos tipos:

- Univariante o escalar: sólo se analiza una serie temporal en función de su propio pasado.
- Multivariante o vectorial: se analizan varias series temporales a la vez. Cuando se construye un modelo multivariante, se supone que hay cierta dependencia o relación entre los pasados de las diversas series. (31)

En este trabajo se van a considerar únicamente las series temporales discretas y estocásticas, específicamente el modelo univariante o escalar, el cual se puede definir como un conjunto de observaciones de una variable X. Si hay *T* observaciones, se denota por: X *(t)* donde *t=1,…,T,* el cual indica el tiempo en que se observa la variable. Los datos u observaciones se suelen recoger a intervalos iguales de tiempo; es el caso de series mensuales, trimestrales, etc. Las predicciones obtenidas a partir de un modelo univariante no son más que extrapolaciones de los datos observados hasta el momento *T*.

Para llevar a cabo el análisis de series temporales el primer paso es graficar la serie, ya que permite detectar las componentes esenciales de la misma:

Tendencia: es la dirección general de la variable en el período de observación, es decir, el cambio a largo plazo de la media de la serie.

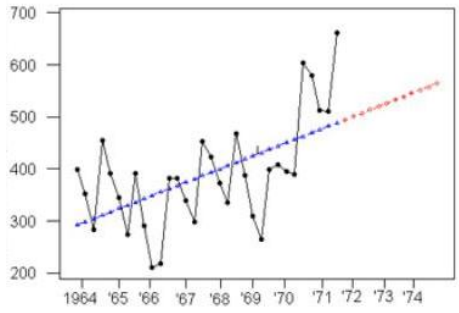


Figura 1.5 Representación de la Tendencia.

Estacionalidad: representa el movimiento periódico de la serie de tiempo. La duración de la unidad del período es generalmente menor que un año. Puede ser un trimestre, un mes, un día, etc. Las principales fuerzas que causan una variación estacional son las condiciones del tiempo.

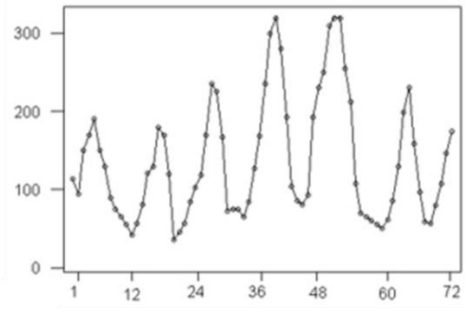


Figura 1.6 Representación de la Estacionalidad.

Ciclo: movimiento cíclico a medio plazo de período superior a un año.

Irregular o Residual (componente aleatoria): movimientos no sistemáticos de la serie que no se pueden predecir. Representa todos los tipos de movimientos de una serie de tiempo que no sea tendencia, variaciones estacionales y fluctuaciones cíclicas.

La serie observada es la unión de todos o algunos de los componentes presentes en la misma. En series largas es posible identificar el componente de ciclo, pero en series no tan largas, los componentes de tendencia y ciclo se confunden, por lo que es bastante habitual que un solo componente recoja los

movimientos de más larga duración. A este componente se le denomina Tendencia-Ciclo, o simplemente Tendencia. Así, el modelo más utilizado en el análisis de series temporales de frecuencia superior o igual a la mensual está formado por Tendencia, Estacionalidad e Irregular. Dos son los esquemas o modelos que generalmente son admitidos sobre la forma en que una serie temporal se descompone:

Aditivo: $X(t) = T(t) + E(t) + I(t)$

Multiplicativo : $X(t) = T(t) \times E(t) \times I(t)$

Donde:

$X(t)$: serie observada en instante t.

$T(t)$: componente de tendencia.

$E(t)$: componente estacional.

$I(t)$: componente irregular (aleatoria).

Usar un modelo u otro para analizar la serie depende del comportamiento de sus componentes. Puede darse el caso de que aunque el modelo apropiado sea el multiplicativo, no resulte sencillo trabajar directamente con él. En este caso, el modelo puede convertirse en aditivo mediante el uso de logaritmos:

$$\mathrm{X}(\mathrm{t}) = \mathrm{T}(\mathrm{t}) \times \mathrm{E}(\mathrm{t}) \times \mathrm{I}(\mathrm{t}) \rightarrow \log \mathrm{X}(\mathrm{t}) = \log \mathrm{T}(\mathrm{t}) + \log \mathrm{E}(\mathrm{t}) + \log \mathrm{I}(\mathrm{t}) \rightarrow \log \mathrm{X}(\mathrm{t}) = \mathrm{T}(\mathrm{t}) + \mathrm{E}(\mathrm{t}) + \mathrm{I}(\mathrm{t}), \quad (32)$$

Métodos para estimar Tendencia:

Hay varios métodos para estimar $T(t)$. Dos de los más utilizados consisten en:

- Suavizar (o filtrar) los valores de la serie.

La idea central es definir a partir de la serie observada una nueva serie que suaviza los efectos ajenos a la tendencia (estacionalidad, efectos aleatorios), de manera que pueda determinarse la dirección de la misma.

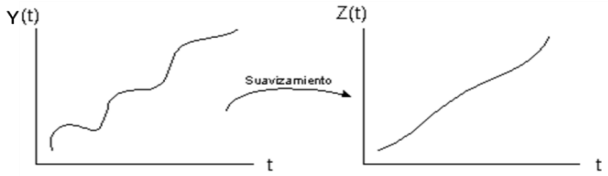


Figura 1.7 Serie original y serie suavizada.

Para ello se utilizan filtros lineales, el más usado es el llamado promedio o media móvil y su objetivo es eliminar de la serie los componentes estacionales y accidentales. Entre sus ventajas se encuentran: es útil para analizar el comportamiento o realidades del pasado, sirve para ver desviaciones ocasionales por encima o por debajo de la tendencia, representa mejor que una línea recta, porque representa la propia realidad. (33) Sin embargo, tiene como inconveniente que no permite calcular valores futuros, por lo tanto no permite predecir. Además no anticipa posibles cambios, sino que los confirma una vez que se han producido.

- Ajustar una función del tiempo, como un polinomio u otra función suave de *t*.

Para hallar la Tendencia por esta vía, el método más utilizado es Mínimos Cuadrados Ordinarios (MCO), esta es una técnica de análisis numérico encuadrada dentro de la optimización matemática, en la que, dados un conjunto de pares ordenados: variable independiente, variable dependiente, y una familia de funciones, se intenta encontrar la función, dentro de dicha familia, que mejor se aproxime a los datos (un mejor ajuste).

Este método presenta muchas ventajas en cuanto a lo fácil de su uso y por lo adecuado del planteamiento estadístico matemático, que permite adecuarse a los supuestos para los modelos econométricos. Es objetivo, pues sólo depende de los resultados experimentales. Es reproducible, ya que proporciona la misma ecuación, sin importar quién realice el análisis. Proporciona una estimación probabilística de la ecuación que representa datos experimentales. Provee intervalos pequeños de error.

Debido a que el suavizamiento de la serie mediante las medias móviles no permite la estimación de valores futuros, para el cálculo de la tendencia se ajustará una función a los datos observados, mediante el método de MCO.

Pasos para realizar la predicción:

- Hallar la función de la tendencia que mejor se ajusta a los datos observados, usando MCO.
- Calcular la tendecia en cada instante de tiempo.
- Calcular las series de residuos y analizar cuál es el modelo correspondiente a la serie observada.
- Calcular la estacionalidad.
- Realizar proyecciones.

Hallar la función de la tendencia que mejor se ajusta a los datos observados, usando MCO.

- **Parábola de MCO :** $Y(t) = a + bt + ct^2$
- **Recta de MCO :** $Y(t) = a + bt$

En la parábola, los coeficientes de regresión se calculan resolviendo el sistema de ecuaciones de la Figura 1.8, mientras que en la recta, se calculan resolviendo las ecuaciones de la Figura 1.9. En ambos casos *N* es el número de observaciones.

$$aN + b\sum_{i=1}^{n} X_i + c\sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} Y_i$$

$$a\sum_{i=1}^{n} X_i + b\sum_{i=1}^{n} X_i^2 + c\sum_{i=1}^{n} X_i^3 = \sum_{i=1}^{n} X_i Y_i$$

$$a\sum_{i=1}^{n} X_i^2 + b\sum_{i=1}^{n} X_i^3 + c\sum_{i=1}^{n} X_i^4 = \sum_{i=1}^{n} X_i^2 Y_i$$

Figura 1.8 Sistema de ecuaciones para hallar los coeficientes de regresión en una parábola, mediante MCO.

$$b = \left(N\sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i\right) / \left(N\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} X_i\right)$$

$$a = \left(\sum_{i=1}^{n} X_i^2 \sum_{i=1}^{n} Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} X_i Y_i\right) / \left(N\sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} X_i\right)$$

Figura 1.9 Ecuaciones para hallar los coeficientes de regresión en una recta, mediante MCO.

Coeficientes de correlación

Otro de los aspectos a tener en cuenta a la hora de ajustar es el coeficiente de correlación *(r),* el cual toma valores en el rango de −1 a 1, y da una medida del ajuste realizado. Cuanto más extremo es *r* (más se acerca a −1 o a 1), significa que mejor se ajusta el modelo.

- Coeficiente de correlación lineal

$$r = \left(N \sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i \right) / \left(\sqrt{\left(N \sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} X_i \right)} \sqrt{\left(N \sum_{i=1}^{n} Y_i^2 - \sum_{i=1}^{n} Y_i \sum_{i=1}^{n} Y_i \right)} \right)$$

- Coeficiente de correlación no lineal

Para calcular el coeficiente de correlación no lineal se deben haber calculado previamente los coeficientes de regresión.

$$r = \sqrt{b \left(\sum_{i=1}^{n} X_i Y_i - \left(\sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i \right) / N \right) + c \left(\sum_{i=1}^{n} X_i^2 Y_i - \left(\sum_{i=1}^{n} X_i^2 \sum_{i=1}^{n} Y_i \right) / N \right)}$$

Luego se analizan los coefientes de correlación previamente calculados, pudiéndose determinar cual de las dos curvas es la que mejor se ajusta a los datos observados.

Calcular la tendecia en cada instante de tiempo

Después de seleccionada la curva de mejor ajuste se procede a calcular la tendencia $T(t)$, que no es más que evaluar la función obtenida en cada instante de tiempo.

Calcular las series de residuos y analizar cuál es el modelo correspondiente a la serie observada.

Para escoger el modelo para predecir deben calcularse las series residuales o de Residuos $R(t)$, y analizarse los Coefientes de Variación *(CV).* Una serie residual no es más que la serie generada a partir de la original por eliminación de la Tendencia, esta deberá contener predominantemente fluctuaciones estacionales. Por su parte, el *CV* es lo que, luego de calculada $R(t)$ permite seleccionar entre un modelo y otro para predecir.

Para calcular $R(t)$ se emplean las siguientes expresiones:

Modelo Aditivo: $R(t) = X(t) - \check{T}(t)$ Modelo Multiplicativo: $R(t) = X(t) / \check{T}(t)$

donde $\check{T}(t)$ es la tendencia ya calculada.

Suponiendo que la serie a analizar es trimestral, para cada uno de los modelos, $R(t)$ queda de la siguiente manera:

Período / Estación	1	2	K	Promedio Fila	STD Fila	CV Fila
1	*R(1)*	*R(5)*	*R(4k-3)*	$\check{R}(1)$	S_1	$S1/\check{R}(1)$
2	*R(2)*	*R(6)*	*R(4k-2)*	$\check{R}(2)$	S_2	$S2/\check{R}(2)$
3	*R(3)*	*R(7)*	*R(4k-1)*	$\check{R}(3)$	S_3	$S3/\check{R}(3)$
4	*R(4)*	*R(8)*	*R(4k)*	$\check{R}(4)$	S_4	$S4/\check{R}(4)$

Tabla 1.1 Serie de Residuos.

donde:

$$S_1 = \sqrt{\frac{\left((R(1) - \check{R}(1))^2 + (R(5) - \check{R}(1))^2 + \ldots + (R(4k-3) - \check{R}(1))^2\right)}{N}}$$

$$\ldots$$

$$S_4 = \sqrt{\frac{\left((R(4) - \check{R}(4))^2 + (R(8) - \check{R}(4))^2 + \ldots + (R(4k) - \check{R}(4))^2\right)}{N}}$$

Figura 1.10 Fórmulas para calcular la desviación estándar (STD) de la fila correspondiente.

Una forma de seleccionar el modelo, es por inspección de los *CV.* Luego, comparando dichos coeficientes, parece razonable seleccionar el modelo cuyos coeficientes sean menores en términos absolutos.

Calcular la estacionalidad

Una vez elegido el modelo a utilizar, se procede a estimar la estacionalidad $E(t)$:

- Si el modelo seleccionado es Aditivo, entonces la estacionalidad viene dada por:

$\check{E}(1) = \check{R}(1) - \check{R}$ $\quad\quad$ $\check{E}(2) = \check{R}(2) - \check{R}$

$\check{E}(3) = \check{R}(3) - \check{R}$ $\check{E}(4) = \check{R}(4) - \check{R}$

- Si el modelo seleccionado es Multiplicativo, entonces la estacionalidad viene dada por:

$\check{E}(1) = \check{R}(1) - (\check{R} - 1)$ $\check{E}(2) = \check{R}(2) - (\check{R} - 2)$

$\check{E}\ 3\ = \check{R}(3) - (\check{R} - 3)$ $\check{E}(4) = \check{R}(4) - (\check{R} - 4)$

En ambos casos:

$\check{R} = \sum_{h=1}^{4} \frac{\check{R}(h)}{4}$, donde $\check{R}(h)$ es el promedio de los valores de la fila correspondiente a cada uno de los trimestres.

Realizar proyecciones

Una vez obtenidos los componentes de la serie, se está en condiciones para realizar proyecciones, para ello se emplean las siguientes expresiones, en dependencia del modelo a utilizar.

$X(n+k) = \check{T}(n+k) + \check{E}((n+k)modS)$ $modelo\ Aditivo$

$X(n+k) = \check{T}(n+k) \times \check{E}((n+k)modS)$ $modelo\ Multiplicativo$

Para lograr un mejor entendimiento del procedimiento a efectuar para predecir, se realiza un ejemplo ilustrativo, (Ver **Anexo 1**).

1.9 Arquitectura Cliente–Servidor

La arquitectura Cliente – Servidor (C/S) es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realice se efectúe con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema. En este modelo, el papel del cliente es iniciar el diálogo, enviando peticiones al servidor conforme a algún protocolo asimétrico, pidiéndole que actúe, que le informe, o ambas cosas. El servidor es quien responde las solicitudes del cliente.

Entre sus principales características se encuentran:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.

- El cliente no necesita conocer la lógica del servidor, solo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente. (34)

1.10 Patrón de Diseño de Arquitectura

Modelo-Vista-Controlador

Modelo-Vista-Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.
- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. (35)

1.11 Proceso de Desarrollo de Software

RUP

RUP es un proceso de desarrollo de software que apoyándose en el Lenguaje de Modelado Unificado (UML, por sus siglas en inglés), constituye una de las metodologías estándares más populares para el desarrollo de sistemas orientados a objetos. Posee tres características fundamentales:

- Iterativo e Incremental: el trabajo se divide en partes más pequeñas o mini proyectos, permitiendo que el equilibrio entre casos de uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un

recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

- Guiado por los casos de uso: se define caso de uso como un fragmento de funcionalidad del sistema que proporciona al usuario un valor añadido. Los casos de uso representan los requisitos funcionales del sistema. Además guían su diseño, implementación y prueba.
- Centrado en la arquitectura: permite que todos los involucrados (desarrolladores y usuarios) tengan una visión común y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. (36)

RUP divide el proceso en cuatro fases (Inicio, Elaboración, Construcción y Transición), dentro de las cuales se realizan varias iteraciones en número variable según el proyecto. Los flujos de trabajo son: Modelado del negocio, Requisitos, Análisis y diseño, Implementación, Prueba y Despliegue. Además cuenta con los flujos de soporte Gestión de cambios y configuración, Gestión de proyectos, y Entorno. En cada uno de estos el esfuerzo varía, según la fase en la que se encuentre el proyecto.

UML

UML es utilizado para visualizar, especificar, construir y documentar los artefactos de un sistema de software orientado a objetos. No guía al desarrollador en la forma de realizar el análisis y diseño, ni le indica cuál proceso de desarrollo de software adoptar. Una de las ventajas que tiene sobre otros lenguajes de modelado es que diseñadores diferentes, modelando sistemas diferentes, pueden entender cada uno los diseños de los otros. Dos de las características fundamentales de UML son: viabilidad para corregir errores y tecnología orientada a objetos. Un modelo UML está compuesto por tres clases de bloques de construcción:

- Elementos: abstracciones de cosas reales o ficticias (objetos, acciones, etc.)
- Relaciones: relacionan los elementos entre sí.
- Diagramas: colecciones de elementos con sus relaciones. (37)

1.12 Lenguaje de Programación

Java

El lenguaje Java fue concebido por James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan en Sun Microsystems en 1991, inicialmente se denominó Oak, pero se le puso el nombre de Java en 1995, (38).

Características principales del lenguaje Java:

- Es multiplataforma, es decir, el mismo código Java que funciona en un sistema operativo, funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual de Java.
- Reduce en un cincuenta por ciento los errores más comunes de programación con lenguajes como C y C++ al eliminar muchas de las características de éstos, entre las que destacan: aritmética de punteros, macros (#define) y la necesidad de liberar memoria (free).
- Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo.
- Realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. La comprobación de tipos en Java ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo.
- Las aplicaciones de Java resultan extremadamente seguras, ya que no acceden a zonas delicadas de memoria o del sistema, con lo cual evitan la interacción de ciertos virus. (39)

1.13 Tecnologías Utilizadas

Capa de presentación

XHTML

Acrónimo inglés de eXtensible HyperText Markup Language (Lenguaje Extensible de Marcado de Hipertexto, en español). Es la versión XML (eXtensible Markup Language) de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones más estrictas de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium (W3C) de lograr una web semántica,

donde la información y la forma de presentarla estén claramente separadas. (40) Algunas de las ventajas de XHTML sobre otros formatos son:

- Un mismo documento puede adoptar diseños radicalmente distintos en diferentes aparatos, pudiendo incluso escogerse entre varios diseños para un mismo medio.
- Facilidad de edición directa del código y de mantenimiento.
- Formato abierto, compatible con los nuevos estándares que actualmente está desarrollando el W3C como recomendación para futuros agentes de usuario o navegadores.

JavaServer Faces

La tecnología JavaServer Faces (JSF) constituye un marco de trabajo (framework) de interfaces de usuario del lado del servidor para aplicaciones web basadas en tecnología Java y en el patrón MVC. Es extensible, por lo que pueden crearse nuevos elementos de la interfaz o modificar los ya existentes. Una de sus ventajas es que ofrece una clara separación entre el comportamiento y la presentación. (41)

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entradas, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages (JSP) que permiten expresar una interfaz JSF dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.

Ajax4JSF

Ajax4JSF es una librería de código abierto, que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código JavaScript. Mediante esta librería se puede variar el ciclo de vida de una petición JSF, recargar

determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. (42)

Facelets

Facelets es un framework simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego asociarle los componentes JSF específicos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF. (43) Tener un código ordenado y reutilizable (sobre todo en aplicaciones grandes), simplificación de desarrollo y facilidad en el mantenimiento son algunas de las ventajas que trae consigo su uso. Además, permite la definición de disposición de páginas basada en plantillas, la composición de componentes, creación de etiquetas personalizadas, desarrollo amigable para el diseñador gráfico y creación de librerías de componentes.

RichFaces

RichFaces es una librería de componentes visuales para JSF, escrita en su origen por Exadel y adquirida por JBoss. Además, RichFaces posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF. (44)

Son características de RichFaces las siguientes:

- Se integra perfectamente en el ciclo de vida de JSF.
- Incluye funcionalidades Ajax, de modo que nunca se ve el JavaScript y tiene un contenedor Ajax propio.
- Soporta Facelets.
- Soporta CSS.
- Es un proyecto de código abierto, activo y con una comunidad también activa.

Seam UI

Seam UI es una serie de controles JSF altamente integrables con JBoss Seam. Adicionan varias mejoras a JSF, desde validación, expresiones Extended EL e integración de la navegación en la interfaz de usuario. (45)

Capa de negocio

JBoss Seam 2.1.1

JBoss Seam es un framework que integra la capa de presentación con la capa de negocios y persistencia. Con Seam basta agregar anotaciones propias de este a los objetos Entidad y Session de Enterprise JavaBeans, logrando con esto escribir menos código Java y XML. Otra característica importante es que pueden hacerse validaciones en los POJOs (Plain Old Java Object). (46)

JFreeChart 1.0.13

JFreeChart es una biblioteca libre de gráficos cien por ciento de Java, que facilita la labor de los desarrolladores al proporcionar conjuntos de gráficos de calidad profesional. El conjunto amplio de características de JFreeChart incluye:

- Una API consistente y bien documentada, admitiendo una amplia gama de tipos de gráficos.
- Un diseño flexible que es fácil de ampliar y está dirigido a las aplicaciones del lado del servidor y del cliente.
- Compatibilidad con muchos tipos de salida, archivos de imagen (incluyendo PNG y JPEG) y formatos de archivo de gráficos vectoriales (incluidos PDF, EPS y SVG). (47)

JasperReports 4.5.0

Es una librería de clases de Java de código abierto diseñada para facilitar el agregar capacidades de reporte a las aplicaciones Java. No es una herramienta por sí sola, por lo que no se puede instalar. Para utilizar JasperReports es necesario añadirlo a las aplicaciones Java por medio de la inclusión de su librería.

Está escrito completamente en Java y puede ser usado en gran variedad de aplicaciones de Java, para generar contenido dinámico. Su propósito principal es ayudar a crear documentos de tipo páginas, preparados para imprimir en una forma simple y flexible. JasperReports se usa comúnmente con iReport, una herramienta de código abierto para la edición de informes.

Capa de acceso a datos

Java Persistence API

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java y está incluida en el estándar Enterprise JavaBeans 3.0. Esta API busca unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue su diseño es no perder las ventajas de la orientación a objetos al interactuar con una base de datos, como sucedía con Enterprise JavaBeans 2, y permitir usar objetos regulares conocidos como POJOs. (48)

Enterprise JavaBeans 3.0

Enterprise JavaBeans 3.0 (EJB3) permite construir aplicaciones de negocios portables, reutilizables y escalables usando el lenguaje de programación Java. Ofrece estándares para aplicaciones de negocios basadas en componentes, orientadas a objetos y distribuidas. (49)

Hibernate 3.0.0 GA

Hibernate es un framework que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación mediante archivos declarativos XML o anotaciones en los Beans de las entidades que permiten establecer estas relaciones. (50)

Es software libre, distribuido bajo los términos de las licencias GNU y Licencia Pública General Reducida (LGPL, por sus siglas en inglés). Como todas las herramientas de su tipo, busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información le permite a la aplicación manipular los datos de la base operando sobre objetos, con todas las características de la Programación Orientada a Objetos (POO).

Tecnologías horizontales

Java Enterprise Edition 5

Java Enterprise Edition 5 (JEE5) es una plataforma para dar soporte y desarrollar software para las empresas. Esta edición dio un gran paso hacia la excelencia, pues añadió nuevas tecnologías de punta que le incorporan a la plataforma: comodidad, mejor rendimiento y tiempo de desarrollo reducido, permitiendo a los desarrolladores disminuir la cantidad de código necesario en la implementación. (51)

Entre sus características se encuentran:

- Es un estándar del sector para desarrollar aplicaciones, robustas, escalables y seguras para el servidor.
- Proporciona APIs de servicios web, modelos de componentes, gestión y comunicación que lo convierten en el estándar del sector para implementar aplicaciones SOA (Arquitectura Orientada a Servicios, en español).
- Define una infraestructura para aplicaciones que pueden implementar distintas organizaciones de forma independiente, pero se comporta de forma coherente en todas ellas.

Java Runtime Environment 6

Java Runtime Environment 6 (JRE6), es un conjunto de utilidades que permite la ejecución de programas Java sobre todas las plataformas soportadas. JVM (Máquina Virtual Java) es una instancia de JRE en tiempo de ejecución. (52) El JRE es lo mínimo que debe contener un sistema para poder ejecutar una aplicación Java sobre el mismo.

1.14 Herramientas Utilizadas

Visual Paradigm for UML 8.0

Visual Paradigm for UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: negocio, requerimientos, análisis y diseño, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de calidad con un menor coste; permitiendo dibujar todos los tipos de diagramas así como generar código y documentación. Proporciona abundantes

tutoriales de UML, demostraciones interactivas y proyectos. Cuenta con la posibilidad de diseñar los esquemas correspondientes al modelo Entidad-Relación. (53)

Eclipse 3.4.2 Ganymede

Eclipse es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), de código abierto y multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-Liviano" basadas en navegadores. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. (54)

PostgreSQL 8.4

PostgreSQL es un Sistema Gestor de Base de Datos (SGBD) relacional orientado a objeto y libre, publicado bajo la licencia Berkeley Software Distribution (BSD). Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad, vistas, etc. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa, sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales que trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group). (55) Es multiplataforma y utiliza el lenguaje SQL para llevar a cabo sus búsquedas de información, las bases de datos generadas dentro de servidores de SQL son bases de datos relacionales.

PostgreSQL destaca por su amplísima lista de prestaciones, que lo hacen capaz de competir con cualquier SGBD comercial, las siguientes son algunas de ellas:

- Está desarrollado en C, con herramientas como Yacc y Lex.
- La API de acceso al SGBD se encuentra disponible en lenguajes como: C, C++, Java, Perl, PHP, Python y TCL, entre otros.
- Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario.
- Su administración se basa en usuarios y privilegios.
- Los mensajes de error pueden estar en español y hacer ordenaciones correctas con palabras acentuadas o con la letra ‘ñ’.

- Es altamente confiable en cuanto a estabilidad se refiere.
- Puede extenderse con librerías externas para soportar encriptación, búsquedas por similitud fonética (soundex), etc. (56)

PgAdmin III 1.14.2

PgAdmin III es una aplicación gráfica para gestionar el SGBD PostgreSQL, siendo la más completa y popular con licencia de código abierto. Está escrita en C++. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar scripts programados. (57)

JBoss Server 4.2.2 GA

JBoss es un servidor de aplicaciones J2EE implementado en Java. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. (58) Sus principales características son:

- Producto de licencia de código abierto sin coste adicional.
- Confiable a nivel de empresa.
- Orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios del middleware para cualquier objeto de Java.
- Soporte completo para Java Management Extensions (JMX).

iReport

iReport es un constructor y diseñador de informes visual, potente, intuitivo y muy fácil de usar para JasperReports escrito en Java. Permite que los usuarios corrijan visualmente informes complejos con cartas e imágenes. iReport está integrado con JFreeChart, una de las bibliotecas gráficas de código abierto más extendidas para Java. (59)

Algunas de sus características son:

- Soporta internacionalización nativamente.
- Browser de la estructura del documento.
- Recopilador y exportador integrados.
- Tiene asistentes para las plantillas.
- Facilidad de instalación.
- Incluye Wizard's (asistentes) para crear automáticamente informes.
- Soporta JavaBeans como orígenes de datos.

Conclusiones del Capítulo

Las soluciones informáticas estudiadas, aunque ofrecen casi todas las funcionalidades deseables por cualquier institución de salud para el área de almacén, no permiten realizar predicciones del consumo de los productos almacenados. Como resultado de este estudio se determinó desarrollar un componente que sí sea capaz de predecir y que pueda utilizarse en el módulo de Almacén del sistema alas HIS. Para llevar a cabo la predicción se seleccionó la técnica series temporales, por las características que posee. El estudio de las tecnologías y herramientas empleadas en el departamento Sistemas de Gestión Hospitalaria posibilitó conocer sus características, las cuales serán de mucha ayuda para el desarrollo del componente de predicción.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se describen los conceptos relacionados con el ámbito del componente, con los cuales se elabora el modelo de dominio. Se realizan las especificaciones de los requisitos de software, tanto funcionales como no funcionales y se crea el diagrama de casos de uso del sistema.

2.1 Modelo de Dominio

El modelo de dominio, o modelo conceptual se utiliza para capturar los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. Los objetos o conceptos incluidos en el modelo de dominio no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de construcción del sistema. (60)

Conceptos fundamentales del dominio

Para una mejor comprensión del modelo de dominio, se realiza una breve descripción de los conceptos encontrados en el problema.

Institución hospitalaria: centro de salud destinado a la prevención, diagnóstico, tratamiento y si es posible, cura de enfermedades físicas y/o psíquicas, incluyendo la realización de prácticas quirúrgicas. Es donde podrá utilizarse el sistema alas HIS y con él, el componente en cuestión.

Almacén hospitalario: local, espacio o lugar físico perteneciente a una entidad hospitalaria. Está destinado para alojar los productos hasta su distribución por las diferentes áreas del hospital.

Producto: representa todos los medicamentos, materiales y equipos útiles existentes en el almacén de una institución hospitalaria.

Almacenero: persona encargada de realizar las actividades dentro del almacén y que por medio de un ordenador puede acceder al componente para realizar la predicción.

Predicción: es la acción que el componente realiza para emitir un resultado solicitado por el almacenero. Consiste en anticipar el consumo de los productos almacenados.

Consumo: es la acción y efecto de consumir o gastar. Representa el conocimiento del consumo pasado de los productos almacenados.

Diagrama del modelo de dominio

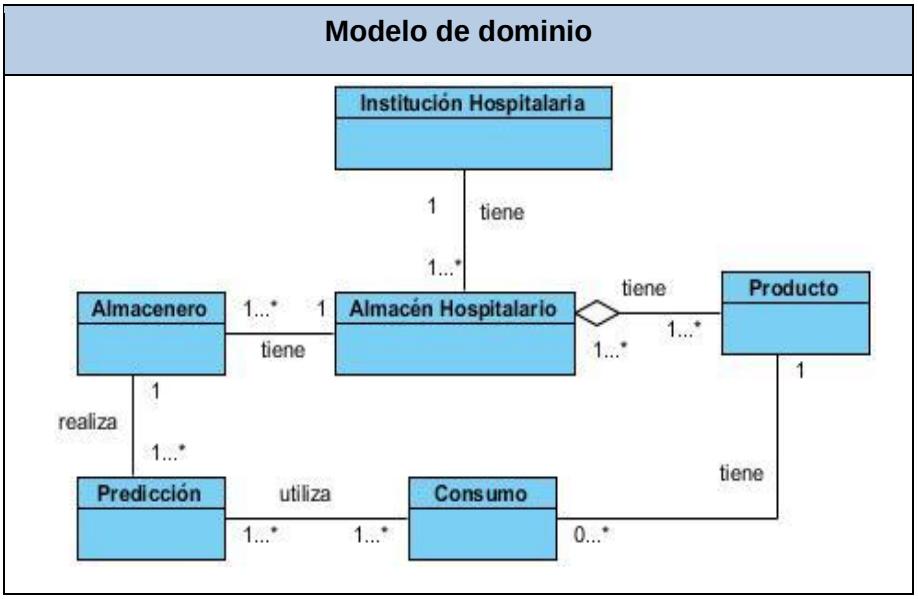


Figura 2.1 Diagrama del modelo de dominio

2.2 Especificación de los Requisitos del Software

Un requisito de software es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente, (61). Los requisitos del software se clasifican en Requisitos Funcionales y Requisitos No Funcionales.

Requisitos Funcionales

Los Requisitos Funcionales (RF) son capacidades o condiciones que el sistema debe cumplir, para satisfacer las necesidades del usuario. Para el componente de predicción, se identificaron los siguientes:

RF01: Crear predicción.

RF02: Seleccionar producto para predicción.

RF03: Ver resultados de predicción.

RF04: Buscar predicción en historial.

RF05: Modificar predicción.

RF06: Eliminar predicción.

RF07: Generar reporte de predicción.

RF08: Generar reporte por rango de fecha.

RF09: Exportar reporte.

Requisitos No Funcionales

Los Requisitos No Funcionales (RNF) son las propiedades o cualidades que el producto debe tener. Son las características que logran que la solución sea más atractiva, usable, rápida y confiable. Existen múltiples categorías para clasificar estos requerimientos, entre las que se encuentran:

Apariencia o interfaz externa

La interfaz del componente debe tener un diseño sencillo y amigable para el usuario, cumpliendo en todo momento las Pautas de Diseño de Interfaz de Usuario definidas para el sistema alas HIS.

Usabilidad

El componente está concebido para ser utilizado por cualquier persona con conocimientos básicos de computación, de una manera rápida e intuitiva. Por tal motivo, los usuarios deberán ser capaces de alcanzar sus objetivos con un mínimo esfuerzo. Para lograrlo, el componente debe admitir flujos alternos, como cancelar la operación y salir de la vista actual. El significado de los íconos utilizados debe ser comprensible para el usuario. Los cuadros de diálogos no deben contener información que sea irrelevante. La entrada incorrecta de datos será detectada e informada al usuario, esta información se mostrará en lenguaje natural, sin códigos específicos del sistema, además se indicará en forma constructiva la posible solución.

Portabilidad

El componente podrá ser utilizado en los sistemas operativos Linux (Oracle Linux 5.5+, 6.x, Ubuntu Linux 10.04 y superior, entre otras versiones) y Windows (XP, Server 2008, Server 2012, Vista, 7, 8, entre otras versiones).

Seguridad

El usuario deberá autenticarse para acceder al sistema. Se registrarán cada una de las acciones realizadas por el usuario. Ninguna información que se ingrese será eliminada físicamente de la base de datos. Las contraseñas podrán cambiarse por el propio usuario o por el administrador del sistema en general.

Software

El componente formará parte del sistema alas HIS, este debe poder desplegarse en los sistemas operativos Windows y Linux, utilizando Java Runtime Environment, JBoss Server y PostgreSQL. El usuario deberá disponer de un navegador web, el cual puede ser Google Chrome, Internet Explorer 7.0, Firefox 3.6 o versiones superiores de estos.

Hardware

El componente debe poder utilizarse en clientes que posean como mínimo memoria RAM de 1GB y microprocesador Intel® Core-2 Duo o Intel® Dual-Core.

Los servidores, por su parte, deberán tener alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables.

- Servidores de base de datos: PowerEdge R910, Procesador Intel® Xeon® CPU E7- 8837 @ 2.67 GHz, 16 GB de memoria RAM, 1199 GB de disco duro.
- Servidores de aplicaciones: PowerEdge R910, Procesador Intel® Xeon® CPU E7- 8837 @ 2.67 GHz, 16 GB de memoria RAM, 1199 GB de disco duro.

2.3 Modelo de Casos de Uso del Sistema

El modelo de casos de uso del sistema describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Este modelo está formado por actores, casos de uso y las relaciones que se establecen entre estos. Responde a cada una de las funcionalidades definidas en el epígrafe anterior.

Actor del sistema

Actor del sistema	Justificación
Almacenero	Persona que interactúa con el componente de predicción, tiene la posibilidad de acceder a todas las funcionalidades del mismo.

Tabla 2.1 Actor del sistema.

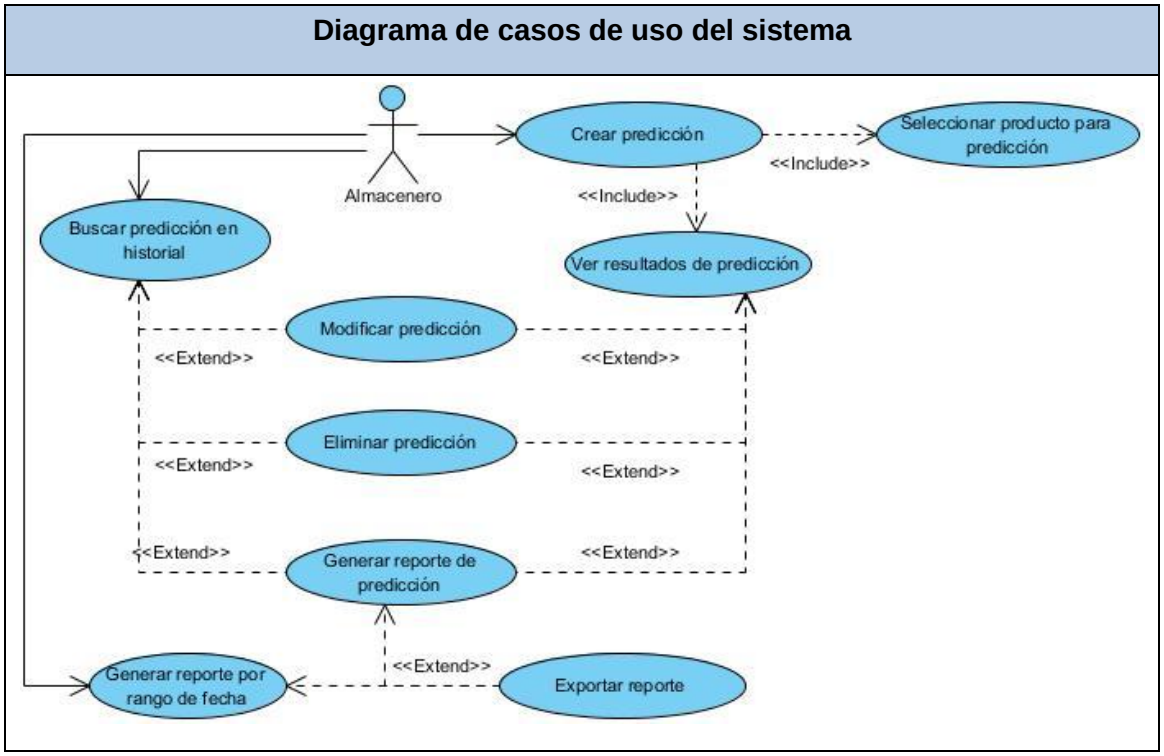


Figura 2.2 Diagrama de casos de uso del sistema.

Descripción textual de casos de uso

La descripción textual de un caso de uso describe los procesos o flujos de actividades que son objeto de automatización en el mismo. La siguiente tabla es un resumen de una de las descripciones de casos de uso identificados.

Crear predicción	
Propósito	Permitir realizar la predicción del consumo de uno o varios productos, para un período determinado por el actor.
Actor	Almacenero
Resumen	El caso de uso inicia cuando el actor accede a la opción Crear predicción, el sistema brinda la posibilidad de introducir y seleccionar los datos para crear la predicción, el actor selecciona e introduce los datos de la predicción y acepta, el sistema crea la(s) predicción(es), el caso de uso termina.
Referencias	RF2
Precondición	Debe haberse seleccionado al menos un producto.
Poscondición	Se creó al menos una predicción.

Tabla 2.2 Descripción textual del caso de uso Crear predicción.

Conclusiones del Capítulo

La tareas realizadas en este capítulo permitieron definir cada uno de los requisitos del componente, tanto funcionales como no funcionales y obtener, mediante el Proceso Unificado de Desarrollo, los artefactos correspondientes al modelo de domino y diagrama de casos de uso del sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

El presente capítulo tiene como objetivo llevar a cabo el análisis y diseño del componente. Para ello se describe la arquitectura y se elaboran los diagramas de paquetes, clases web e interacción. Se brinda además una descripción de las clases identificadas para su posterior implementación y se exponen las estrategias de reutilización empleadas en el desarrollo del componente de predicción propuesto.

3.1 Descripción de la Arquitectura. Fundamentación

La arquitectura de software es un conjunto de patrones y abstracciones que proporcionan el marco de referencia necesario para guiar la construcción del sistema. Permite a los programadores, diseñadores y analistas trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado. (62)

Para el desarrollo del componente se hará uso del patrón de arquitectura MVC (descrito en el capítulo 1). Este patrón es útil principalmente en aplicaciones que manejan gran cantidad de datos y donde se requiera una mejor separación de conceptos para que el desarrollo esté mejor estructurado.

La vista o presentación es la representación gráfica de los datos disponibles para la interacción con el usuario. La misma estará compuesta por páginas XHTML, desarrolladas básicamente con JSF, utilizando las librerías Ajax4JSF y RichFaces, que se complementan fácilmente con la plataforma de integración JBoss Seam. Se utilizan también componentes Seam de interfaz de usuario y Facelets como motor de plantillas.

El controlador es el encargado de manejar y responder las solicitudes que realiza el usuario, procesando la información y modificando los datos en caso que sea necesario. Estará compuesto por clases Java controladoras, que definen la lógica del negocio conjuntamente con los datos que se validan en la capa de presentación. Mediante el framework JBoss Seam se les especifica el contexto en que se encuentran, los cuales definen el estado de los datos y las entidades que manipulan. Se utilizan además las librerías JFreeChart y JasperReports, para la generación de gráficas y reportes respectivamente.

En el modelo o capa de datos se usará Hibernate como herramienta de mapeo objeto relacional, que es la implementación de EJB 3.0 y JPA.

3.2 Estrategias de Reutilización

La reutilización de código fuente en la construcción de sistemas facilita su desarrollo, ya que permite reducir tiempo, minimizar las redundancias y aprovechar el trabajo anterior. En cada uno de los módulos del sistema alas HIS, se hace uso de diversos componentes reutilizables, con el objetivo de lograr uniformidad, limpieza y organización en el código, incrementando así la calidad y eficiencia del producto. Estos componentes son: la clase *Bitacora*, para controlar las trazas de las acciones que son llevadas a cabo por los usuarios; la interfaz *IActiveModule,* para conocer en qué módulo y entidad se encuentra el usuario que está utilizando el sistema; la clase *Usuario*, que ofrece los datos del usuario que está trabajando con la aplicación en tiempo real; la clase *ReportManager, que* brinda las funcionalidades más significativas para la generación de los reportes y *ExportModalPanel,* que permite seleccionar el formato en el cual se exportará el reporte generado.

3.3 Modelo de Diseño

El modelo de diseño permite describir la realización física de los casos de uso, centrándose en los requisitos del sistema. Es usado como entrada inicial en las actividades de implementación y prueba, por lo que debe ser lo suficientemente específico para que el sistema pueda ser implementado sin ambigüedades.

Durante la confección del diseño se utilizan patrones de diseño, estos se consideran como una solución a un problema. Entre los patrones de diseño que más se utilizan se encuentran los patrones generales para la asignación de responsabilidades (GRASP, por sus siglas en inglés). Los patrones GRASP permiten asignarle a cada clase las tareas a realizar en correspondencia con la información que manejan, dándole el poder de instanciar otras clases en dependencia de la responsabilidad asignada, poniéndose de manifiesto el Experto y Creador. Otros de los patrones utilizados son el de Bajo Acoplamiento y Alta Cohesión, estos permiten que las distintas clases del sistema interactúen entre sí, sin que se afecte la reutilización de las mismas o el correcto funcionamiento de las clases por separado.

Diagrama de paquetes

Para la elaboración del modelo de diseño se propone una estructura en paquetes, ya que permite dividir el sistema en fragmentos manejables para su implementación. Este tipo de diagrama muestra como el sistema se divide en agrupaciones lógicas, mostrando las dependencias entre ellas.

El paquete *Repositorio de Clases* contiene tres subpaquetes en su interior: *Entidades, Sesiones y Vistas*. El subpaquete *Entidades* contiene las entidades autogeneradas y las personalizadas. Las entidades autogeneradas, como su nombre lo indica, se autogeneran desde la base de datos utilizando el ORM Hibernate (permite la generación de objetos Java desde la base de datos). Las entidades personalizadas son aquellas que se modifican, por lo que pueden heredar de las autogeneradas. El subpaquete *Sesiones* está conformado por las clases controladoras autogeneradas por el entorno de desarrollo, las clases controladoras personalizadas, que son las especializaciones de las controladoras autogeneradas y las clases controladoras del proceso, que son las clases controladoras de los funcionalidades automatizadas. El subpaquete *Vistas* está compuesto por los contenidos web referentes a las páginas clientes y a los formularios de cada una. El paquete *Realizar Predicción* se compone de subpaquetes, que responden a la realización de los casos de uso.

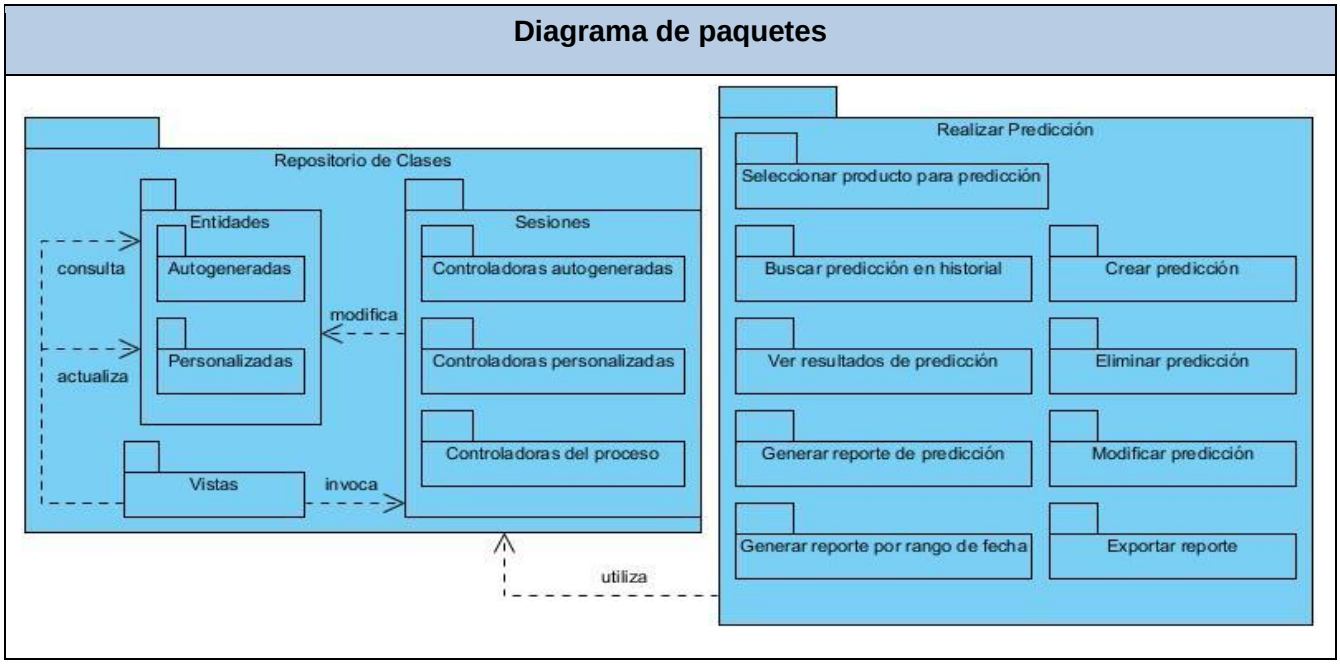


Figura 3.1 Diagrama de paquetes del diseño.

Diagramas de clases del diseño

Para modelar el diagrama de clases del diseño se emplea UML, el cual provee los elementos necesarios para representar estereotipos web. Estos elementos tienen como principales componentes las clases

Server Page, Client Page y Form, para la representación de las clases contenedoras de código de las páginas servidoras, clientes y los formularios respectivamente. Además se hace uso dos clases muy importantes: RichFaces API y EntityManager. RichFaces API representa la integración de los componentes de interfaz de usuario, validación y conversión de datos de RichFaces con las páginas clientes. La clase EntityManager representa la clase con las que se relacionan las controladoras para llevar a cabo las operaciones de persistencia de datos.

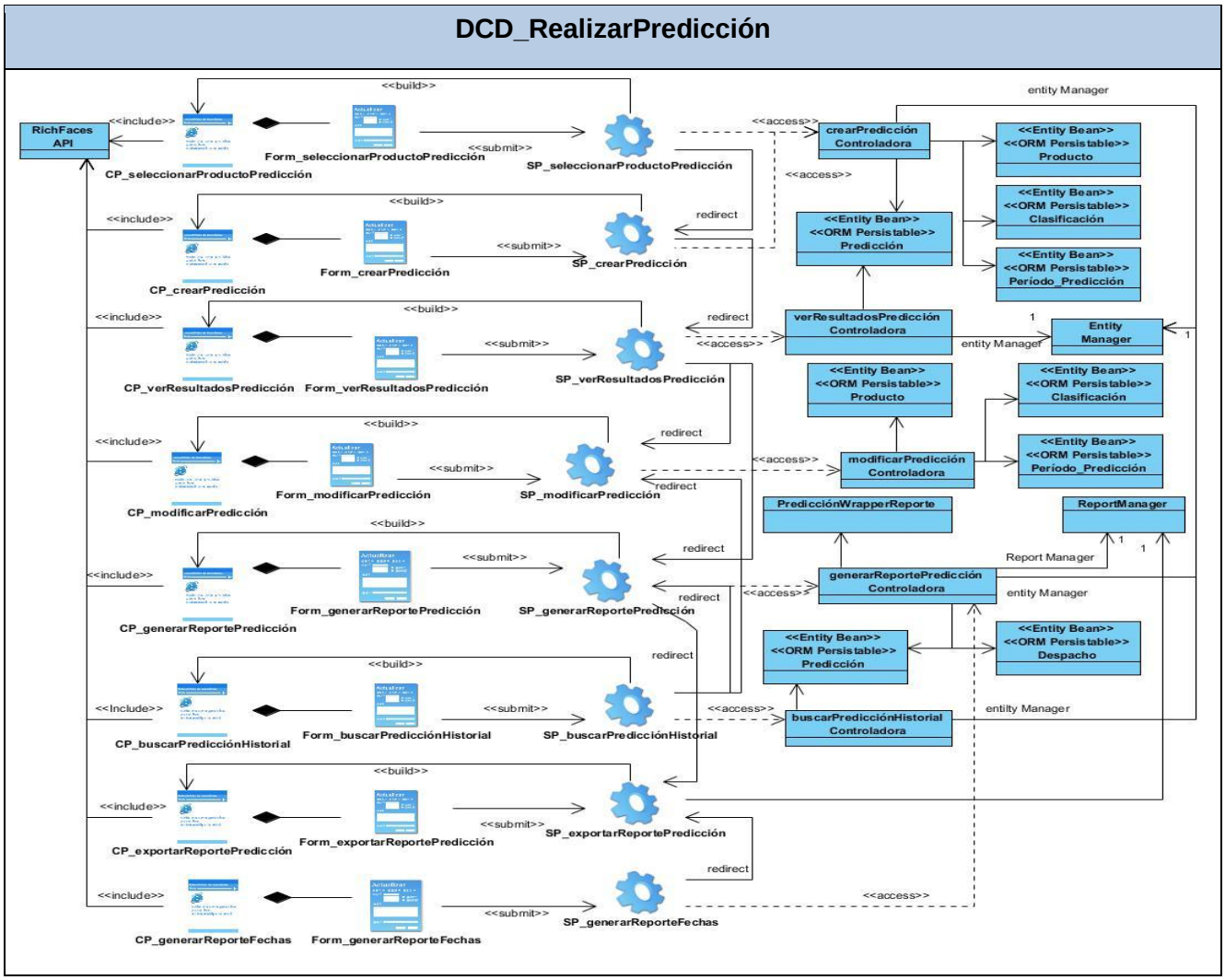


Figura 3.2 Diagrama de clases del diseño Realizar predicción.

Diagrama de interacción

Los diagramas de interacción consisten en un conjunto de objetos y sus relaciones, incluyendo los mensajes que pueden enviarse entre ellos. Se dividen en diagramas de colaboración y diagramas de secuencia. Los de colaboración muestran una interacción organizada en relación a los objetos que efectúan operaciones y los de secuencia muestran los objetos que participan en la interacción mediante sus líneas de vida y los mensajes que intercambian, organizados en forma secuencial. El diagrama de interacción seleccionado es el de secuencia.

Diagramas de secuencia correspondiente a uno de los casos de uso identificados

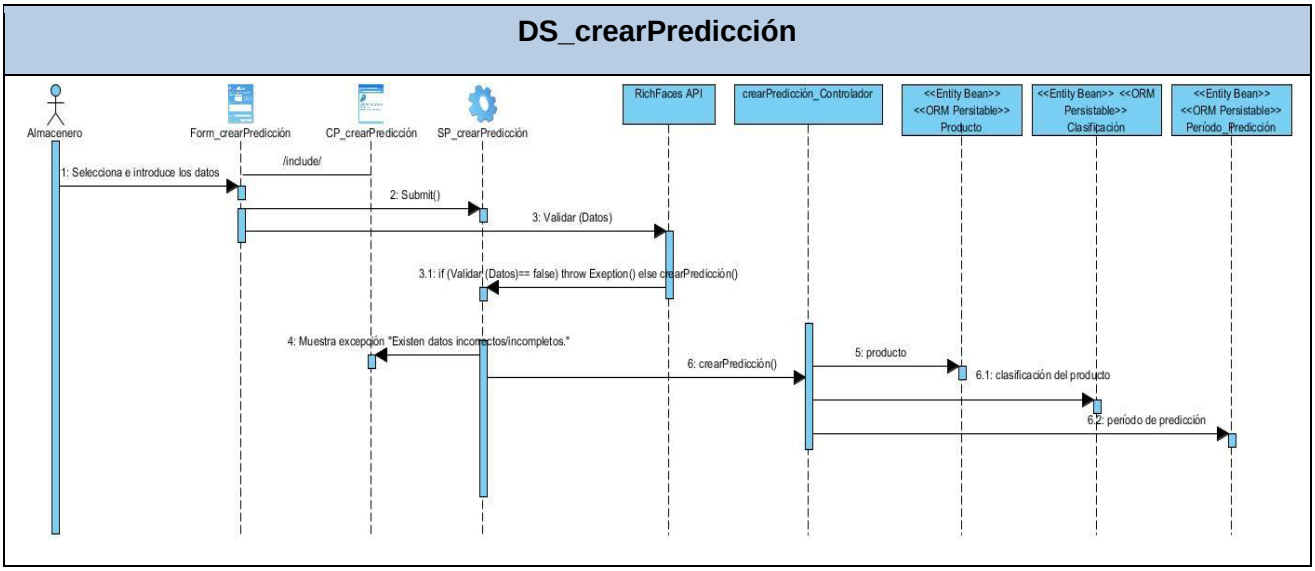


Figura 3.3 Diagrama de secuencia crearPredicción.

Descripción de las clases del diseño

Las clases de diseño se agrupan en:

Páginas Clientes: están compuestas por código XHTML, CSS, JavaScript. Son interpretadas por los navegadores web presentándole al usuario la interfaz con la que puede interactuar con el sistema.

Páginas Servidoras: están compuestas por componentes Facelets, RichFaces, JSF, Seam UI, así como código XHTML. Todo este código será ejecutado en el servidor web, generando páginas clientes que pueden ser representadas por los navegadores web.

Formularios: un formulario XHTML es una sección de un documento enmarcado entre tags <form> y que puede contener elementos especiales llamados controles, como: casillas de verificación y menús. Los usuarios normalmente "completan" un formulario modificando sus controles (introduciendo texto, seleccionando objetos de un menú, etc.), y lo envían al servidor donde estos son procesados.

Controladoras: implementan la lógica del negocio que se está informatizando, generalmente cada una de estas se encarga de la implementación de un caso de uso o un proceso en dependencia de la complejidad de los mismos.

La descripción de las clases identificadas permite la comprensión de las mismas, para su posterior implementación.

crearPrediccionControladora	
Tipo de clase: controladora	
Atributos	**Tipo**
cantidadPeriodo	long
entityManager	EntityManager
productoCustomList	ProductoCustomList
seleccionProducto	Map<Long, Boolean>
clasificacionProduct	String
firstSearch	Boolean
periodo	String
listPredicciones	List<Long>

Para cada responsabilidad:	
Nombre:	buscar()
Descripción:	Busca los productos para los cuales se realizará la predicción.
Nombre:	isMark(Long id)
Descripción:	Se utiliza para saber si el producto fue seleccionado o no.
Nombre:	crearPrediccion()
Descripción:	Crea la predicción a partir de los atributos definidos
Nombre:	ArrayList<String> listClasificaciones()
Descripción:	Retorna la lista de clasificaciones de los productos, que pueden ser: material, medicamento o equipo.

Tabla 3.1 Descripción de la clase CrearPrediccionControladora

Conclusiones del Capítulo

Con el desarrollo de este capítulo se logró describir la arquitectura a emplear y obtener los diagramas de paquetes, clases web e interacción. Se identificaron las clases que deben ser implementadas posteriormente, así como un conjunto de componentes que pueden ser reutilizados, permitiendo reducir tiempo y minimizar las redundancias en el desarrollo del componente.

CAPÍTULO 4: IMPLEMENTACIÓN

La implementación es el principal flujo de trabajo en la fase de construcción y está determinado por el lenguaje de programación. En este capítulo se lleva a cabo la elaboración del modelo de datos, así como la descripción de las tablas más importantes que lo forman. Se elaboran los diagramas de despliegue y componentes. Se definen también la seguridad, el tratamiento de errores, las estrategias de codificación y los estilos de código a utilizar.

4.1 Modelo de Datos

Un modelo de datos es utilizado para la descripción de una base de datos. Por lo general, permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base). (63) Está compuesto por entidades, atributos y sus relaciones.

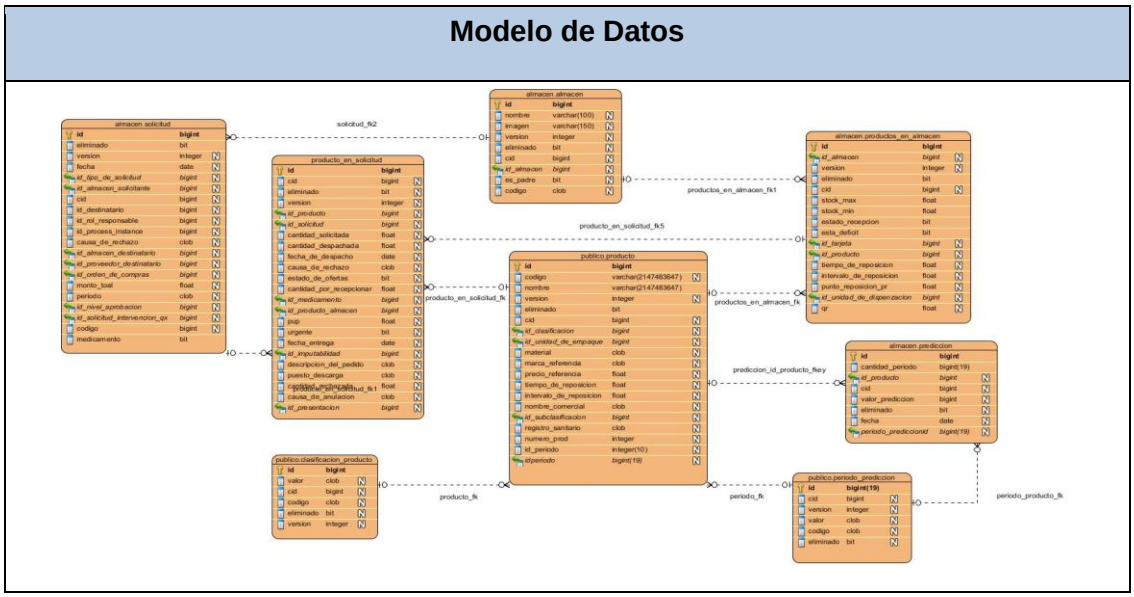


Figura 4.1 Modelo de datos

Descripción de las tablas de la base de datos

Atributos comunes en todas las entidades		
Atributos	**Tipo**	**Descripción**
id	bigInt	Id necesario en cada entidad para las referencias en las relaciones entre tablas.
version	integer	Indica con qué versión de la entidad se está trabajando. Es usado para garantizar que se está trabajando con la versión de la entidad más actualizada que existe en la base de datos.
eliminado	boolean	Permite la eliminación lógica con que cuenta el sistema, cuando está en verdadero indica que la entidad está eliminada.
cid	bigInt	Permite identificar quién realiza alguna acción sobre la entidad.

Tabla 4.1 Descripción de los atributos comunes entre todas las entidades.

producto		
Descripción: recoge la información de los productos existentes en el sistema.		
Atributos	**Tipo**	**Descripción**
codigo	varchar	Código que identifica al producto.
nombre	varchar	Nombre del producto.
id_clasificacion	BigInt	Identificador de la entidad clasificación.
material	varchar	Material del producto.
nombre_comercial	varchar	Nombre con el cual se comercializa el producto.

Tabla 4.2 Descripción de la tabla producto.

prediccion		
Descripción: recoge la información relacionada con la predicción del consumo de los productos.		
Atributos	**Tipo**	**Descripción**
cantidad_periodo	bigint	Tiempo para el que se realizó la predicción.
id_producto	bigint	Identificador de la entidad producto.
fecha	date	Fecha en que se realizó la predicción.
valor_prediccion	bigint	Valor resultante de la predicción.

Tabla 4.3 Descripción de la tabla prediccion.

productos_en_almacen		
Descripción: almacena los datos de la relación de las tablas producto y almacén.		
Atributos	**Tipo**	**Descripción**
id_almacen	bigint	Identificador de la entidad almacén.
id_producto	bigint	Identificador de la entidad producto.
esta_deficit	bit	Permite saber si el producto en el almacén está en déficit.

Tabla 4.4 Descripción de la tabla productos_en_almacen

productos_en_solicitud		
Descripción: recoge la información vinculada a los despachos realizados.		
Atributos	**Tipo**	**Descripción**
fecha_de_despacho	date	Fecha en la cual se realizó el despacho.

Id_producto	Bigint	Identificador de la entidad producto.
Id_solicitud	Bigint	Identificador de la entidad solicitud.
id_producto_almacen	bigint	Identificador de la entidad productos_en_almacen.
cantidad_solicitada	float	La cantidad de productos que fueron solicitados.

Tabla 4.5 Descripción de la tabla productos_en_solicitud

clacificacion_producto		
Descripción: tiene la clasificación que el producto puede tener (medicamento, equipo o material)		
Atributos	**Tipo**	**Descripción**
valor	bigint	Valor que puede tomar el nomenclador.
codigo	bigint	Permite la identificación del valor independientemente del idioma en que esté la aplicación. Permite la comparación del valor.

Tabla 4.6 Descripción de la tabla clasificacion_producto

almacen		
Descripción: recoge la información del almacén.		
Atributos	**Tipo**	**Descripción**
Id_almacen	bigint	Código que identifica al almacén.
nombre	varchar	Nombre del almacén.

Tabla 4.7 Descripción de la tabla almacen.

periodo_prediccion		
Descripción: tiene los períodos para los cuales puede hacerse la predicción (semestral, trimestral, mensual y anual).		
Atributos	**Tipo**	**Descripción**
valor	bigint	Valor que puede tomar el nomenclador.
codigo	bigint	Permite la identificación del valor independientemente del idioma en que esté la aplicación. Permite la comparación del valor.

Tabla 4.8 Descripción de la tabla periodo_prediccion.

4.2 Diagrama de Despliegue

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema.

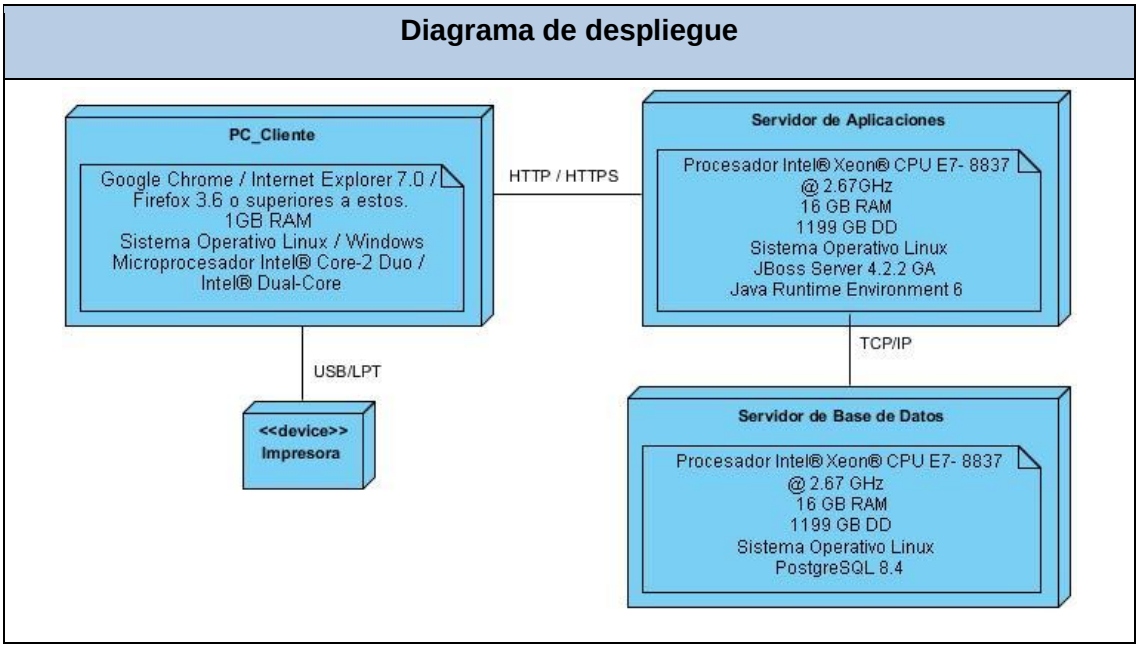


Figura 4.2 Diagrama de despliegue.

4.3 Diagrama de Componentes

Los diagramas de componentes muestran los componentes de software y las relaciones entre ellos.

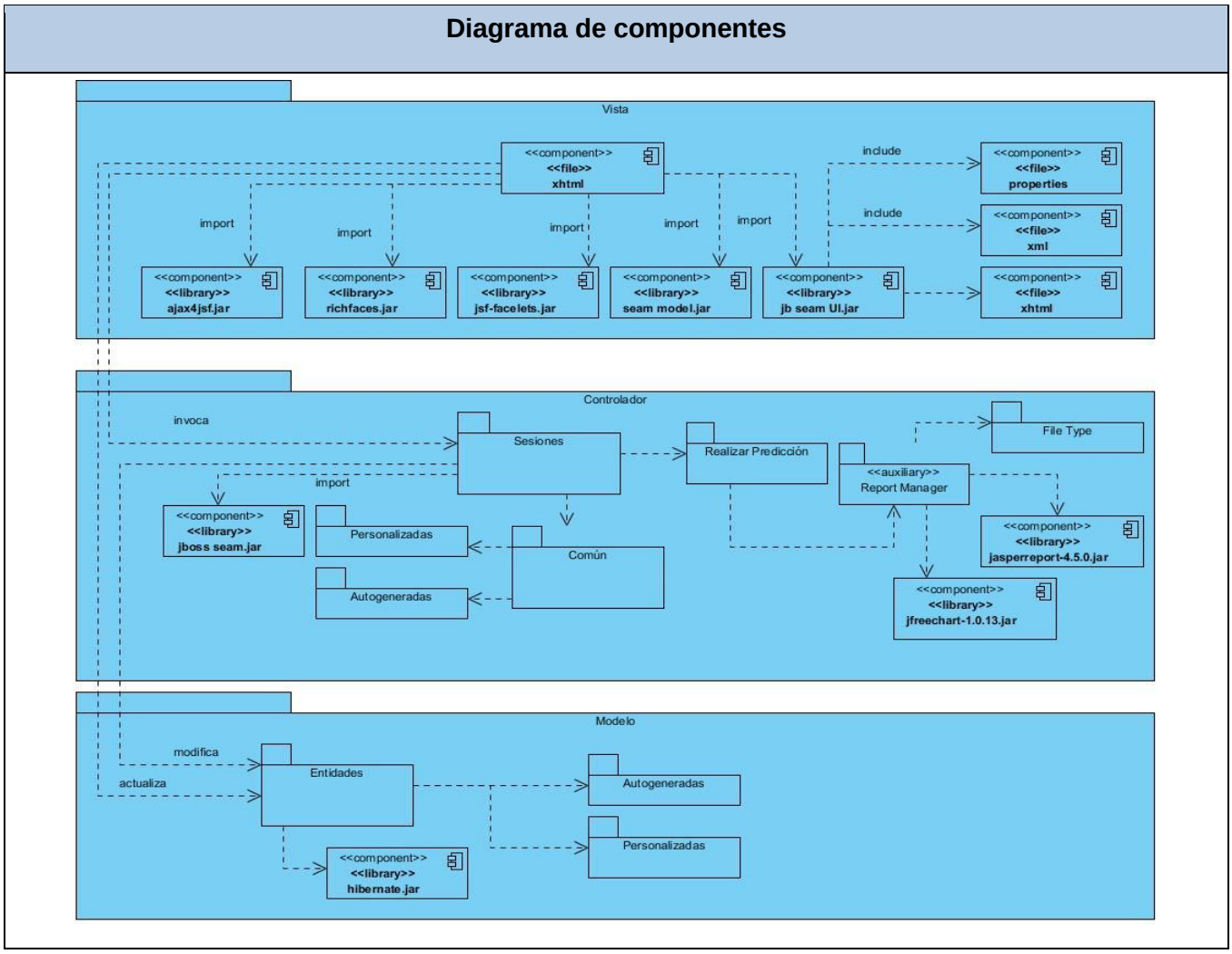


Figura 4.3 Diagrama de componentes.

4.4 Tratamiento de Errores

En Java, como en cualquier otro lenguaje de programación, pueden existir situaciones en las que el programa falle, estos errores se conocen como excepciones. Para capturar las excepciones se emplean las instrucciones **try** y **catch**. Entre las llaves de **try** se escribe el código que hará funcional el programa.

Para capturar la excepción que puede generar este código se utiliza **catch**, entre sus llaves se escribe el código para tratar el error. Se utiliza el componente *FacesMessages* del framework Seam, el cual se encarga de mostrar los mensajes en la interfaz de usuario.

4.5 Seguridad

Un sistema informático se considera seguro cuando garantiza la confidencialidad, integridad y disponibilidad de sus datos. El sistema alas HIS en general provee un conjunto de funcionalidades que se encargan de brindar seguridad, entre ellas se encuentran iniciar y cerrar sesión de trabajo.

Para iniciar la sesión de trabajo, se accede al sistema a partir de la inserción del nombre de usuario y contraseña correspondientes. El sistema por su parte, verifica que los datos introducidos sean válidos y da acceso al módulo, según los permisos otorgados. Mediante el cierre de la sesión, se permite la conclusión de las tareas realizadas en el sistema. Particularmente, para hacer uso del componente de predicción, el usuario debe tener permiso de acceso al módulo Almacén.

Las contraseñas podrán ser cambiadas únicamente por los propios usuarios o por el administrador del sistema en general. Los datos ingresados en el componente no serán eliminados físicamente de la base de datos. Además se llevará un registro de trazas, el cual permitirá almacenar todas las acciones realizadas por el usuario.

4.6 Estrategias de Codificación. Estándares y Estilos a utilizar

Los estándares de codificación son reglas que se siguen para lograr uniformidad en la escritura del código fuente y con ello, brindar un mayor entendimiento del mismo. Su empleo reduce el riesgo de que los desarrolladores introduzcan errores. Para la creación del componente, se hace uso de los siguientes:

- **Métodos y variables**

Los métodos y variables deben escribirse utilizando el estilo lowerCamelCase, es decir, con la primera letra del nombre en minúsculas y con la primera letra de cada palabra interna en mayúsculas. En los métodos no se permiten caracteres especiales, mientras que en las variables se deben evitar en la medida de lo posible. El nombre ha de ser lo suficientemente descriptivo, de forma que al leerlo, se conozca su propósito.

Ejemplo: int cantPeriodo; private void calcularTendencia()

- **Comentarios**

Debe evitarse el uso abusivo de comentarios, así como el uso de caracteres especiales dentro de los mismos. Deben comentarse cada clase/interfaz o método creado, de forma abreviada y siempre en tercera persona.

- **Declaraciones**
  - La declaración de las variables locales a una clase, método o bloque de código se realizan al principio del mismo y no justo antes de utilizarse la variable. Se recomienda una declaración por línea, ya que facilita los comentarios.

    Ejemplo: double coefCorrelacion; // coeficiente de correlación

    double numObservaciones; // cantidad de observaciones
  - No debe dejarse ningún espacio en blanco entre el nombre de un método y el paréntesis «(» que abre su lista de parámetros.

    Ejemplo: public void analizarModelo( )
  - La llave de apertura «{» aparece al final de la misma línea de la sentencia de declaración y la de cierre «}» empieza una nueva línea, luego de todas las líneas de sentencias.

    Ejemplo: public double realizarPrediccion(int cantPeriodo) {

    …

    }
- **Espacios en blanco**
  - Debe dejarse un espacio entre una palabra clave del lenguaje y un paréntesis.

    Ejemplo: if (modelo == 1) {

    …

    }
  - Todos los operadores binarios excepto «.» deben separarse de sus operandos con espacios en blanco. Los espacios en blanco no deben separar los operadores unarios, incremento («++») y decremento («--») de sus operandos.

    Ejemplo: sX2Y += X * X * Y;

- **Líneas en blanco**

Deben utilizarse entre métodos, así como entre las distintas secciones lógicas de estos, para facilitar la lectura.

Conclusiones del Capítulo

Con el desarrollo de las tareas correspondientes a este capítulo se logró elaborar el modelo de datos, que muestra las tablas de la base de datos que se utilizan. Se realizaron los diagramas de componentes y despliegue. Se obtuvo un componente de predicción de consumo de productos funcional, que fortalece el módulo Almacén del sistema alas HIS.

CONCLUSIONES

El desarrollo de las tareas de la investigación posibilitó dar cumplimiento al objetivo general y arribar a las siguientes conclusiones:

- Los sistemas informáticos analizados no permiten realizar predicciones del consumo de los productos existentes en los almacenes de las instituciones hospitalarias, se seleccionó la técnica Series Temporales como la adecuada para el desarrollo del componente de predicción.

- Los conceptos básicos relacionados con el ámbito del componente y los requisitos de software definidos permitieron obtener el modelo de domino y el diagrama de casos de uso del sistema.

- Se obtuvieron los artefactos correspondientes al componente de predicción de consumo, mediante el Proceso Unificado de Desarrollo.

- Se desarrolló un componente de predicción de consumo de productos funcional, que fortalece el módulo Almacén del sistema alas HIS.

RECOMENDACIÓN

Para una posterior versión del componente se recomienda:

- Incluir el modelo multivariante para llevar a cabo la predicción, de modo que puedan analizarse varias series temporales a la vez.

REFERENCIAS BIBLIOGRÁFICAS

1. **Universidad de las Ciencias Informáticas.** *Sistema de Información de Salud Alas HIS.* La habana. Cuba : s.n.

2. —. CESIM - Centro de Informática Médica. [En línea] [Citado el: 4 de diciembre de 2012.] http://gespro.cesim.prod.uci.cu.

3. **Ídem (1).**

4. **Universidad de las Ciencias Informáticas.** *Manual de Usuario.Módulo Emergencias del sistema alas HIS.* La Habana.Cuba : s.n.

5. **CNT SISTEMAS DE INFORMACIÓN S.A.** Pacientes, Software-Administración y Gestión de IPS – Clínicas -Hospitales. [En línea] [Citado el: 12 de diciembre de 2012.] http://www.guiadesolucionestic.com/soluciones-verticales/sector-salud-seguridad-social-/administracion-y-gestion-de-ips-clinicas-hospitales/289-pacientesrEs.

6. **TCA Software Solutions.** Assist-Sistema de Información Integral para Hospitales. [En línea] [Citado el: 12 de diciembre de 2012.] http://www.tcasoftwaresolutions.com/assist/es.

7. **grupo TCA.** *Sistema de Información para Hospitales y Servicios Clínicos Assist.*

8.**iSOFT.** Descripción x-HIS. [En línea] [Citado el: 13 de diciembre de 2012.] http://ittrade.com.mx/index.php?option=com_phocadownload&view=category&download=61:descripci.

9. Hospital Management System (HMS) - MediNous. [En línea] [Citado el: 13 de diciembre de 2012.] http://www.medinous.com/general-stores-inventory.html.

10. **Casimiro, María Pilar González.** *Técnicas de predicción económica.* 2009. 978-84-692-3815-8.

11. El rincón del vago. *Almacenes sanitarios.* [En línea] 1998. [Citado el: 15 de diciembre de 2013.] http://html.rincondelvago.com/almacenes-sanitarios.html.

12. **Web Minig.** KDD: Proceso de Extracción de conocimiento. [En línea] 2013. [Citado el: 16 de febrero de 2013.] http://www.webmining.cl/2011/01/proceso-de-extraccion-de-conocimiento/.

13. **Vallejos, Sofia J.** Minería de Datos. [En línea] 2006. [Citado el: 16 de febrero de 2013.] http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/Mineria_Datos_Vallejos.pdf.

14. **Ídem (12).**

15. **Ídem (12).**

16. **Acosta Aguilera, Manuel Ernesto.** *Minería de datos y descubrimiento de conocimiento.* Universidad de La Habana.Cuba : s.n.

17. **Aluja, Tomás.** *La minería de datos,entre la estadística y la inteligencia artificial.* 2001.

18. **García Bermúdez , José Antonio y Acevedo Ramirez, Ángela María .** Análisis para predicción de ventas utilizando minería de datos En almacenes de ventas de grandes superficies. [En línea] 2010. [Citado el: 26 de enero de 2013.] http://repositorio.utp.edu.co/dspace/bitstream/11059/1339/1/006312G216.pdf.

19. **Ídem (13).**

20. Slidechare. *Metodología CRISP-DM.* [En línea] 2013. [Citado el: 25 de mayo de 2013.] http://www.slideshare.net/oalonso/metodologa-de-data-mining-crisp.

21. **Dataprix.** Metodología CRISP-DM para minería de datos. [En línea] [Citado el: 16 de febrero de 2013.] http://www.dataprix.com/la-metodolog%C3%AD-crisp-dm.

22. **Goicochea, Aníbal.** CRISP-DM, Una metodología para proyectos de Minería de Datos. [En línea] [Citado el: 24 de abril de 2013.] http://anibalgoicochea.com/2009/08/11/crisp-dm-una-metodologia-para-proyectos-de-mineria-de-datos/.

23. **Ídem (18).**

24. **Ídem (17).**

25. **Miguel Quintales, Luis A, y otros, y otros.** Aplicación de técnicas de minería de datos en la construcción y validación de modelos predictivos y asociativos a partir de especificaciones de requisitos de

software. [En línea] [Citado el: 16 de febrero de 2013.] http://www.sc.ehu.es/jiwdocoj/remis/docs/minerw.pdf.

26. **Ídem (17).**

27. Toma de decisión con certidumbre. [En línea] [Citado el: 27 de febrero de 2013.] http://www.ehu.es/Degypi/Metodologia/montecarloyarboles.htm.

28. **Ídem (17).**

29. **Molina López, José Manuel y García Herrero , Jesús.** *Técnicas de análisis de datos. Aplicaciones prácticas utilizando microsoft excel y weka.* Madrid : s.n., 2004.

30. **Ídem (17).**

31. **Ídem (10).**

32. **Ídem (10).**

33. **Rincón del vago.** Tendencias estadísticas. [En línea] 1998. [Citado el: 25 de febrero de 2013.] http://html.rincondelvago.com/tendencias-estadisticas.html.

34. Redindustria. Arquitecturas cliente/servidor . [En línea] [Citado el: 12 de Febrero de 2013.] http://redindustria.blogspot.com/2008/03/arquitectruas-clienteservidor.html.

35. **Díaz González, Yanette y Fernández Romero, Yenisleidy.** Revista digital de las tecnologías de la información y las comunicaciones. Telemática. *Patrón Modelo-Vista-Controlador.* [En línea] [Citado el: 12 de junio de 2013.] http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/15. 1729-3804.

36. *Rational Unified Process (RUP).* Departamento de Sistemas Informáticos y Computación.Universidad Politécnica de Valencia. : s.n.

37. **Hernández Orallo, Enrique.** *El Lenguaje Unificado de Modelado (UML).*

38. Capítulo 2 El lenguaje Java. [En línea] [Citado el: 6 de diciembre de 2012.] http://www.dma.fi.upm.es/mabellanas/voronoi/java/java.htm#2.1.1.

39. **Cifuentes Briceño , Wilmer de Jesús .** *Modelo de datos para un sistema automatizado de control de costos de proyecto .* [En línea] septiembre de 2004. [Citado el: 20 de marzo de 2013.] http://tesis.luz.edu.ve/tde_busca/arquivo.php?codArquivo=2235.

40.WebTaller. [En línea] [Citado el: 18 de enero de 2013.] http://www.webtaller.com/construccion/lenguajes/html/lecciones/que-es-xhtml.php .

41. *Tutorial de JavaServer Faces .*

42. **Ramos, Juan Alonso.** Adictos al trabajo. *Introducción a Ajax4jsf.* [En línea] [Citado el: 4 de febrero de 2013.] http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Ajax4Jsf.

43. *Vista de análisis usando técnicas de agrupamiento para el sistema.* **Ochoa Reyes, Alexeis Joel y Orellana García, Arturo .** Universidad de las Ciencias Informáticas.La Habana, Cuba : s.n., 2012.

44. **Suárez, Jose Manuel Sánchez.** Adictos al trabajo. [En línea] 2013. [Citado el: 4 de mayo de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro.

45. **Corbillón, Javier Oliver.** *Desarrollo de funcionalidades del módulo Facturación del sistema alas HIS orientado al consumo de productos farmacéuticos.* Habana.Cuba : s.n., 2012.

46. **Web Application. Plataforma J2EE.** JBoss Seam Framework. [En línea] febrero de 2008. [Citado el: 20 de mayo de 2013.] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/.

47. Softpedia . *JFreeChart.* [En línea] [Citado el: 20 de mayo de 2013.] http://www.softpedia.es/programa-JFreeChart--18622.html.

48. **Lorenzo, Amaya Alvarez y Mora Maure, Mirelio.** *Desarrollo de funcionalidades para la especialidad de Psicología del módulo Consulta Externa del sistema alas HIS.* La Habana.Cuba : s.n., 2012.

49. **Universidad de los Andes, Departamento de Sistemas.** Enterprise Java Bean 3. [En línea] [Citado el: 26 de enero de 2013.] http://sistemas.uniandes.edu.co/~isis3702/dokuwiki/lib/exe/fetch.php?media=principal:isis3702-ejb3.pdf.

50.**Ecured.** Hibernate. [En línea] [Citado el: 5 de febrero de 2013.] http://www.ecured.cu/index.php/Hibernate.

51.**Shannon, Bill.** oracle.com. [En línea] [Citado el: 25 de enero de 2013.] http://www.oracle.com/technetwork/articles/javase/shannon-qa-138710.html..

52. **Lucifer, Prometeo.** Java Runtime Environment – JRE. [En línea] [Citado el: 5 de febrero de 2013.] http://www.elleonplateadodeojosrojos.es/blog/java-runtime-environment-jre/.

53. *Submódulo denuncia del sistema de investigación e información policial. Modelado del negocio.* **Sanchez, Eduardo Alfonso.** Universidad de las Ciencias Informáticas.La Habana,Cuba : s.n.

54. **Ecured.** Eclipse, entorno de desarrollo integrado. [En línea] [Citado el: 25 de enero de 2013.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

55. **Scribd.** pstrgreSQL-Investigación. [En línea] [Citado el: 20 de enero de 2013.] http://es.scribd.com/doc/36570462/postgreSQL-investigacion.

56. **Dataprix.** Prestaciones. [En línea] [Citado el: 20 de enero de 2013.] http://www.dataprix.com/12-prestaciones.

57. Guía de Ubuntu. [En línea] [Citado el: 29 de enero de 2013.] http://www.guia-ubuntu.com/index.php?title=PgAdmin_III&printable=yes.

58. **Alférez Sánchez, José A.** *instalación, configuración y administración del Servidor de aplicaciones JBoss.*

59. Open ERP. [En línea] [Citado el: 20 de Mayo de 2013.] http://openerpspain.com/ck14-articulos/ck20-anexos/ireport/.

60. **Campderrich Falgueras Benet, Maspons Ramon.** *Ingeniería del software.* Cataluña : s.n., 2003. 9788484297932.

61. **Jacobson, I. and Rumbaugh, G. Booch y J.** *El Proceso Unificado de Desarrollo de software.*

62. Buenas Tareas. *Arquitectura del software.* [En línea] noviembre de 2010. [Citado el: 25 de febrero de 2013.] http://www.buenastareas.com/ensayos/Todos-Est%C3%A1n-Locos/7850850.html.

63. Modelo de Datos. [En línea] [Citado el: 20 de marzo de 2013.] http://definicion.de/modelo-de-datos.

BIBLIOGRAFÍA CONSULTADA

- **Arencibia, Joel y Bonet, Isis.** Predicción de resistencia a fármacos del VIH utilizando Multiclasificadores.
- **Belmonte Fernández, Oscar.** *Introducción al lenguaje de programación Java. Una guía básica.*
- **Colibrí Software.** CS Almaceness- Software Selección. [En línea] http://www.softwareseleccion.com/cs+almacenes-p-3197.
- **Espinoza, Humberto.** *PostgreSQL Una Alternativa de DBMS Open Source.* 2005.
- **Gatica Lara, Florina , Fernández Puerto, Fernando J y Hernández López, Ana María.** Manual de Introducción a la Informática Médica. *Sistemas de Información Hospitalaria.* [En línea] 2003. http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf.
- **Martín Rodríguez, Gloria .** *Modelos estructurales en el contexto de las series.* Universidad de La Laguna.Campus de Guajara: s.n.
- **Molinero, Luis M.** *Análisis de series temporales.* 2004.
- **Newton, Mark, y otros.** WildFly. *RichFaces.* [En línea] 2008. http://www.jboss.org/richfaces.
- **Pérez, Pedro Y. Piñero.** *Algorítmos genéticos breve monografía.* Universidad de las Ciencias Informáticas : s.n., 2004.
- **Rodríguez-Piñero, Piedad Tolmos.** Introducción a los algoritmos genéticos y sus aplicaciones. [En línea] http://www.uv.es/asepuma/X/J24C.pdf.
- **Sommerville, Ian.** *Ingeniería de Software.* Madrid: s.n., 2005. 84-7829-074-5.
- **Universidad politécnica de Cartagena.** *Series Temporales.Tema 5.* Cartagena: s.n.
- **Web Tutoriales.** Try and Catch. [En línea] http://www.webtutoriales.com/articulos/try-and-catch.
- **Xabier Basogain Olabe.** *Redes Neuronales y sus aplicaciones.* Escuela Superior de Ingeniería de Bilbao, EHU : s.n.

ANEXOS

1. Ejemplo Ilustrativo

En la siguiente tabla se relacionan los consumos asociados a un producto A, en los últimos seis años. Dichos consumos se efectuaron trimestralmente.

Nota: como el objetivo de este ejemplo es ilustrar el procedimiento a efectuar, solo se tomaron las observaciones correspondientes a los últimos seis años, pero es importante resaltar que la fiabilidad de la predicción será mayor a medida que exista mayor número de observaciones antecedentes.

Período	Años 2001-2006					
Trimestre	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**
I	15.0	19.0	10.0	18.0	18.0	18.0
II	20.0	8.0	11.0	14.0	19.0	23.0
III	13.0	12.0	8.0	15.0	23.0	22.0
IV	15.0	13.0	15.0	14.0	20.0	25.0

Tabla A.1 Observaciones correspondientes al período 2001-2006.

Sea *t* cada uno de los 24 trimestres comprendidos entre 2001-2006, o sea que *t* = 1 para el primer trimestre de 2001, *t* = 2 para el segundo trimestre, y así sucesivamente. Donde 1≤ t ≤ 24 para cada uno de los trimestres en cuestión.

El primer paso es encontrar la función de la tendencia *T(t)* que mejor se ajusta a los datos observados, para ello se emplea el método de los Mínimos Cuadrados.

$$T(t) = 0.3973\mathrm{t} + 11.1992 \qquad T(t) = 17.4871 - 1.0536\mathrm{t} + 0.0580\mathrm{t}^2$$

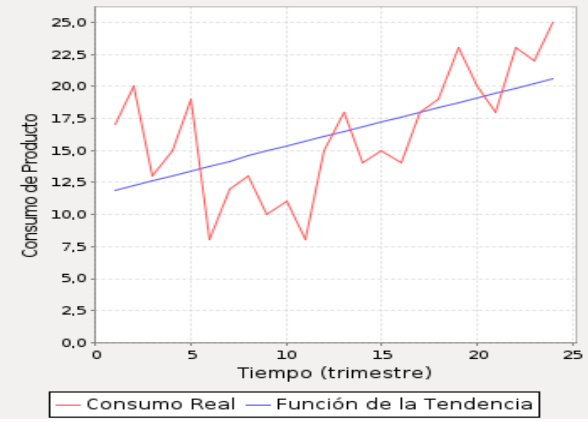


Figura A.1 Ajuste de la recta por el método MCO.

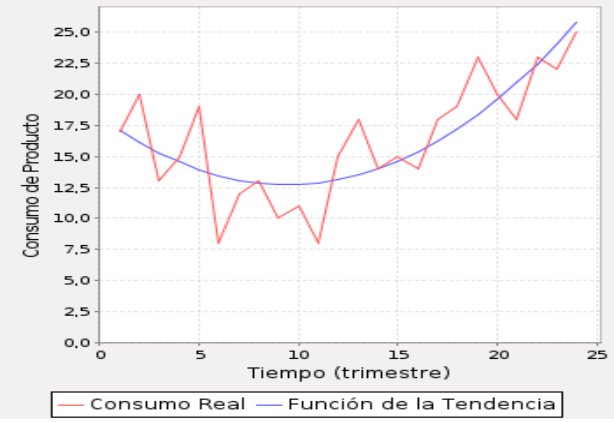


Figura A.2 Ajuste de la parábola por el método MCO.

Para saber cuál de las dos funciones se ajusta mejor se analizan los coeficientes de correlación *(r)*. Se conoce que*(r)* toma valores en el rango de −1 a 1 y a medida que sea más extremo (más se acerca a −1 ó a 1), significa que mejor se ajusta la curva. En este caso el cálculo de los coeficientes quedó de la siguiente manera: para la función lineal r=0.49, mientras que para la función cuadrática r=0.81.

Al analizar estos coeficientes se tiene que el mejor ajuste se obtiene a partir de la parábola de MCO, siendo esta la función de la tendencia: $T(t) = 17.4871 - 1.0536\mathrm{t} + 0.0580\mathrm{t}^2$. Una vez obtenida la función basta con evaluar en cada instante de tiempo para estimar (calcular) la tendencia $\check{T}(t)$, quedando de la siguiente manera:

Período	Años 2001-2006					
Trimestre	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**
I	16.4915	13.6699	12.7056	13.5986	16.3491	20.9568
II	15.6120	13.2547	12.7547	14.1121	17.3269	22.3989
III	14.8485	12.9556	12.9199	14.7417	18.4208	23.9572
IV	14.2011	12.7725	13.2013	15.4873	19.6307	25.6315

Tabla A.2 Valor de la tendencia en cada instante de tiempo.

Después de calculadas las tendencias se analiza el modelo. Para este caso, luego de haber analizado los CV, se concluyó que el modelo adecuado es el multiplicativo. La serie de Residuos $R(t)$ para este modelo queda de la siguiente manera:

Período	Años 2001-2006						
Trimestre	**2001**	**2002**	**2003**	**2004**	**2005**	**2006**	$\check{\mathrm{R}}(\boldsymbol{h})$
I	0.9095	1.3899	0.7870	1.3236	1.1009	0.8589	1.0616
II	1.2810	0.6035	0.8624	0.9920	1.0965	1.0268	0.9770
III	0.8755	0.9262	0.6191	1.0175	1.2485	0.9183	0.9342
IV	1.0562	1.0178	1.1362	0.9039	1.0188	0.9753	1.0180
							$\check{\mathrm{R}} = \mathbf{0.9977}$

Tabla A.3 Serie de residuos obtenida.

Una vez calculados los residuos se estima la Estacionalidad $\check{\mathrm{E}}(t)$:

$\check{\mathrm{E}}(1) = 1.0616 - (0.9977 - 1) =$1.0639 $\quad \check{\mathrm{E}}(2) = 0.9770 - (0.9977 - 1) = 0.9793$

$\check{\mathrm{E}}(3) = 0.9342 - (0.9977 - 1) = 0.9364 \quad \check{\mathrm{E}}(4) = 1.0180 - (0.9977 - 1) = 1.0203$

Después haber calculado los componentes de la serie pueden realizarse predicciones:

$X(n+k) = \check{\mathrm{T}}(n+k)\ x\ \check{\mathrm{E}}((n+k) mod S)$, donde: n es la cantidad de observaciones y k es el período para el cual se quiere predecir.

Por ejemplo, se desea predecir el consumo del producto A para dentro de dos trimestres. Así, k= 2 y n= 24; de esta forma se tiene que el período en cuestión se corresponde al segundo trimestre a partir de la última observación, por tanto, la estacionalidad que corresponde es: $\check{\mathrm{E}}(2) = 0.9793$.

$X(26) = (17.4871 - 1.0536\ 26\ + 0.0580(26)^2)\ x\ 0.9793$

$X(26) = 28.72$, por lo que para dentro de 2 trimestres el consumo del producto A será aproximadamente de 28.72 U.

Nota: Entre más próximo sea el período a estimar en correspondencia con el último dato observado, más exacta será la predicción.

2. Principales interfaces del componente de predicción de consumo

Figura A.3 Crear predicción.

Figura A.4 Buscar predicción en historial.

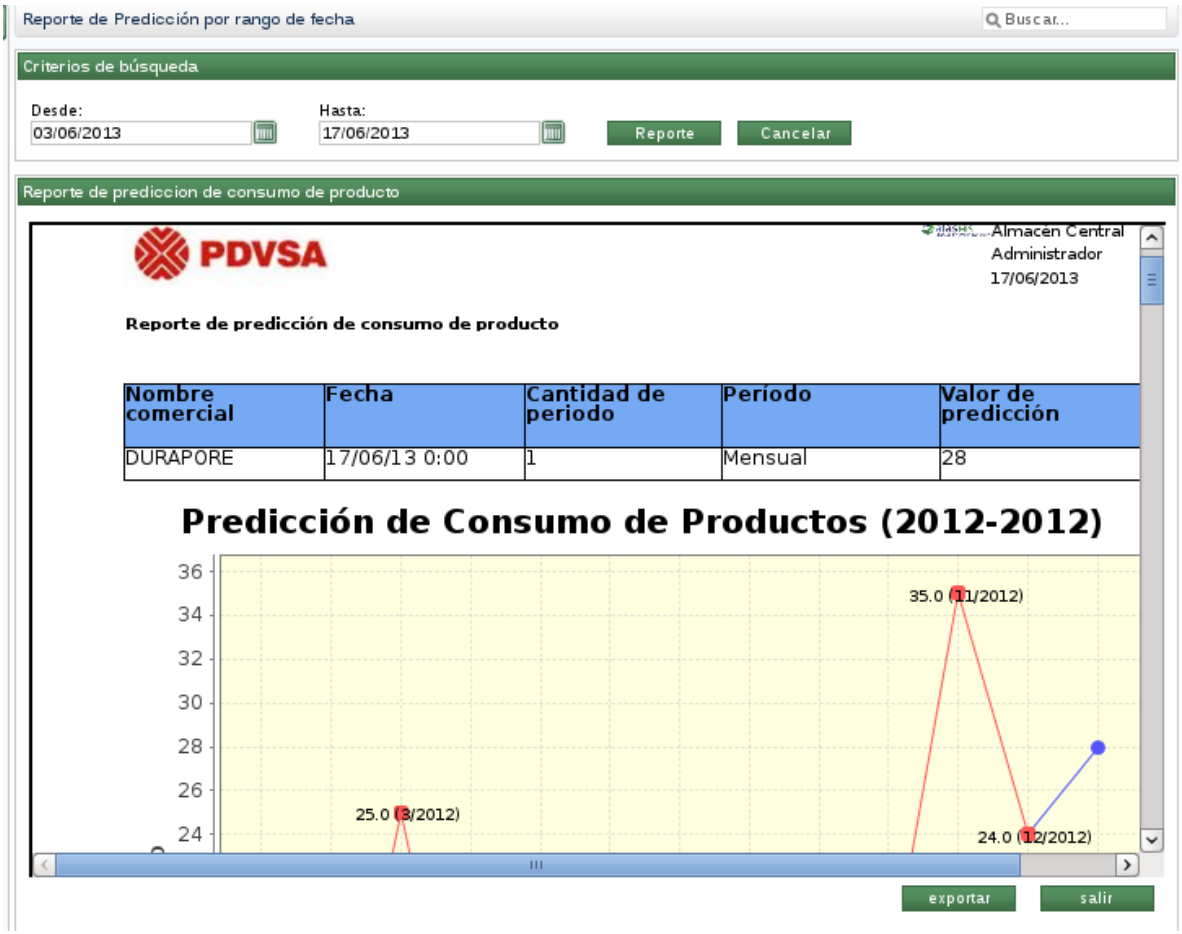


Figura A.5 Generar reporte por rango de fecha.

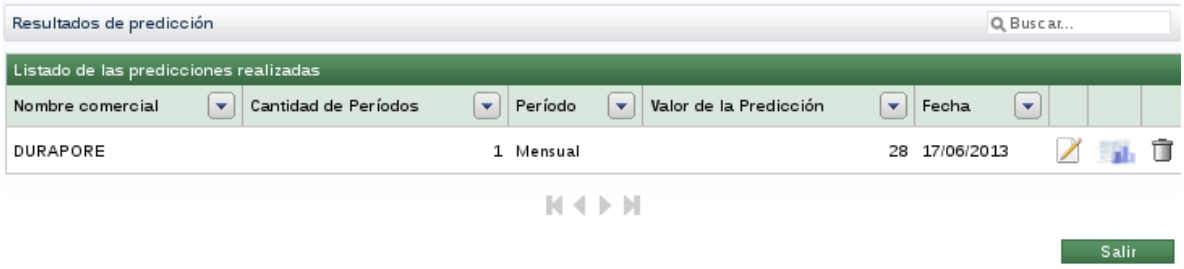


Figura A.6 Ver resultados de predicción.

Figura A.7 Modificar predicción

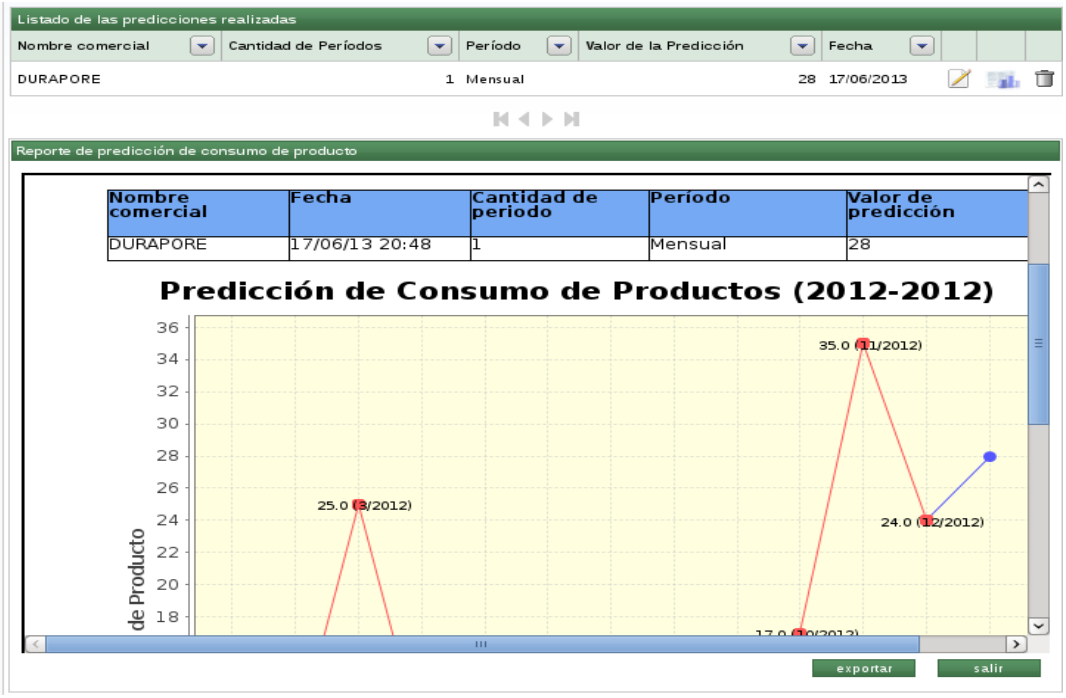


Figura A.8 Generar reporte de predicción.

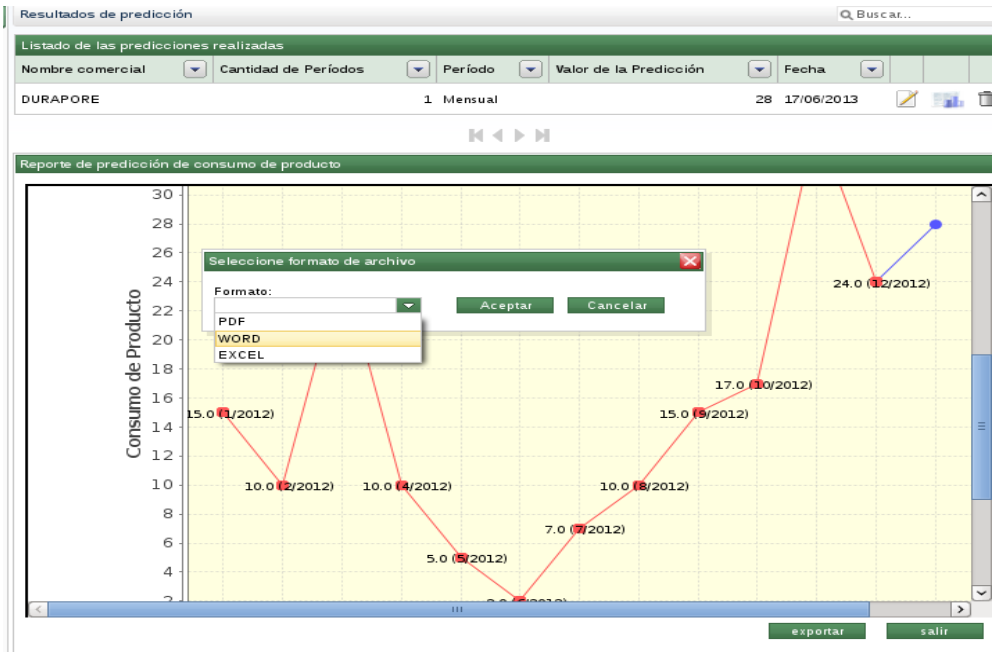


Figura A.9 Exportar reporte de predicción.

GLOSARIO DE TÉRMINOS

- **API** (Application Programming Interface): conjunto de funciones y procedimientos que poseen algunas librerías con el objetivo de ser utilizadas por otro software como una capa de abstracción.
- **Bean:** componente *software* que puede ser reutilizado, además puede ser manipulado visualmente por una herramienta de programación en lenguaje Java.
- **Dataset:** es una colección de datos que por lo general se presentan en forma tabular, donde cada columna representa una variable en particular y cada fila corresponde a un miembro determinado del conjunto de datos en cuestión.
- **Extrapolar:** suponer que el curso de los acontecimientos continuará en el futuro.
- **Framework:** estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.
- **Lex:** programa para generar analizadores léxicos (en inglés scanners o lexers).
- **LPT:** puerto paralelo para la conexión entre una computadora y la impresora, u otro dispositivo.
- **Middleware:** es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones que son necesarias en los sistemas distribuidos.
- **Multiplataforma:** término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.
- **Plain Old Java Object (POJO):** enfatiza el uso de clases simples y que no dependen de un *framework* en especial.
- **Stock:** este término se utiliza para referirse a los artículos que permanecen en un almacén, a la espera de una posterior utilización. Son recursos ociosos que tienen un valor económico y que están pendientes de ser vendidos o empleados en el proceso productivo.
- **TCP-IP:** conjunto de protocolos de red que permiten la transmisión de datos entre redes de computadoras.
- **UI:** Interfaz de usuario.
- **USB:** puerto que sirve para conectar periféricos a una computadora.
- **Yacc:** programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, "Otro generador de compiladores más".