Universidad de las Ciencias Informáticas

Facultad 3

Componente de gestión de nomencladores de la Ventanilla Única de Comercio Exterior de Cuba.

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

Autor: Daniel Arturo Casals Amat

Tutor: Ing. Leonardo D. Antúnez Naranjo

La Habana, Junio 2014

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco 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_____.

Daniel Arturo Casals Amat

_____

Firma del Autor

Ing. Leonardo D. Antúnez Naranjo

_____

Firma del Tutor

Datos de contacto

Autor:

Daniel Arturo Casals Amat

Correo Electrónico: dacasals@estudiantes.uci.cu

Tutores:

Leonardo D. Antúnez Naranjo.
Correo Electrónico: ldantunez@uci.cu

Agradecimientos

Agradezco a mi familia por el apoyo durante todos estos años, principalmente a mis padres y a mi hermano, gracias por ser perfectos.
A mi novia por todos los buenos momentos que hemos compartido y por siempre estar ahí para mí, eres mi alegría.
A mis compañeros de grupo y amigos por compartir conmigo estos 5 años que han sido los mejores de mi vida.
A todos los profesionales de la VUCEC, por toda la ayuda y todo el conocimiento que he adquirido durante este año. Por compartir como amigos sin descuidar el profesionalismo que debe existir en cualquier grupo de trabajo.
A todos los profesores que me han impartido clases por contribuir de una forma u otra a mi formación profesional.
A la REVOLUCIÓN por la oportunidad de realizar este sueño.

Dedicatoria

Especialmente a mi abuela Oneida que es el mejor ejemplo de lucha y sacrificio que conozco en pos de la felicidad de la familia.

A mis padres y a mi familia en general.

Resumen

La Ventanilla Única de Comercio Exterior de la República de Cuba es un sistema encargado de gestionar las operaciones comerciales en nuestro país, permitiendo que todas estas actividades se realicen a través de un único punto de acceso. Las operaciones llevadas a cabo en la VUCEC requieren de la utilización de nomencladores, estos son definidos por los Organismos de Administración Central del Estado o por los propios administradores de la VUCEC. Los nomencladores facilitan la flexibilidad del sistema y posibilitan una actualización rápida del mismo. La gestión actual de los nomencladores limita la consistencia de los datos y afecta el rendimiento general del sistema.

La presente investigación tiene como objetivo desarrollar un componente que garantice la consistencia de los datos asociados a los nomencladores y que contribuya a mejorar el rendimiento del sistema. Para su cumplimiento se realizó un estudio de soluciones informáticas asociadas a la gestión de nomencladores, se utilizó el modelo de desarrollo del Centro de Informatización de la Gestión de Entidades (CEIGE), y el conjunto de tecnologías definidas por el proyecto, destacando a Symfony2 como marco de trabajo.

Como resultado de la investigación se obtuvo un componente capaz de gestionar los nomencladores de la VUCEC dinámicamente a través de una interfaz única. Este controla la vigencia de los mismos, emitiendo alertas y mostrando reportes a los administradores para el mantenimiento del sistema. Garantiza además la consistencia de los datos asociados a los nomencladores y brinda servicios internos para el consumo de los demás componentes de la VUCEC.

Palabras claves: consistencia, nomencladores, vigencia, rendimiento.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con los avances científico-técnicos logrados internacionalmente y el auge de las Tecnologías de la Informática y las Comunicaciones (TIC), se han incrementado las operaciones comerciales a nivel mundial. El desarrollo del comercio internacional a partir de la explotación de las potencialidades de internet ha propiciado el surgimiento de nuevas estrategias que tienen como objetivo crear mayores oportunidades para favorecer el intercambio comercial.

En Cuba, por sus condiciones político-económicas, se tiene conciencia de la necesidad de favorecer este tipo de actividades comerciales, así como establecer condiciones necesarias para que los distintos trámites que se realizan se desarrollen con dinamismo, libres de trabas burocráticas que lo afecten. Con este objetivo surge la idea de implementar un sistema de Ventanilla Única de Comercio Exterior (VUCE), el cual se encuentra en proceso de desarrollo actualmente como uno de los proyectos que lleva a cabo la Universidad de las Ciencias Informáticas (UCI).

La VUCE se conceptualiza como "un sistema integrado que permite a las partes involucradas en el comercio exterior y transporte internacional gestionar, a través de medios electrónicos, los trámites requeridos por las entidades competentes de acuerdo con la normatividad vigente, o solicitados por dichas partes, para el tránsito, ingreso o salida del territorio nacional de mercancías".(1)

La Ventanilla Única de Comercio Exterior de la República de Cuba (VUCEC) es un sistema encargado de la gestión de las operaciones comerciales que tienen lugar en el país, el cual permite que todas estas actividades se realicen a través de un único punto de acceso que constituye la cara comercial de Cuba ante el mundo.

Las operaciones llevadas a cabo en la VUCEC requieren de la utilización de una serie de datos que constituyen un sistema de clasificación y de codificación para designar conceptos fundamentales tales como países, aranceles, tarifas, monedas, plazos, aduanas, modos de transporte, ministerios, formas de pago, entidades, etc., que facilitan la flexibilidad del sistema y la posibilidad de actualización rápida. Estos datos, en lo adelante denominados **nomencladores o codificadores**, son almacenados en **Tablas de Control** y son establecidos por los Organismos de la Administración Central del Estado (OACE) o por los propios administradores de la VUCEC y generalmente tienden a variar poco en el transcurso de un año.

Sin embargo, la propia naturaleza de la VUCEC dinamiza sobremanera el proceso de creación de nuevos tipos de nomencladores y por consiguiente surge la necesidad de contar con un mecanismo para la gestión de los mismos.

Se considera que los datos son consistentes si se controla la redundancia y si se garantiza que estos ofrecen en todo momento información real del sistema. Debido a la prioridad de implementar algunas funcionalidades de la VUCEC como la recepción de documentos y solicitudes no se implementó con anterioridad un mecanismo que determinara la vigencia de determinado nomenclador, inhabilitara los datos que dependen del mismo si fuera necesario y que permitiera emitir alertas sobre la vigencia de este. Este hecho provoca que con el paso del tiempo surjan inconsistencias en los datos asociados a los nomencladores.

Por otro lado, los datos que son establecidos por los OACE no siguen ningún patrón o estructura común lo que dificulta la gestión de los nomencladores e imposibilita brindar servicios para que sistemas autorizados puedan gestionar los nomencladores sin necesidad de la intervención de los administradores de la VUCEC.

El rendimiento de la VUCEC está condicionado por la capacidad de generar una respuesta a las peticiones de los usuarios dentro de un tiempo esperado. La consultas implementadas en la versión actual de la VUCEC relacionadas con los nomencladores no tienen en cuenta técnicas de optimización lo que provoca que dichos desarrollos incidan negativamente en el rendimiento general del sistema.

A las dificultades planteadas anteriormente se suma que los administradores necesitan a menudo conocer las relaciones que presentan las tablas nomencladoras con las demás tablas de la VUCEC así como identificar en qué medida es utilizado determinado nomenclador con el objetivo de realizar tareas de mantenimiento. Obtener esta información en el estado actual del sistema es una tarea engorrosa, pues se necesita tener un gran dominio del sistema, interactuar directamente con la Base de Datos o revisar el código fuente. En estas condiciones se destina tiempo, recursos humanos y de cómputo para realizar tareas que suelen ser repetitivas en lugar de ser destinados a otros tareas más significativas para el desarrollo del sistema.

Teniendo en cuenta lo antes expuesto, se plantea el siguiente **problema a resolver:** la gestión de los nomencladores de la Ventanilla Única de Comercio Exterior de Cuba limita la consistencia de los datos y afecta el rendimiento del sistema.

Se define como **Objeto de estudio** de esta investigación**:** el proceso de gestión de nomencladores.

El **Campo de acción abarca:** el proceso de gestión de nomencladores en el sistema VUCEC.

Para dar solución al problema planteado se precisa como **Objetivo General:** desarrollar un componente

que garantice la consistencia de los datos asociados a los nomencladores de la VUCEC y que contribuya a mejorar el rendimiento del sistema.

Idea a defender

Si se desarrolla un componente que gestione de forma centralizada los nomencladores de la VUCEC teniendo en cuenta la vigencia de los mismos, se contribuirá a mejorar el rendimiento del sistema y permitirá elevar la consistencia de los datos asociados a los nomencladores.

Para cumplir el objetivo general de la investigación se plantean los siguientes Objetivos Específicos:

1. Establecer el marco conceptual de referencia.
2. Diseñar e implementar el componente de gestión de los nomencladores.
3. Validar la solución.

Para dar cumplimiento a los objetivos específicos planteados se proponen las siguientes **Tareas de la investigación:**

1. Recopilación de bibliografía referente a los estándares de gestión de nomencladores.
2. Selección de la bibliografía.
3. Análisis de la bibliografía.
4. Análisis de los procesos de gestión de nomencladores.
5. Formalización del estándar de acceso controlado a los nomencladores.
6. Modelado del diseño de la aplicación.
7. Implementación del componente de gestión de nomencladores de la VUCEC.
8. Implementación del generador de reportes y estadísticas del componente de gestión de nomencladores de la VUCEC.
9. Validación de los resultados obtenidos mediante la aplicación de métricas de calidad, pruebas funcionales y pruebas de rendimiento.
10. Integración del componente a la arquitectura base de la VUCEC.
11. Exposición de los resultados.

Métodos de investigación.

En el desarrollo de la investigación se utilizan los siguientes métodos:

Análisis y síntesis: para el análisis de los distintos factores que dan origen a la situación problemática de forma independiente y a través del proceso de síntesis establecer las relaciones e interacciones que tienen estos factores entre sí.

Inductivo – deductivo: para identificar generalidades a partir del estudio de un conjunto de soluciones informáticas asociadas a la gestión de nomencladores y deducir características particulares que debe poseer el componente resultante de esta investigación.

Observación: es un registro visual del funcionamiento del entorno real en el que se desarrolla el problema planteado y permite identificar aspectos significativos para su solución.

A continuación un resumen de cada capítulo:

Capitulo1: Fundamentación teórica: en este capítulo se explican las metodologías, lenguajes y herramientas utilizados para el desarrollo de la aplicación e incluye un estudio de algunas soluciones informáticas asociadas a la gestión de nomencladores.

Capitulo2: Propuesta de solución: en este capítulo se realiza un breve análisis del funcionamiento actual del negocio, características del componente de gestión de nomencladores y detalles relacionados con el diseño.

Capitulo3: Implementación y validación de la solución: en este capítulo se detalla el resultado final de la investigación, características esenciales así como el aporte práctico del componente de gestión de nomencladores. Se muestran los elementos por los que se validó la solución implementada.

1. CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1. Introducción del capítulo

En el presente capítulo se abordan los aspectos fundamentales que permiten la descripción teórica del componente a implementar. Se analiza el modelo Entidad-Atributo-Valor (Entity-Attribute-Value, EAV). Se realiza un estudio de algunas soluciones informáticas del ámbito nacional e internacional que realizan la gestión de nomencladores, identificándose características que puedan ser valiosas para la definición del componente, además se describen las tecnologías utilizadas para el desarrollo del componente.

1.2. Elementos conceptuales de partida

Nomenclador:

Según la Real Academia Española, un nomenclador es un catálogo que tiene la nomenclatura de una ciencia. A su vez el Diccionario Enciclopédico Larousse lo define como un catálogo de nombres de carácter técnico u oficial. En la actualidad este es un término muy utilizado en las distintas esferas económicas y sociales, pues nomenclar es una forma organizada de clasificar, agrupar y estandarizar información.

Es por ello que en el ámbito de esta investigación se define como nomencladores a un conjunto de datos que constituyen un sistema de clasificación y codificación para designar conceptos como: países, aranceles, tarifas, monedas, plazos, aduanas, modos de transporte, ministerios, formas de pago, entidades, etc.

Componente

Un componente es una parte reemplazable, casi independiente y no trivial de un sistema que cumple una función clara en el contexto de una arquitectura bien definida. (23) Por lo general los componentes son diseñados de forma que puedan ser reutilizados en distintas aplicaciones, estos exponen interfaces que permiten utilizar sus funcionalidades sin necesidad de revelar detalles internos de los procesos que realizan. La solución propuesta se concibe como un componente teniendo en cuenta que debe presentar las características antes mencionadas. Otra de las razones es que el marco de trabajo (ver Symfony 2 en el epígrafe 1.4.5) utilizado por la VUCEC está formado por un conjunto de componentes reutilizables que facilitan el desarrollo del software, para dicho desarrollo se recomienda que las funcionalidades del sistema se organicen por bundles (concepto semejante al de componente).

A continuación se describen algunos patrones que se han tenido en cuenta para desarrollar la solución.

Patrón Softdelete (eliminación suave)

Softdelete es esencialmente la capacidad de ocultar una entrada en la Base de Datos en lugar de eliminar todo rastro de la entrada. Esto significa que los datos pueden ser recuperados para su uso en una fecha posterior. Otra forma de verlo es como un mecanismo de archivado de los datos.(32) El mecanismo que se sigue para lograr esto es establecer una bandera en una tabla existente que indica que un registro se ha suprimido, en lugar de eliminar realmente el registro.

Algunos marcos de trabajo, entre los que se encuentra Symfony, implementan o pueden extender alguna implementación de este comportamiento, en el caso de Symfony puede ser incluido mediante el uso de extensiones de algún ORM[1]. Con el uso de Doctrine [2]en su versión 2, se logra este comportamiento agregando un campo de fecha ("deletedAt") que marca una tupla como eliminada en el momento en que es "eliminada", permaneciendo en la Base de Datos de forma oculta para posteriores consultas de recuperación de datos.

En esta investigación se propone el uso de este patrón, pues es precisamente el ideal para manejar la vigencia de los nomencladores registrados en el sistema, teniendo en cuenta que:

- Uno de los requisitos es que se mantenga un histórico de los nomencladores aun cuando estos pierdan su vigencia.
- Dada las relaciones que se establecen entre las tablas de nomencladores con las demás tablas de la VUCEC no es posible eliminar los nomencladores relacionados.

Patrón Entidad-Atributo-Valor (EAV)

El modelo EAV también conocido como Objeto-Atributo-Valor o Esquema Abierto, se usa en casos donde el número de atributos (propiedades, parámetros), usados para describir una entidad u objeto es potencialmente grande, pero que aplicados individualmente a una entidad concreta es pequeño. (35)

Este modelo se suele resolver sobre el modelo relacional con tres tablas: la de entidad, la de atributos posibles de dicha entidad y la de valores. La información se registra conceptualmente en la tabla de

[1] (Object Relational Mapping), es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de Base de Datos relacional utilizado en el desarrollo de una aplicación.

[2] Tipo de ORM.

valores, que relaciona la información a través de tres columnas: clave foránea de entidad, clave foránea de atributo y valor.

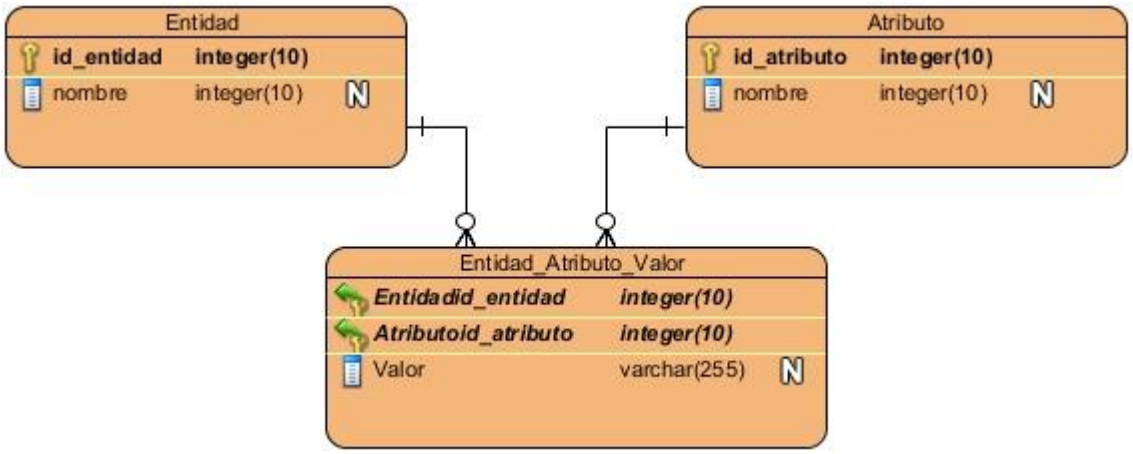


Fig. 1. Diagrama simple de EAV.

En principio, la adopción de este modelo resuelve el problema de gestionar los nomencladores de la VUCEC, un modelo simple como este permite registrar la información referente a los nomencladores, por lo que el mecanismo a implementar para gestionar los nomencladores en ciertos aspectos se facilita.

A pesar de las ventajas antes expuestas, este modelo presenta algunos problemas generales como:

- Afecta el rendimiento de acciones de recuperación de datos, la tabla que registra los valores de cada columna de las entidades aumenta sobremanera, debido al hecho de que se inserta una nueva tupla por cada atributo de cada instancia de la entidad que se quiera insertar, por lo que aumenta el tiempo de las consultas de recuperación de datos.
- Tipo de dato único para todos los atributos, sean numéricos, fechas o cadenas, que luego deberán convertirse, o complicar el modelo con distintas tablas de atributos en función del tipo de dato.

En la presente investigación se descarta el uso de este patrón por las siguientes razones:

- Una de las deficiencias de este patrón es que afecta el rendimiento de las consultas de recuperación de datos, el cual es uno de los objetivos fundamentales de la investigación.
- El modelo de datos que propone este patrón (utilizar tres tablas en lugar de una por cada tipo de nomenclador) implica redefinir todas las consultas ya creadas con anterioridad relacionadas con las tablas nomencladoras e influye directamente en el tiempo de desarrollo necesario.
- Con este modelo se desaprovechan una serie de ventajas que brinda el ORM de acceso a datos Doctrine en su versión 2, como: la validación de los campos de cada entidad, la facilidad con que

se implementan las consultas de recuperación de datos y el uso de los metadatos de las entidades.

- El uso de índices en las tablas no implica grandes mejoras en el rendimiento debido a que no existe una tabla para cada nomenclador definido en la VUCEC. El uso de índices para disminuir el tiempo de respuesta es un mecanismo muy utilizado en la VUCEC por sus beneficios los cuales se verían afectados por el uso de este patrón.

1.3. Soluciones informáticas estudiadas

A continuación se resumen las características fundamentales de algunas soluciones nacionales e internaciones asociadas a gestión de nomencladores atendiendo principalmente a:

- Tipo de licencia.
- Accesibilidad del código.
- Tecnologías utilizadas para su desarrollo.

1.3.1. Nomenclador del sistema Balance y Planificación de Insumos Médicos (alas BAP).

El Sistema para la Gestión de Nomencladores es una aplicación desarrollada en la UCI, a partir de la necesidad de corregir en un principio las deficiencias Módulo-Nomenclador del sistema alas BAP.

El Sistema para la Gestión de Nomencladores fue desarrollado utilizando Symfony como marco de trabajo de desarrollo de aplicaciones Web que implementa el patrón de arquitectura de software Modelo Vista Controlador (MVC). Utiliza el Mapeador de Objetos Relacional (ORM por sus siglas en inglés) Doctrine para el acceso a datos y el marco de trabajo JavaScript ExtJS 3.1 para generar las vistas. Es un sistema configurable y flexible ante posibles cambios en la gestión de nomencladores de un sistema, permite el almacenamiento y gestión de nomencladores y sirve como mecanismo de apoyo a otras aplicaciones para la gestión y utilización de nomencladores, brindando dicha información a través de servicios Web XML utilizando el protocolo SOAP[3], agilizando de esta manera el tiempo de desarrollo de estos sistemas. (2)

Las funcionalidades que brinda este sistema son:

- Gestionar los campos de los elementos nomenclados.
- Gestionar el código de cada nomenclador.
- Gestionar los grupos de los elementos a nomenclar.

[3] SOAP: por sus siglas en inglés Simple Object Access Protocol. Protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido.

- Asociar los campos a los nomencladores.
- Brindar servicios web que permitan el consumo de estos nomencladores por otras aplicaciones.
- Gestionar la información relacionada con cada nomenclador. (2)

El proceso de gestión de nomencladores que se desarrolla, basa su funcionamiento en el registro de los campos que son asociados luego a los nomencladores existentes, registrando finalmente, la información relacionada a cada nomenclador en una única interfaz, lo que agrega dinamismo al proceso.

Utilizar esta solución tiene una serie de inconvenientes entre los que se encuentra que las relaciones entre las tablas de control y demás tablas de la VUCEC las cuales están establecidas desde inicios del desarrollo del sistema impiden migrar la gestión de nomencladores a un sistema externo. Esta solución implicaría realizar un volumen importante de cambios que repercuten de manera negativa en el tiempo y los recursos humanos disponibles para esta investigación. Sumado a esto está el hecho de que no todos los problemas de esta investigación son resueltos, en tal caso se encuentra el control de la vigencia de los nomencladores el cual es uno de los problemas fundamentales de la VUCEC. Otra de las funcionalidades no satisfechas por este sistema es la obtención de reportes del estado de las relaciones de los nomencladores, por lo que habría que implementarla utilizando las tecnologías propias del sistema Nomenclador de Información las cuales son diferentes a las utilizadas por los desarrolladores de la VUCEC. Esto incide negativamente en el tiempo necesario para desarrollar la investigación.

Otra de las dificultades es que el sistema está patentado en la actualidad y el uso del mismo implicaría establecer un contrato oficial.

1.3.2. **OpenBravo**

Software desarrollado por la Universidad de Navarra en España, es un Sistema de Gestión Empresarial(Enterprise Resource Planning, ERP) de código abierto, para entornos web, dirigido a pequeñas y medianas empresas. (4)

Características:

- Implementado en el lenguaje Java.
- Aplicación completamente web que ha sido desarrollada siguiendo el patrón Modelo Vista Controlador (Model, View, Control, MVC).
- Soporte para Bases de Datos PostgreSQL y Oracle.
- Se ejecuta sobre un servidor Apache Tomcat. (4)

La funcionalidad fundamental de OpenBravo es su “Gestión de datos maestros” encargada de gestionar toda la información que será accesible al resto de los módulos, permitiendo de esta manera un intercambio de información estandarizada entre todos ellos, garantizando así la coherencia y consistencia de la información registrada en el sistema. Aquí se mantienen los datos de productos, personas, empresas, monedas, tarifas, proveedores, entre otros, que se van a usar en el resto de la aplicación. (4)
Otro de los patrones que utiliza OpenBravo es el MDD (Desarrollo Dirigido por Modelos) el cual supone un modelo de diseño de software que depende de metadata almacenada en un diccionario para modelar el comportamiento de la aplicación. A partir de este patrón y utilizando el WAD (Wizard for Application Development) de OpenBravo es que se genera todo el código de los CRUD[4] de gestión de los datos maestros del OpenBravo. (31)
La principal deficiencia de utilizar este sistema para la gestión de los nomencladores se encuentra en que está desarrollado en una plataforma diferente a la utilizada por el sistema VUCEC que impide la reutilización del código encargado de gestionar los datos maestros.
Una posible solución podría ser utilizar al menos los módulos bases de OpenBravo y otros necesarios incluyendo el módulo que gestiona los datos maestros como una solución independiente al sistema VUCEC, además se necesitaría definir los nomencladores en una Base de Datos externa a la de la VUCEC y modificar todas las operaciones de recuperación y gestión de datos sobre los nomencladores que ya están implementadas con anterioridad. Esta solución implicaría un costo elevado de tiempo y de recursos humanos, además del hecho de que condiciona a la VUCEC a depender de un sistema externo para su despliegue.

1.3.3. Cedrux[5]

El Sistema Integral de Gestión CEDRUX es desarrollado por la Universidad de las Ciencias Informáticas en coordinación con el Ministerio de Finanzas y Precios. El producto permite reconocer operaciones en varias monedas, registrando las tasas de cambio y realizando conversiones entre las mismas, es

[4] CRUD es el acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete). Se usa para referirse a las funciones básicas en Bases de Datos o la capa de persistencia en un software.

[5] Actualmente adoptó el nombre “XEDRO ERP”

multiusuario y multientidad. Es una solución genérica para la gestión de entidades, que es aplicable a todos los sectores del país, gestiona las áreas de Capital Humano, Contabilidad, Logística, Costos, Procesos y Finanzas. Es un sistema multiplataforma y esta implementado bajo licencias libres siguiendo el principio de independencia tecnológica del país. (5)

Para su desarrollo se utilizó el marco de trabajo Sauxe sobre el cual trabaja dicho proyecto. Algunas de las herramientas de este marco de trabajo son:

- Extjs 2.2: Librería construida con JavaScript que proporciona una interfaz cuya potencia radica en la rica colección de componentes para el diseño de interfaces del lado del cliente.
- Zend_Ext Framework: Es un marco de trabajo de código abierto, diseñado para PHP5 o superior. Se deriva de Zend Framework por lo que cumple con todas sus características. Tiene incorporado el ORM Doctrine para trabajo en la capa de abstracción a Base de Datos. (5)

En todos los módulos que componen el sistema Cedrux se gestionan los nomencladores referentes al mismo y se implementan servicios para que los demás puedan obtener acceso de manera controlada a la información que se registra en estas tablas. Un ejemplo de esto es el Módulo Estructura y Composición, el cual tiene como objetivo brindar al usuario la posibilidad de definir la estructura organizativa en la cual se va a ubicar su entidad y la estructura dentro de dicha entidad, así como permitir que se puedan especificar los cargos por los cuales van a estar compuestas las diferentes áreas dentro de las unidades. Además provee servicios al resto de los módulos presentes en el sistema Cedrux que lo necesiten. (7)

El paquete Gestionar Nomencladores se encarga de la gestión de algunos nomencladores cuya estructura en el sistema está bien definida, estos se gestionan de forma estática implementando una lógica específica para cada uno de ellos, entre estos nomencladores se encuentran: Órgano, Subcategoría, Nivel jerárquico, Tipo cifra, Cargo civil, Calificador, Cargo militar, Responsabilidades, Categoría ocupacional, Tipo de escala salarial, Grupo de complejidad, Escala salarial, Medios, Nivel de utilización entre otros.

Para cada nomenclador se crea una tabla cuyo nombre comienza con el prefijo “nom_” y le sigue el nombre de la tabla (ejemplo: nom_cargo_militar, nom_tipo_calificador), los campos comunes que comparten los diferentes nomencladores registrados son:

- código: es un identificador único para cada nomenclador.
- denominación: hace referencia al nombre o la información que va a ser mostrada al usuario.
- fecha Inicio: fecha de registrado el nomenclador en el sistema

- fecha Fin: fecha en que se elimina el nomenclador(vale aclarar que se utiliza el comportamiento Softdelete o eliminación suave nativo del ORM Doctrine, la tupla nunca se elimina físicamente del sistema, solo es marcada como eliminada permaneciendo invisible para próximas consultas de recuperación de datos)

El otro conjunto de nomencladores del sistema cuya estructura de atributos varía en dependencia de su tipo se gestionan de forma dinámica implementando el patrón EAV antes descrito. Se proveen funcionalidades diferentes para la gestión de las tablas, los campos y las relaciones entre las tablas.

La principal desventaja de utilizar este módulo es que la implementación realizada para gestionar los nomencladores sigue el patrón EAV rendimiento del sistema por las razones antes expuestas. Tampoco se generan reportes sobre los datos asociados a los nomencladores la cual es una de las necesidades de las que parte esta investigación.

No obstante la estructura para definir las tablas de registro de nomencladores (los campos comunes) es conveniente para la definición de la estructura de las tablas de control del VUCEC, será tenida en cuenta en posteriores fases de la investigación.

1.3.4. SAP[6]

Software desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. La corporación se ha desarrollado hasta convertirse en la quinta más grande compañía mundial de software. El sistema SAP está compuesto por diferentes módulos, cada módulo se encarga de brindar una solución específica sobre un área empresarial, los mismos interactúan entre si y comparten información.

Entre los módulos de SAP se encuentra el módulo Controlling (CO) el cual pertenece al sector de módulos Financieros de SAP, su tarea principal es la planificación contable, permitiéndole a las empresas que lo utilizan gestionar claramente su estructura de costos y ayudarle en la gestión de tomar decisiones relacionadas con el control de los costos del negocio.

SAP CO lleva a cabo su gestión mediante lo que se denomina en SAP "Transacciones". Las transacciones son programas que se encargan de cumplir una tarea en particular dentro del sistema SAP. Para utilizar las transacciones de SAP CO, también es necesario definir ciertos **datos maestros.**

[6] (Systeme, Anwendungen und Produkte) (Sistemas, Aplicaciones y Productos). El nombre es al mismo tiempo el de una empresa y el de un sistema informático.

Estos datos están constituidos por Información almacenada en Bases de Datos del sistema SAP los cuales contienen información que no cambia con frecuencia, como por ejemplo el porcentaje del IVA.(6)
Para comenzar a trabajar con las transacciones de SAP CO se deben definir algunos datos maestros esenciales como:

- **Centro de costo:** es una unidad de negocio, tal como una unidad de producción que incurre en gastos dentro de la empresa. Se asigna un código de centro de costo a cada unidad de negocios de la empresa. Los costos incurridos por una unidad de negocio son registrados bajo su código de centro de costo correspondiente.
- **Elemento de costo:** describe el origen de los costos dentro de la empresa. Por ejemplo, los gastos incurridos en salarios, teléfono y electricidad son considerados como costos iniciales dentro de la empresa. Cada transacción relacionada con estos costos registrará bajo un elemento de costos los gastos.
- **Centro de beneficio:** es una unidad de negocios tal como una unidad de venta que obtiene ingresos dentro de la empresa. Un código de centro de beneficio es asignado a cada unidad de negocios. Los ingresos obtenidos por cada unidad de negocios son registrados bajo su correspondiente código de centro de beneficio.(6)

Principales inconvenientes:

- Es un programa privativo, con elevado coste de licencia.
- No ofrece la posibilidad de acceder al código fuente del producto para su personalización.
- No se ajusta a las necesidades de la VUCEC.

1.3.5. GINA

GINA es un sistema Web para la Aduana General de la República de Cuba (AGR). Esta solución es multiplataforma por lo puede ser utilizado tanto en sistemas operativos basados en GNU/Linux como en las versiones del sistema operativo Windows. A su vez son multiplataforma también el lenguaje utilizado PHP, los marcos de trabajo Symfony y ExtJs y el sistema gestor de Base de Datos Oracle.
El GINA está compuesto por varios módulos entre los que se encuentra Tabla de Control (TC) siendo indispensable para el funcionamiento de los diferentes módulos del sistema que necesitan acceder a estos datos de forma flexible mediante una interfaz única. Los datos son introducidos con posibilidad de ser modificados, es válido aclarar en el caso de las modificaciones y eliminaciones que estas se realizan de

forma lógica actualizando sólo la fecha de vencimiento del artículo, esto ocurre en todas las tablas debido a que por requerimientos del sistema, la información se debe almacenar durante cinco años.

El módulo TC permite:

- Gestionar los nomencladores y codificadores dinámicamente mediante una interfaz única.
- Actualizar la información manteniendo el histórico y la consistencia de los datos.
- Obtención de reportes para consultas.

Para el desarrollo de TC se tuvo en cuenta que:

- Los campos comunes para todas las tablas son Código, Descripción, Fecha de Inicio, Fecha de Fin, Created_At (fecha de creación de un registro) y Deleted_At (fecha de eliminación de un registro).
- Los artículos en las tablas tienen incluidos los campos de fecha f_inicio y f_fin para controlar su validez. Esto se debe a que no pueden ser eliminados, solo se actualiza la fecha de fin y este continúa en la tabla para efectos de auditoria posteriores.
- Las fechas de inicio y fin de validez son validadas al ser introducidas en el sistema y no se permite introducir un artículo con vigencia anterior a la fecha en que se está haciendo la actualización. ni insertar artículos cuya fecha de inicio sea menor que la actual.
- Los campos Created_At y Deleted_At son llenados automáticamente con la fecha de creación del registro (fecha actual del sistema) y la fecha de eliminación del registro (infinito 31/12/9999).
- Si las tablas se relacionan con otros datos de otras tablas en una relación, en el momento de la actualización la relación se actualiza también, o sea se debe garantizar la consistencia de los datos.

Este componente de Gestión de Tablas de Control perteneciente al sistema GINA[7] (uno de los proyectos pertenecientes al mismo departamento al que pertenece el proyecto que desarrolla el sistema VUCEC) es una de las fuentes principales de estudio para el desarrollo de esta investigación. Teniendo en cuenta que los sistemas GINA y VUCEC intercambian información mediante servicios WEB que utilizan el protocolo SOAP y comparten ORACLE como Sistema Gestor de Base de Datos, la estructura de definición de las tablas de control de la VUCEC va a ser implementada siguiendo la mayoría de las buenas prácticas adoptadas por el GINA para su componente de Gestión de Tablas de Control.

El código del componente puede ser accedido para su estudio pero no es reutilizable dado que este trabaja con tecnologías diferentes a la del sistema VUCEC (desarrollado en una versión inferior del marco

[7] Sistema Integral de Gestión Aduanera.

de trabajo Symfony, la cual es incompatible a la versión 2.1 que utiliza el VUCEC y el trabajo con las vistas está realizado con el marco de trabajo ExtJs el cual es distinto al utilizado por la VUCEC).

Utilizar el módulo Tablas de Control del sistema GINA como un sistema independiente que gestione los nomencladores de la VUCEC crea una dependencia no deseada que pone en riesgo la disponibilidad de la VUCEC. Si el módulo no se encuentra disponible en algún momento se afecta también la disponibilidad de la VUCEC pues esta depende de los nomencladores para realizar sus operaciones.

Este módulo no provee un mecanismo a través de la interfaz del software para conocer las relaciones que presentan los nomencladores registrados o el estado de los mismos. Dicha funcionalidad habría que implementarla utilizando las tecnologías de desarrollo del GINA, lo cual incide en el tiempo de desarrollo de manera negativa, pues se necesita dominar las mismas antes de proceder con su desarrollo.

A lo anterior se suma que el marco de trabajo Symfony utilizado por el GINA dejó de recibir soporte desde noviembre del 2012 por lo que todas las brechas de seguridad detectadas a partir de esa fecha son problemas a los que se enfrentan las aplicaciones que lo utilizan.

	SAP	OpenBravo	Sistema Nomenclador de Información	Cedrux	GINA
Tipo de licencia.	Licencias privativas. Elevado coste.	Openbravo Public License Versión 1.1 (Adaptación de Mozilla Public License)	Presenta una patente registrada en el CENDA (Centro Nacional de Derecho de Autor).	No presenta restricciones por licencia.	No presenta restricciones por licencia.
Acceso al código	No.	Acceso parcial.	Imposible sin establecer un contrato oficial.	Si	Si
Tecnologías utilizadas para su desarrollo	NetWeaver.	Java. Como SGBD PostgreSQL u Oracle.	Symfony v1.4. ExtJS v3.1 SGBD: PostgreSQL.	Zend_Ext Framework. ExtJS v2.2. PostgreSQL.	Symfony v1.4. ExtJS v3.0 SGBD: Oracle 11G.
Soporte de las	Si	Si	No	Si	No

tecnologías					

1.4. Herramientas, lenguajes y modelo de desarrollo

1.4.1. Modelo de desarrollo

El desarrollo de software no es una tarea fácil, prueba de ello es que existen numerosas propuestas metodológicas tanto ágiles como tradicionales que inciden en distintas dimensiones en el proceso de desarrollo. (8)

Las instituciones que llevan varios años en la producción de software han obtenido gran experiencia utilizando todas estas metodologías de desarrollo, llevándolas a crear modelos propios que definen las fases del ciclo de vida de los proyectos que se van a desarrollar. El Centro de Informatización de la Gestión de Entidades (CEIGE) cuenta con un modelo de este tipo que incluye la especificación de las actividades de cada una de las fases del ciclo de vida de los proyectos del centro, teniendo en cuenta los procesos de Integración de Modelos de Madurez de Capacidades (CMMI[8]) nivel dos para la UCI. Se detallan por tanto los artefactos a generar en cada momento, independientemente de las herramientas o métodos que se utilicen para ello, en el Modelo de Desarrollo de Software Versión 1.2 (9)

A continuación se detallan los artefactos a generar (solo los incluidos en este documento) por cada una de las **disciplinas** definidas en el Modelo de desarrollo:

Disciplina Modelado del Negocio: Modelo conceptual.

Disciplina Requisitos: especificación de Requisitos de software, Descripción de requisitos de Software, Criterios para validar requisitos del cliente.

Disciplina Análisis y Diseño: Modelo de datos, Diccionario de Datos, Diagrama de Clases del componente, Diagrama de Clases del diseño con estereotipos web, Diseño de Casos de Pruebas basado en Requisitos.

Disciplina Implementación: Diagrama de componentes.

Disciplina Pruebas internas: (no se incluyen los artefactos generados durante esta disciplina).

1.4.2. Lenguaje de modelado: Lenguaje Unificado de Modelado (UML)

Este lenguaje prescribe un conjunto de notaciones y diagramas estándares para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan; posibilitando así visualizar, especificar y documentar los artefactos o toda información que se obtiene o

[8] **CMMI:** por sus siglas en inglés Capability Maturity Model Integration.

modifica durante un proceso de desarrollo de software, además de poder utilizarse para modelar distintos tipos de sistemas de software, hardware y organizaciones del mundo real.(10)

Principales características:

- Posibilita mejores tiempos totales de desarrollo.
- Permite modelar sistemas, no sólo de software.
- Tecnología orientada a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

1.4.3. Herramienta Case: Visual Paradigm para UML 8.0

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, haciéndolas mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.(30)

1.4.4. Lenguajes de programación

1.4.4.1. PHP (PHP Hipertext Preprocesor)

PHP es el acrónimo recursivo que significa PHP Hypertext Pre-processor, es un lenguaje de programación del "lado del servidor"[9] gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.(11)

PHP es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, pues permite exportar el sitio desarrollado en PHP de un sistema a otro fácilmente. Este

[9] Es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a Bases de Datos, conexiones en red y otras tareas para crear la página final que verá el cliente.

lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado, dicha librería cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red.

Principales capacidades de PHP:

- Compatibilidad con las Bases de Datos más comunes, como MySQL, Oracle, Informix, y ODBC
- Incluye funciones para el envío de correo electrónico, subida de archivos.
- Para desarrollar en este lenguaje no se requiere tener grandes capacidades de hardware. Además tiene una de las comunidades más grandes en Internet, por lo que es fácil encontrar ayuda, documentación, artículos, noticias y demás recursos.(11)

1.4.4.2. JavaScript

Es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento, siendo uno de los más utilizados con este fin y que brinda muchas posibilidades, porque permite la programación de pequeños scripts, pero también de programas más grandes orientados a objetos con funciones, estructuras de datos, entre otras.

JavaScript pone a disposición del programador todos los elementos que forman la página web, para que este pueda acceder a ellos y modificarlos dinámicamente, posibilitándole el control total sobre la misma. (12)

jQuery 1.8.2

jQuery es una biblioteca o marco de trabajo de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.(33)

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. jQuery es un producto con una aceptación muy buena por parte de los programadores, la comunidad de creadores de plugins o componentes es muy amplia, lo que hace fácil encontrar soluciones útiles a muchas de las necesidades de un software en desarrollo.(26)

Bootstrap

Bootstrap es un marco de trabajo que permite crear interfaces web con CSS y JavaScript que adaptan la interfaz dependiendo del tamaño del dispositivo en el que se visualice de forma nativa, es decir, automáticamente se adapta al tamaño de un ordenador o de una Tableta sin que el usuario tenga que hacer nada, esto se denomina diseño adaptativo o Responsive Design. El marco de trabajo trae varios elementos con estilos predefinidos fáciles de configurar: botones, menús desplegables y formularios, incluyendo todos sus elementos e integración con jQuery para ofrecer ventanas y tooltips dinámicos. (27) Por si solo un marco de trabajo no aporta todas las soluciones a las necesidades de un sistema. Dentro de las mismas se pueden evidenciar la compatibilidad de navegadores, aumentar las facilidades del trabajo con interfaces, entre otras especificidades.

1.4.5. Marco de trabajo: Symfony 2.1

Es un marco de trabajo que ayuda a simplificar el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Proporciona además varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.(14)

Entre las características más destacadas que ofrece a los desarrolladores de productos de software se encuentran las siguientes:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).
- Independiente del sistema gestor de Bases de Datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.

- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo. (14)

1.4.6. Gestor de Base de Datos: Oracle 11g

Se utiliza Oracle 11g como Sistema Gestor de Base de Datos (SGBD) el cual es considerado uno de los más completos y potentes a nivel mundial, destacado por su soporte de transacciones, estabilidad, escalabilidad y ser multiplataforma. Utiliza la arquitectura cliente/servidor. Ha incorporado en su sistema el modelo objeto-relacional, pero al mismo tiempo garantiza la compatibilidad con el tradicional modelo relacional de datos. (28)

1.4.7. Entorno de Desarrollo Integrado (IDE): NetBeans 7.3

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para otros lenguajes de programación como PHP, JavaScript, Ajax, Groovy y Grails, y C / C + +. Existe además un número importante de módulos para extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de PHP o soporte para el sistema de control de versiones SVN. (13)

1.4.8. Servidor de Aplicaciones: Apache 2.2.2

Apache es el servidor web por excelencia, su facilidad de configuración, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Se ejecuta en gran cantidad de sistemas operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita, de código abierto y altamente configurable de diseño modular por lo que resulta muy sencillo ampliar las sus capacidades. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y es posible configurarlo para que ejecute un determinado script cuando esto suceda.(15)

1.4.9. Controlador de versiones: Subversion (SVN)

Es un sistema de control de versiones libre y de código fuente abierto. Gestiona un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. Pudiendo ser estos ficheros código fuente, archivos multimedia, o cualquier otro tipo de documento.(16)

1.5. Conclusiones del capítulo

Una vez realizado el estudio del marco teórico de la investigación se arribó a las siguientes conclusiones:

- El estudio de los principales conceptos referentes a los nomencladores y los patrones asociados a la gestión de los mismo permitió: desechar la utilización del patrón EAV debido a sus deficiencias de rendimiento en la recuperación de los datos; adoptar el comportamiento Softdelete implementado como extensión del ORM Doctrine2 para controlar la vigencia de los nomencladores evitando tener que redefinir las consultas realizadas.
- Se realizó un estudio de los sistemas OpenBravo, Sistema Nomenclador de Información, SAP, Cedrux y GINA que arrojó un conjunto de buenas prácticas a tener en cuenta para la definición del componente de gestión de nomencladores entre las que destaca la definición de un conjunto de campos comunes que deben poseer las tablas nomencladoras de la VUCEC.
- Se estableció como modelo de desarrollo a seguir el definido para los proyectos del CEIGE en su versión 1.2. Para la descripción y modelado del sistema se estableció la herramienta CASE Visual Paradigm para UML 8.0.
- Se definió como IDE de desarrollo el NetBeans 7.3 y como marco de trabajo Symfony en su versión 2.1. Se definió el uso de los marcos de trabajo Bootstrap 2.3 y JQuery 1.9 para el desarrollo y enriquecimiento de las vistas. Como Sistema Gestor de Base de Datos se estableció Oracle en su versión 11g.

2. CAPÍTULO 2. PROPUESTA DE SOLUCIÓN.

2.1. Introducción

La solución propuesta se sustenta sobre la base de lo expuesto en este capítulo, en la cual se definen las actividades más importantes que posibilitaron la modelación del negocio. Se detallan los requisitos de software identificados, el diseño de la aplicación, así como las características específicas del sistema.

2.2. Componente Nomencladores

El presente módulo surge debido a la necesidad de gestionar los nomencladores existentes mediante una interfaz única. La estructura y contenido de los nomencladores o codificadores se definieron a partir de los estudios de los sistemas previamente analizados, se siguen principalmente las buenas prácticas definidas para el módulo Tablas de Control del sistema GINA.

Los nomencladores que se almacenan deben tener la posibilidad de ser modificados, es válido aclarar que las modificaciones y eliminaciones debido al tipo de relaciones que se establecen con las demás tablas del sistema deben ser de forma lógica, actualizando sólo la fecha de expiración de la vigencia. Esto ocurre en todas las tablas que gestiona el componte, pues por requerimientos del sistema se guarda un histórico de la información que se almacena en estas tablas para mantener la consistencia del sistema. Se debe garantizar además que estas informaciones sean actualizadas por las personas autorizadas. Para garantizar el cumplimiento de su objetivo el módulo debe ser capaz de:

- Definir e implementar un mecanismo de recuperación de datos que tenga en cuenta la vigencia del nomenclador.
- Gestionar dinámicamente los nomencladores mediante una interfaz única.
- Actualizar la información manteniendo el histórico y la consistencia de los datos.
- Crear servicios para la obtención de datos de los nomencladores.
- Definir mecanismo para la sincronización de nomencladores con el GINA.
- Notificar mensajes de alerta para nomencladores cuya vigencia esté cercana a concluir.
- Generar reportes estadísticos sobre los datos almacenados por el tipo de nomenclador.

2.3. Generalidades

Para el desarrollo del componente Nomencladores se tuvieron en cuenta los siguientes aspectos:

Los campos comunes para todas las tablas son: Código, Descripción, Created_At (fecha de registro de un nomenclador) y Deleted_At (fecha de eliminación de un nomenclador registrado).

Ejemplo:

Tabla 1. Ejemplo de tabla nomencladora tc_estado_civil.

ID_ESTADO_CIVIL	CODIGO	DESCRIPCION	CREATED_AT	DELETED_AT
1	0001	Casado	21/10/2008	02/02/2022
2	0002	Soltero	21/10/2008	22/02/2022

Los nomencladores vigentes serán aquellos en los cuales el campo Created_At sea menor o igual que la fecha actual y Deleted_At mayor o igual que la fecha actual o vacía.

Las tablas de nomencladores tienen definidos los campos Created_At y Deleted_At con los cuales se controla su validez, debido a que estos no pueden ser eliminados, se actualiza la fecha de fin y este continúa en la tabla para efectos de auditorías posteriores.

Las fechas Created_At y Deleted_At son validadas al ser tecleadas y no se permite introducir un nomenclador cuya vigencia expire antes de la fecha en que se está realizando la actualización, a su vez, tampoco es permitido insertar nomencladores cuya fecha de inicio sea menor que la fecha actual.

2.4. Disciplina Modelado del negocio. Modelo de dominio

Un modelo de dominio o modelo conceptual explica los conceptos significativos y sus relaciones del dominio de un problema y constituye una entrada necesaria para varios de los artefactos que se generarán en el diseño de la solución. Se utilizan para representar la realidad a un alto nivel de abstracción y según Craig Larman en su libro “UML y Patrones” es el artefacto más importante que se crea durante el análisis orientado a objetos.

A continuación se muestra el modelo conceptual que engloba los conceptos más importantes y sus relaciones en el dominio de esta investigación.

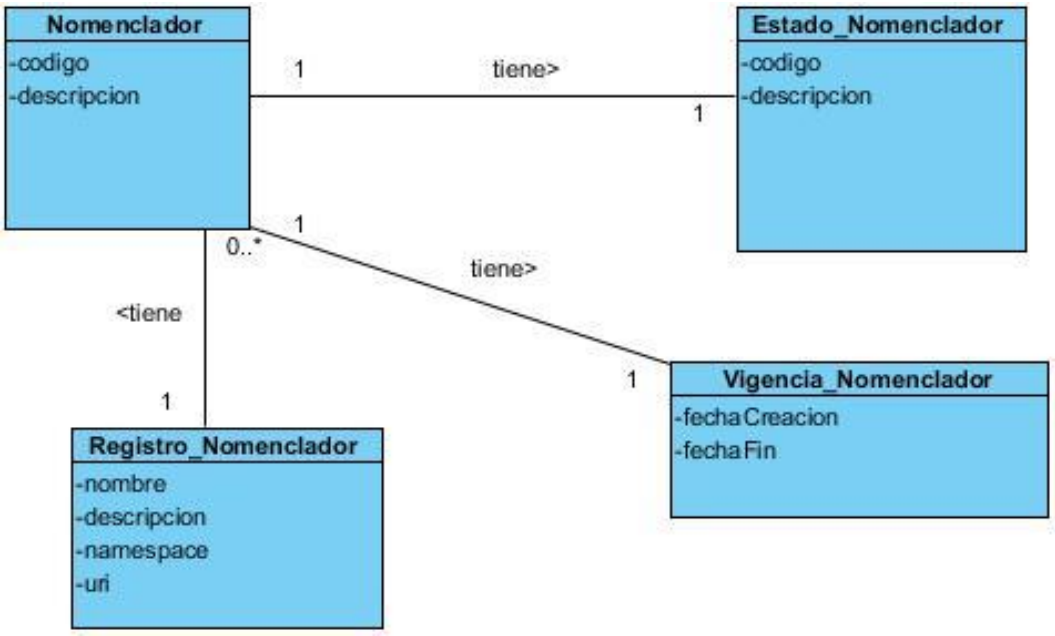


Fig. 2. Modelo Conceptual.

A continuación se describe una de las clases pertenecientes a este modelo:

Nomenclador

Descripción	Son los nomencladores por tipos a gestionar.					
Atributos						
Nombre	**Descripción**	**Tipo**	**¿Puede ser nulo?**	**¿Es único?**	**Restricciones**	
					Clases válidas	Clases no válidas
código	Código identificador del nomenclador.	String	No	Si	Caracteres numéricos 0..9	Caracteres alfabéticos A...Z, a...z. Caracteres Especiales, /?][';!~@#$%^&*()_+-.

descripción	Texto que describe el nomenclador.	String	No	Si	Caracteres alfabéticos A...Z, a...z.	Caracteres numéricos 0..9 Caracteres Especiales. / ?][';!~@#$%^&*()_+-.

2.5. Disciplina Requisitos. Requisitos de Software

2.5.1. Técnicas para la captura de requisitos

Los requisitos de software comprenden necesidades de información y control, funcionalidad del producto y comportamiento, rendimiento general del producto, diseño, restricciones de la interfaz y otras necesidades especiales.

Las técnicas utilizadas para la captura de requerimientos fueron: estudio de los sistemas existentes y tormenta de ideas. Se realizó un estudio del sistema VUCEC identificándose los requerimientos generales del mismo, además se acumularon ideas para tener una perspectiva general de las necesidades del componente en conjunto con el jefe de proyecto, el cliente y el analista principal.

2.5.2. Requisitos Funcionales

Los requerimientos o requisitos funcionales son capacidades o condiciones que el sistema debe poseer. Son un conjunto de características requeridas por el sistema, que expresan su capacidad de acción; una funcionalidad.(34)

Para el diseño e implementación del componente Nomencladores se identificaron los siguientes requisitos:

1. Gestionar nomencladores.
    - Adicionar nomencladores.
    - Modificar nomencladores.
    - Eliminar nomencladores.
    - Buscar nomencladores.
2. Filtrar nomencladores por estado de uso.
3. Notificar nomencladores cuya vigencia está al terminar.
4. Servicios para la obtención de datos de los nomencladores.
    - Obtener los datos de un nomenclador dado el código y el nombre de una tabla nomencladora.
    - Obtener todos los nomencladores vigentes dado el nombre de una tabla nomencladora.

- Obtener todos los nomencladores, dado el nombre de una tabla nomencladora y el estado del nomenclador.
- Obtener nomenclador, dado el nombre de una tabla nomencladora, código del nomenclador y estado del nomenclador.
- Actualización de nomencladores con el GINA.

5. Generar reportes estadísticos de nomencladores
    - Cantidad de tuplas por tipo de nomenclador.
    - Cantidad total de tuplas agrupadas por nombre de tabla nomencladora.
    - Cantidad de datos relacionados por nombre de tabla nomencladora.
    - Tablas nomencladoras con mayor cantidad de nomencladores sin utilizar.
    - Generar reportes de vigencia.
6. Registrar Tipo de Nomenclador
    - Adicionar Tipo de Nomenclador.
    - Modificar Tipo de Nomenclador.
    - Eliminar Tipo de Nomenclador.
    - Buscar Tipo de Nomenclador.

A continuación se muestra la descripción de requisitos correspondiente a la agrupación de requisitos Gestionar Nomencladores. Para ver el resto de las descripciones consultar Anexo 2(ver tablas 20 - 35).

2.5.2.1. Descripción del Requisito Gestionar Nomencladores.

Tabla 2. Descripción textual del requisito Adicionar nomenclador.

Precondiciones	1. El usuario debe tener los permisos necesarios y estar autenticado en el sistema. 2. El usuario debe seleccionar del menú la opción “Gestionar Nomencladores”.
Flujo de eventos	
Flujo básico Adicionar nomenclador	
1.	El sistema muestra una interfaz con dos tablas, la tabla 1 con: nombre de los tipos de nomencladores y la tabla 2 vacía.
2.	El usuario selecciona el tipo de nomenclador al que se le desea adicionar un nomenclador.
3.	El sistema activa el botón adicionar. El sistema configura la tabla 2 con los campos: Código, Descripción, Fecha

	creación, Fecha expira y demás campos específicos del tipo de nomenclador y carga los nomencladores registrados correspondientes al tipo de nomenclador seleccionado a través de una petición Ajax .(ver prototipo adicionar nomenclador)
4.	El usuario selecciona la opción Adicionar.
5.	El sistema muestra una formulario con los campos: Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador.
6.	El usuario introduce los datos Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador.
7.	El usuario selecciona la opción “Aceptar”. En caso de que todos los campos requeridos no estén llenos ver flujo alterno *7.a Datos incompletos*. En caso de que exista un nomenclador con el mismo código o descripción ver flujo alterno *7.b Nomenclador repetido.* En caso de seleccionar cancelar ver flujo alterno *7.c Cancelar*.
8.	El sistema registra el nuevo nomenclador.
9.	El sistema muestra un mensaje notificando que el nomenclador se ha registrado correctamente.
10.	Concluye el requisito.
Pos-condiciones	
1.	Se actualiza la tabla 2 incluyendo el nuevo nomenclador adicionado.
Flujos alternativos	
Flujo alternativo 7.a Datos incompletos.	
1.	El sistema muestra un mensaje de error informando que es obligatorio llenar los campos marcados como requeridos.
2.	Vuelve al paso 6 del flujo básico.
Pos-condiciones	
1	Se muestra un mensaje de error informando que es obligatorio llenar los campos marcados como requeridos.
Flujo alternativo 7.b Nomenclador repetido.	
1	Se muestra un mensaje de error informando que ya existe un nomenclador con ese nombre.
2	Vuelve al paso 6 del flujo básico.
Pos-condiciones	
1	Se muestra un mensaje de error informando que ya existe un nomenclador con ese código o descripción.
Flujo Alterno 7.c Cancelar	
1	Concluye la ejecución del requisito.
Pos-condiciones	
1	Concluye el requisito.
Validaciones	

1	1. Todos los campos son obligatorios. 2. El campo código debe ser una cadena de caracteres numéricos que comienza con “000” 3. El campo descripción no debe tener caracteres extraños ni números. 4. El campo Fecha creación debe tener una fecha mayor o igual a la fecha actual. 5. El campo Fecha expira debe tener una fecha mayor que Fecha creación.
Conceptos	**Registro_Nomenclador** Visibles en la interfaz: Nombre Utilizados internamente: Namespace, Uri.
Requisitos especiales	N/A
Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario

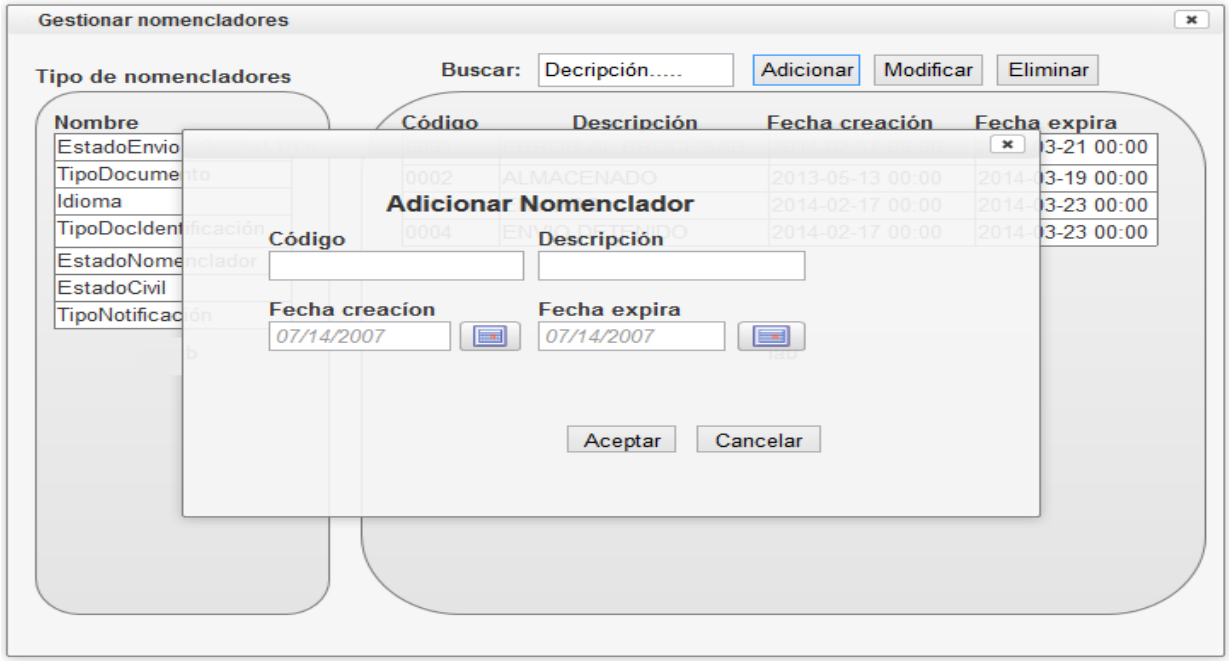


Fig. 3. Prototipo elemental de interfaz gráfica de usuario.

Tabla 3. Descripción textual del requisito Modificar nomenclador.

Precondiciones	1. El usuario debe tener los permisos necesarios y estar autenticado en el sistema.

	2. El usuario debe seleccionar del menú la opción “Gestionar Nomencladores”.
Flujo de eventos	
Flujo básico Modificar nomenclador	
1.	El sistema muestra la interfaz con dos tablas, la tabla 1 con: nombre de los tipos de nomencladores y la tabla 2 vacía.
2.	El usuario selecciona el tipo de nomenclador al que se le desea modificar un nomenclador.
3.	El sistema activa el botón Adicionar. El sistema configura la tabla 2 con los con los campos: Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador y carga los nomencladores registrados correspondientes al tipo de nomenclador seleccionado a través de una petición Ajax .(ver prototipo modificar nomenclador)
4.	El usuario selecciona el nomenclador de la tabla 2 a ser modificado.
5.	El sistema activa el botón Modificar y el botón Eliminar.
6.	El usuario selecciona la opción Modificar.
7.	El sistema muestra una formulario con los campos: Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador.
8.	El usuario modifica alguno(s) de los campos: Código, Descripción, Fecha creación, Fecha expira o de los demás campos específicos del tipo de nomenclador.
9.	El usuario selecciona la opción “Aceptar”. En caso de que todos los campos requeridos no estén llenos ver flujo alterno *9.a Datos incompletos*. En caso de que exista un nomenclador con el mismo código o descripción ver flujo alterno *9.b Nomenclador repetido.* En caso de seleccionar cancelar ver flujo alterno *9.c Cancelar*.
10.	El sistema modifica el nomenclador seleccionado.
11.	El sistema muestra un mensaje notificando que el nomenclador se ha modificado

	correctamente.
12.	Concluye el requisito.
Pos-condiciones	
1.	Se actualiza la tabla 2 incluyendo el nomenclador modificado con sus cambios.
Flujos alternativos	
Flujo alternativo 9.a Datos incompletos.	
1	El sistema muestra un mensaje de error informando que es obligatorio llenar los campos marcados como requeridos.
2	Vuelve al paso 8 del flujo básico.
Pos-condiciones	
1	Se muestra un mensaje de error informando que es obligatorio llenar los campos marcados como requeridos.
Flujo alternativo 9.b Nomenclador repetido.	
1	Se muestra un mensaje de error informando que ya existe un nomenclador con ese nombre.
2	Vuelve al paso 3 del flujo básico.
Pos-condiciones	
1	Se muestra un mensaje de error informando que ya existe un nomenclador con ese código o descripción.
Flujo Alterno 9.c Cancelar	
1	Concluye la ejecución del requisito.
Pos-condiciones	
1	Concluye el requisito.
Validaciones	
1	• Todos los campos son obligatorios. • El campo código debe ser una cadena de cuatro caracteres numéricos que comienza con “000” • El campo descripción no debe tener caracteres extraños ni números. • El campo Fecha creación debe tener una fecha mayor o igual a la fecha actual. • El campo Fecha expira debe tener una fecha mayor que Fecha creación.

Conceptos	**Registro_Nomenclador**	Visibles en la interfaz: Nombre Utilizados internamente: Namespace, Uri.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	
Prototipo elemental de interfaz gráfica de usuario		

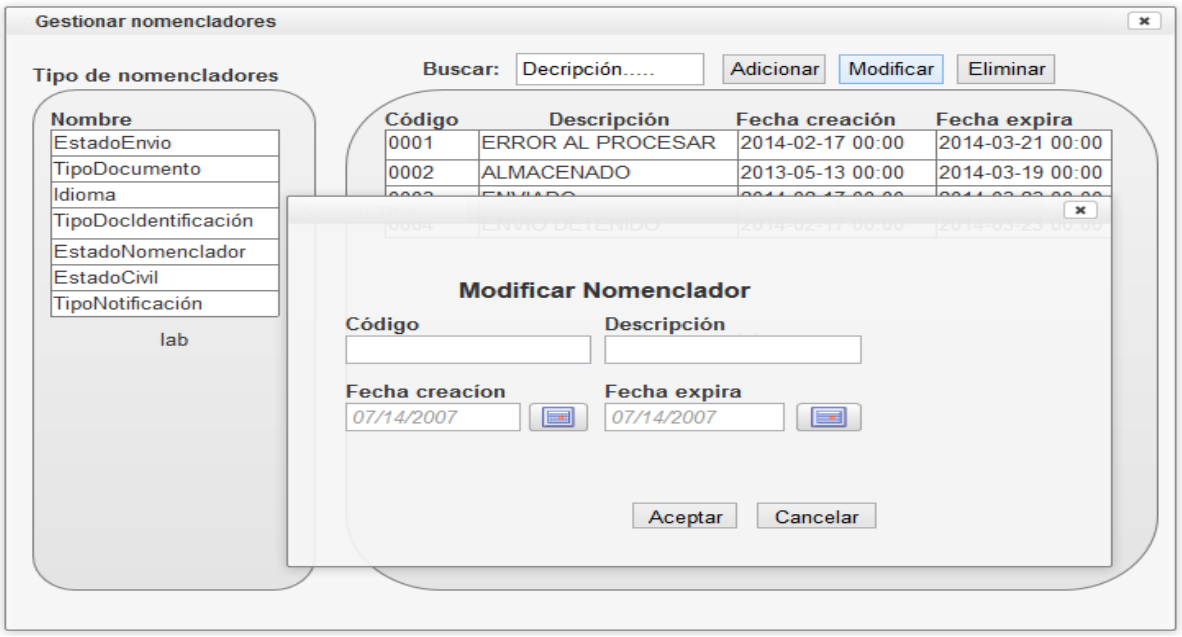


Fig. 4. Prototipo elemental de interfaz gráfica de usuario.

Tabla 4. Descripción textual del requisito Eliminar nomenclador.

Precondiciones	1. El usuario debe tener los permisos necesarios y estar autenticado en el sistema. 2. El usuario debe seleccionar del menú la opción “Gestionar Nomencladores”.
Flujo de eventos	
Flujo básico Eliminar nomenclador	

1	El sistema muestra la interfaz con dos tablas, la tabla 1 con: nombre de los tipos de nomencladores y la tabla 2 vacía.
2	El usuario selecciona el tipo de nomenclador al que se le desea modificar un nomenclador.
3	El sistema activa el botón Adicionar. El sistema configura la tabla 2 con los campos: Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador y carga los nomencladores registrados correspondientes al tipo de nomenclador seleccionado a través de una petición Ajax. (ver prototipo modificar nomenclador)
4	El usuario selecciona el o los nomencladores de la tabla 2 a ser eliminados.
5	El usuario selecciona la opción Eliminar.
6	El sistema muestra una ventana de confirmación de la eliminación. En caso de seleccionar Cancelar ver flujo alterno *6.c Cancelar*.
7	El sistema elimina todos los nomencladores seleccionados.
8	El sistema muestra un mensaje por cada nomenclador eliminado notificando que se ha eliminado correctamente.
9	Concluye el requisito.
Pos-condiciones	
1	Se actualiza la tabla 2.
Flujos alternativos	
Flujo Alterno 6.c Cancelar	
1	Concluye la ejecución del requisito.
Pos-condiciones	
1	Concluye el requisito.
Validaciones	
N/A	
Conceptos	**Registro_Nomenclador** Visibles en la interfaz: Nombre Utilizados internamente:

		Namespace, Uri.
	Nomenclador	Visibles en la interfaz: Código, Descripción, Fecha creación, Fecha eliminación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario

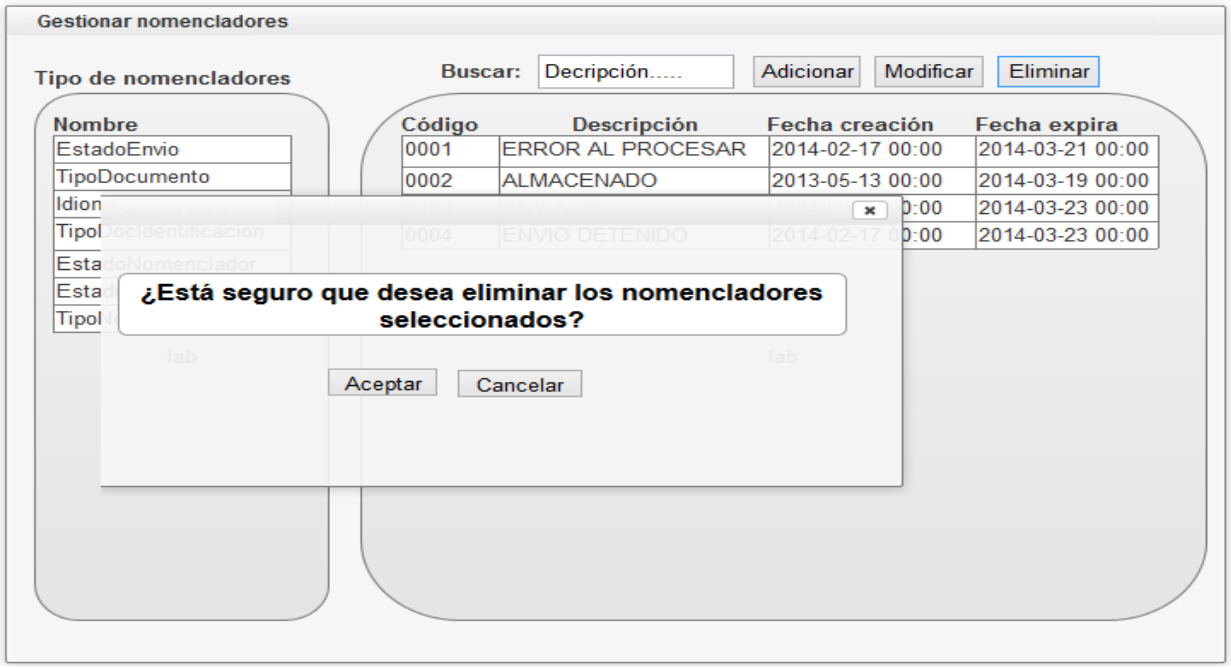


Fig. 5. Prototipo elemental de interfaz gráfica de usuario.

Tabla 5. Descripción textual del requisito Buscar nomenclador.

Precondiciones	1. El usuario debe tener los permisos necesarios y estar autenticado en el sistema. 2. El usuario debe seleccionar del menú la opción “Gestionar Nomencladores”.

Flujo de eventos		
Flujo básico Buscar nomenclador		
1	El sistema muestra la interfaz con dos tablas, la tabla 1 con: nombre de los tipos de nomencladores y la tabla 2 vacía.	
2	El usuario selecciona el tipo de nomenclador de la tabla 1.	
3	El sistema activa el botón Adicionar. El sistema configura la tabla 2 con los campos: Código, Descripción, Fecha creación, Fecha expira y demás campos específicos del tipo de nomenclador y carga los nomencladores registrados correspondientes al tipo de nomenclador seleccionado a través de una petición Ajax. (ver prototipo buscar nomenclador)	
4	El usuario introduce la descripción del nomenclador en el campo de texto de búsqueda.	
5	El sistema filtra por la descripción a través de una petición Ajax y muestra nomencladores coincidentes	
6	Concluye el requisito.	
Pos-condiciones		
1	Se actualiza la tabla 2.	
Flujos alternativos		
Validaciones		
N/A		
Conceptos	**Registro_Nomenclador**	Visibles en la interfaz: Nombre Utilizados internamente: Namespace, Uri.
	Nomenclador	Visibles en la interfaz: Código, Descripción, Fecha creación, Fecha eliminación Utilizados internamente: N/A
Requisitos especiales	N/A	

Asuntos pendientes	N/A

Prototipo elemental de interfaz gráfica de usuario

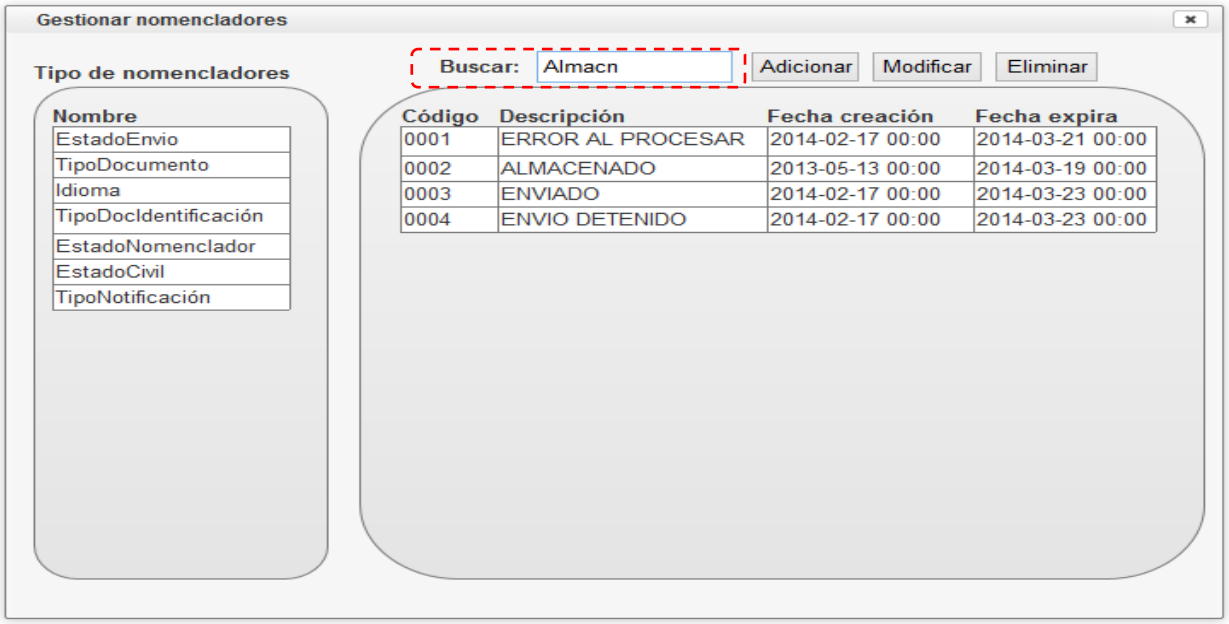


Fig. 6. Prototipo elemental de interfaz gráfica de usuario.

2.5.3. Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares a utilizar, a menudo se aplican al sistema en su totalidad. Como su nombre sugiere, son aquellos requerimientos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. (18) A continuación se describen los requisitos no funcionales que fueron detectados.

1. **Funcionalidad**

   1.1. Precisión

   - Las respuestas del sistema ante búsquedas corresponderán en un 100% a lo solicitado en los criterios.
   - Los resultados de operaciones de negocio en el sistema deben ser 100% correctos.

   1.2. Interoperabilidad

- El sistema permite que otros sistemas pertenecientes a otras instituciones u organismos se comuniquen y envíen datos electrónicamente mediante servicios web.

1.3. Seguridad

- El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.

2. Confiabilidad

2.1. Madurez

- El sistema no permitirá la entrada de datos incorrectos.
- El sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.
- Ninguna información que se haya ingresado en el sistema y haya sido asociada a alguna operación será eliminada físicamente de la Base de Datos.
- El sistema contendrá un mecanismo de alertas y avisos sobre cambios de estados y/o realización de operaciones.

2.2. Tolerancia ante fallos

- El sistema permite detectar fallos internos y notificar al administrador de la ocurrencia de estos.

3. Usabilidad

3.1. Comprensibilidad

- Todos los mensajes de error del sistema deberán incluir una descripción textual del error.
- El orden de las etiquetas de funcionalidades en el menú responderá a las dependencias de ejecución del negocio.
- Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.

3.2. Cognoscibilidad

- La iconografía utilizada será única en cada caso, permitiendo representar todos los conceptos del dominio de la aplicación con un ícono distintivo.
- Los errores cometidos por los usuarios les serán notificados.
- El sistema mostrará las opciones desactivadas siempre que no se hayan cumplido las condiciones previas para su activación.

3.3. Operabilidad

- El sistema contará con un menú que permitirá acceder a todas las funcionalidades para entrar los datos y procesar la información.
- El sistema expondrá el menú general en todo momento para que pueda ser utilizado por el usuario en cualquier momento.
- La confirmación de la entrada de datos deberá poder hacerse mediante el uso de mouse o teclado.

3.4. Atracción

- El sistema diferenciará los mensajes de información de los mensajes de error y de advertencia valiéndose de distintos íconos y colores para cada tipo.
- El sistema presentará los términos capitalizados, es decir, la primera palabra tendrá su primera letra en mayúsculas.
- La tipografía y colores serán estándares en toda la aplicación.

4. Eficiencia

Rendimiento

- El sistema debe responder al cliente de la realización del procedimiento requerido en un lapso de tiempo menor de tres segundos.

5. Mantenibilidad

Diagnosticabilidad

- El sistema debe poseer un mecanismo de almacenamiento, detección y tratamiento de errores.
- El sistema desagregará las funcionalidades por niveles de reutilización, modificación y/o mantenimiento.

6. Soporte

Software

Para el cliente:

- Navegador Mozilla Firefox 7.0 o superior o Chrome 1.0 o superior.
- Sistema operativo Windows 98 o superior o Linux.
- Para el servidor:
- Sistema operativo Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.3.6 disponible o superior. Este debe estar configurado con la extensión “oci” incluida.

- Un servidor de Base de Datos Oracle en su versión 11g.

Hardware

Para el servidor:

- Requerimientos mínimos: procesador Core 2 Duo a 2.0GHz de velocidad de procesamiento y
- 4Gb de memoria RAM.
- Al menos 80Gb de espacio libre en disco duro.
- Tarjeta de red.
- Para el cliente:
- Requerimientos mínimos: procesador Pentium IV a 2.0GHz con 512Mb de memoria RAM.
- Tarjeta de red.

2.5.4. Técnicas para la validación de requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto.

El procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE propone para la validación de los requisitos de software utilizar alguna de las técnicas siguientes :Revisiones del Documento de Requerimientos, Construcción de Prototipos y Generación de Casos de Pruebas. (19)

La validación de los requisitos de software identificados se realizó utilizando la combinación de las técnicas siguientes: Revisiones del Documento de Requerimientos y Construcción de prototipos. El documento de Requerimientos de Software fue objeto de revisiones en conjunto por el administrador de Bases de Datos, el jefe de proyecto y el analista principal del proyecto, los cuales realizaron señalamientos y recomendaciones para mejorar la interpretación y el objetivo del mismo. Además se efectuó la construcción de prototipos los cuales fueron aprobados por el personal antes mencionado.

2.6. Disciplina Análisis y Diseño.

Durante esta disciplina se modela el sistema para que soporte todos los requisitos. En ella se genera un importante conjunto de artefactos que constituyen la base para la implementación del sistema. (9)

A continuación se muestran algunos de los artefactos que conforman el Modelo de Diseño [10](para ver el resto de los artefactos consultar Anexo 3), el Modelo de Datos así como algunos de los patrones de diseño aplicados.

2.6.1. Diagrama de clases de diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, contiene información como clases, asociaciones, atributos, métodos y dependencias. A continuación se describe el diagrama de clases del diseño basado en estereotipos web que se realizó durante el proceso de desarrollo perteneciente al requisito funcional Gestionar Nomencladores.

El diagrama de la Fig. 7 contiene dos páginas clientes, la principal y la clase gestionarNomencladores. La primera de estas es la encargada de hacer un link a la página servidor indicándole que debe construir una instancia de la página cliente gestionarNomencladores la cual se encargara de mostrar todos los datos de los nomencladores. Para ello está compuesta por dos tablas, la primera contiene los tipos de nomencladores registrados, una vez que se selecciona uno se hace una petición a la página servidora la cual hace una llamada al método gestionarNomencladoresDadoTipo () ubicado en la clase controladora, finalmente este devuelve todos los nomencladores registrados para ser mostrados en la segunda tabla. Esta página cliente también contiene un formulario en el cual se adicionan y se modifican los datos de un nomenclador, los mismo son enviados a la página servidora, esta hace una llamada al método adicionarNomencladores() o modificarNomencladores() los cuales están ubicados en la clase controladora, estos son los encargados de actualizar la información del nomenclador usando las clases entidades definidas en el paquete DCNomencladoresEntidades (ver paquete en el diagrama de clases).

[10] Modelo de Diseño: Es el conjunto de diagramas que describen el diseño lógico. Comprende los diagramas de clases de software, diagramas de secuencia, diagramas de paquetes, entre otros.(28)

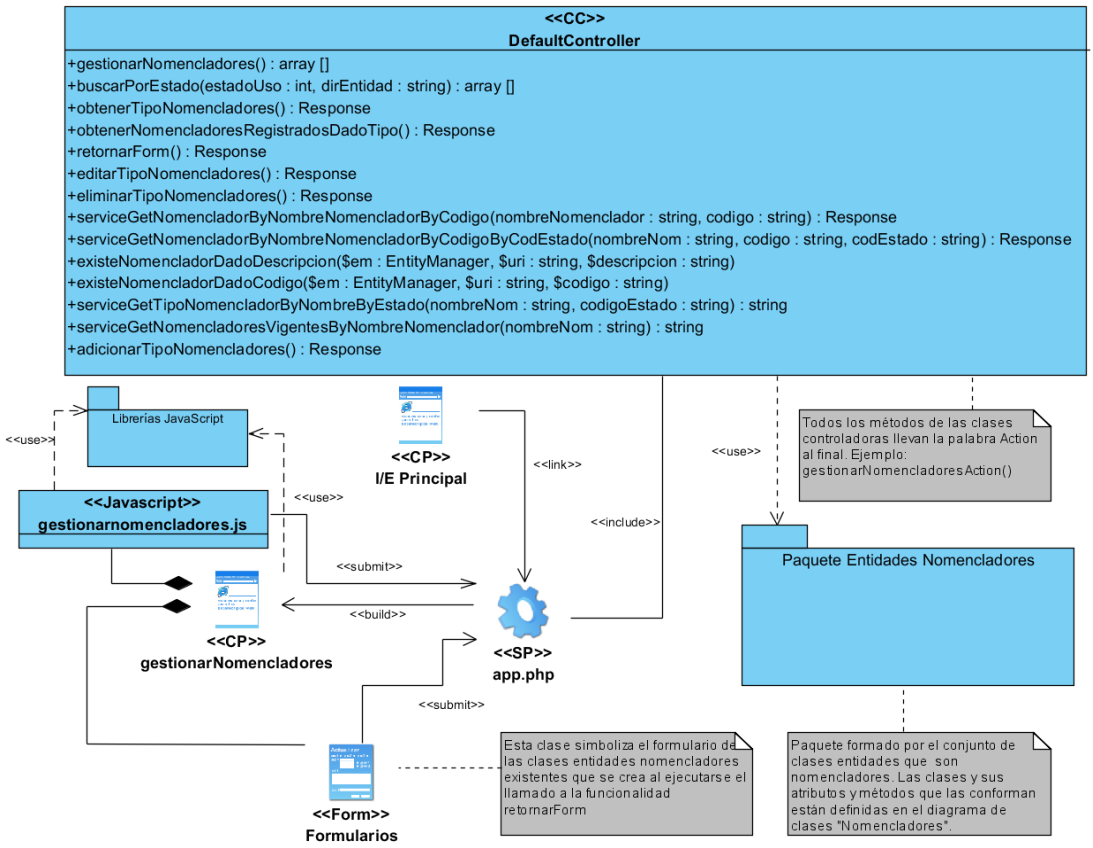


Fig. 7. Diagrama de clases con estereotipos web. Gestionar Nomencladores.

2.6.2. Diagrama de paquetes

Un paquete es un mecanismo utilizado para agrupar y organizar los elementos modelados con UML, facilitando de esta forma el manejo de los modelos de un sistema complejo. Permiten dividir un modelo para agrupar y encapsular sus elementos en unidades lógicas individuales, además se pueden utilizar para plantear la arquitectura del sistema a nivel macro. Los paquetes pueden estar anidados unos dentro de otros y puede existir dependencia entre ellos. (22)

El componente está compuesto por los paquetes Bridge, Controller, DCNomencladores Entidades y Resources. El paquete Bridge es el encargado de almacenar los servicios que exporta la VUCEC a los demás componentes externos. El paquete Controller es el encargado de contener la implementación de las funcionalidades del componente Nomencladores. El paquete DCNomencladores Entidades contiene las entidades que se corresponden con los diferentes tipos de nomencladores a gestionar. El paquete

Resources contiene todos los archivos de configuración del componente así como las interfaces y las bibliotecas de JavaScript utilizadas.

A continuación se muestra el diagrama de paquetes del componente Nomencladores:

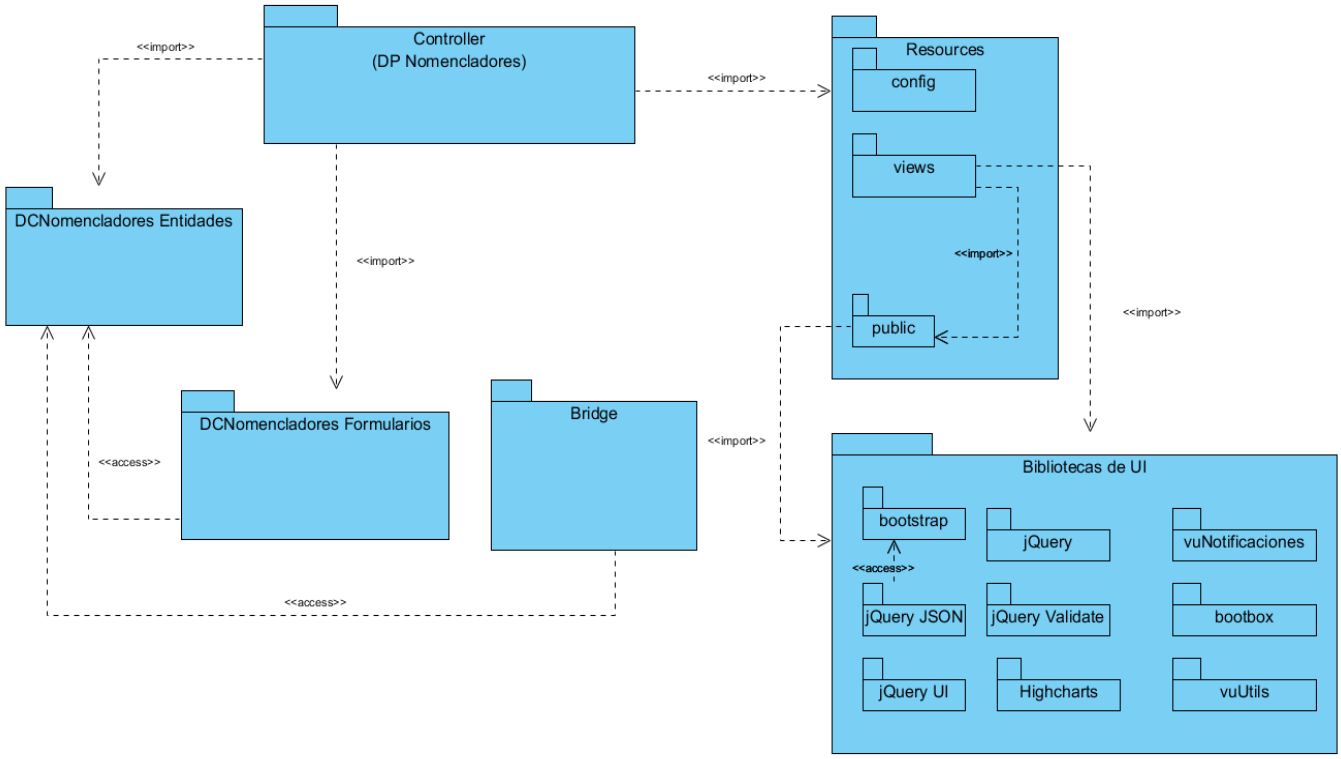


Fig. 8. Diagrama de paquetes.

2.6.3. Patrones Utilizados

Modelo Vista Controlador (Model Views Controller, MVC)

El patrón MVC es un patrón de arquitectura de software encargado de separar la lógica del negocio de la interfaz del usuario y es el más utilizado en aplicaciones Web, pues facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que impide mezclar lenguajes de programación en el mismo código. (20)

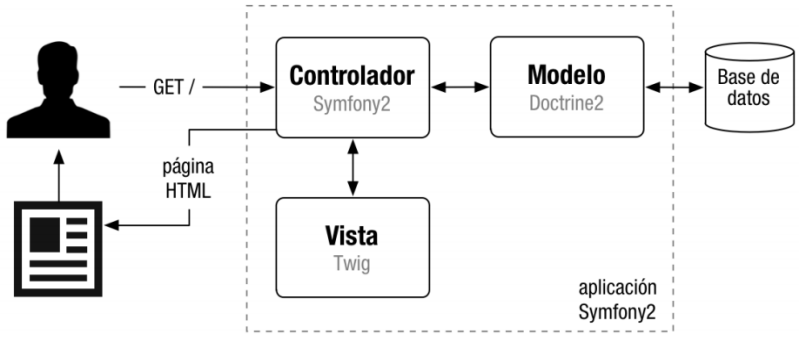


Fig. 9. MVC de una aplicación de Symfony2. (25)

El marco de trabajo Symfony2 está basado en MVC, este toma lo mejor de la arquitectura MVC y lo implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo. El patrón divide las aplicaciones en tres niveles de abstracción. A continuación se detalla la estructura de carpeta que define Symfony2 para su implementación correspondiente a los niveles:

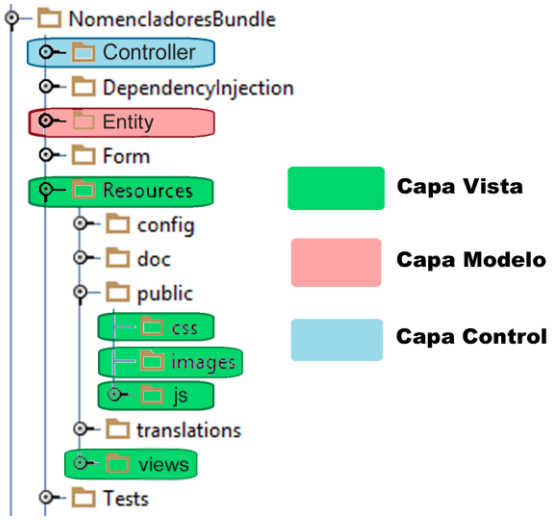


Fig. 10. Estructura de carpetas por niveles de abstracción del componente.

Modelo: en este nivel se encontrarán todas las clases denominadas entidades por Symfony las cuales son generadas por el ORM, estas clases permiten interactuar con la Base de Datos mediante el objeto EntityManager el cual es el verdadero responsable del acceso a los datos.

Vista: En este nivel se encontrarán todas las páginas a visualizar por la persona que interactúe con el sistema. Estas harán uso de la biblioteca JQuery ya mencionada en el capítulo 1 y Bootstrap para mejorar la calidad de las interfaces.

Todas las vistas son desarrolladas con Twig como motor y lenguaje de plantillas. Este permite una sintaxis concisa y limpia, por lo que el código es fácil de leer y de escribir (Sensio Labs, 2010-2012).

Controlador: es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta, lo presente al usuario de una forma entendible.

Patrón Conversor (Mapper) de Base de Datos

Propone crear una clase que sea responsable de materializar y desmaterializar un objeto almacenado (21). En Symfony2 se evidencia en el mecanismo empleado para persistir y recuperar datos de la Base de Datos a través de la correspondencia entre Entidades y la clase EntityManager, esta última es la encargada de establecer la correspondencia entre las Entidades y las tablas físicas de la Base de Datos en base a la lectura de metadatos. De esta forma y conjuntamente con el patrón Active Record (utilizado por Symfony2 para embeber los parámetros de conexión en las clases encargadas del acceso a los datos) se homogeniza el lenguaje utilizado por el programador para persistir y recuperar datos. En esta investigación se hará uso de este patrón respetando el mecanismo definido por Symfony2 para el acceso a los datos almacenados teniendo en cuenta las facilidades que brinda el mismo y su utilidad para la gestión dinámica de las tablas nomencladoras.

Patrones GRASP y GoF

A parte de los patrones mencionados previamente, se utilizan los de Asignación General de Responsabilidad (General Responsability Assignment Software Patterns ,GRASP) los cuales describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y los GoF [11] para controlar la creación (preocupación con el proceso de creación de objetos y clases) de estructuras (composición de las clases y objetos) y sus comportamientos (caracterizan la interacción y la responsabilidad de los objetos y clases) en el software. Estos patrones son de gran importancia porque influyen en la solidez, la capacidad de mantenimiento, la reutilización de código y la obtención de los objetos o las abstracciones adecuadas (Larman, 1999).

Alta Cohesión

La cohesión funcional es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento. Un elemento con responsabilidades altamente relacionadas, y que

[11] Patrones que fueron creados por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides en 1995.

no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas, etcétera.

En el componente Nomencladores su uso se materializa en el hecho de que se crea una clase Entidad para cada nuevo tipo de nomenclador a gestionar la cual almacena información solo de los atributos que la definen y realiza las operaciones necesarias para estos.

Controlador

Este patrón GRASP define que deben existir clases, que se encarguen de manejar los eventos que genera el usuario al interactuar con el sistema, estas clases son las responsables de contener o delegar en otros controladores las funcionalidades que generan la respuesta al usuario, evitando que la lógica del sistema no se maneje en la capa de interfaz.(28) En esta investigación se agrupan las funcionalidades referentes a la gestión de los nomencladores en la clase controladora DefaultController.php y la clase TipoNomencladorController.php es responsable de contener las funcionalidades referentes al registro de información de los tipos de nomencladores existentes. De esta forma cada clase controladora es responsable de brindar las funcionalidades que le corresponden o delegar las mismas en las clases expertas.

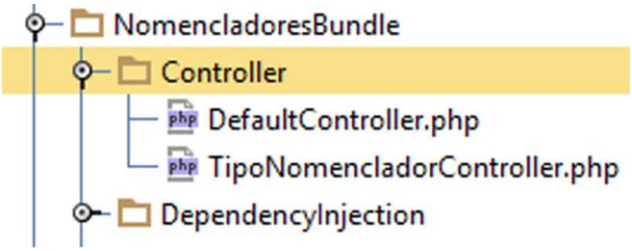


Fig. 11. Controladores de gestión.

Patrón Inyección de dependencias

La inyección de dependencias es un patrón de diseño, consiste en que los objetos que necesita una clase para funcionar no son creados dentro de la propia clase, sino que son suministrados desde afuera.(21) Este patrón se evidencia en la clase NomencladoresBridge, la cual al heredar de la clase Bridge esta le inyecta el contenedor.

2.6.4. Modelo de datos

El modelo de dato se compone de tres piezas fundamentales: el objeto de datos, los atributos que describen el objeto de datos y la relación que conecta los objetos de datos entre sí. (Pressman, 2005)

El modelo de datos de la solución cuenta con diez clases persistentes actualmente. De estas nueve constituyen nomencladores y la otra es la tabla de registro de los tipos de nomencladores a gestionar.

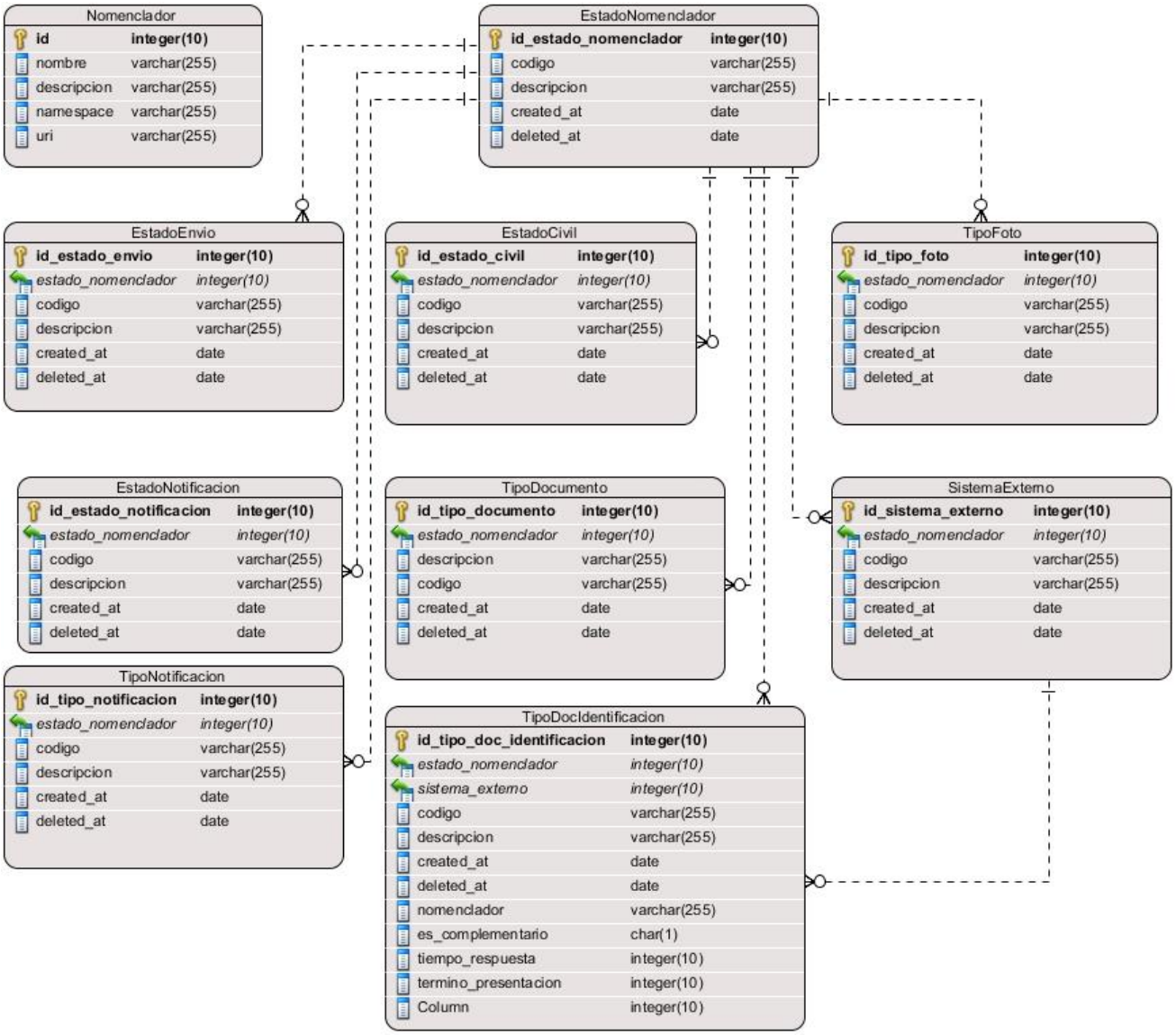


Fig. 12. Diagrama Entidad-Relación con las entidades y sus relaciones del componente.

2.7. Conclusiones del capítulo

El este capítulo se detallaron las características fundamentales de la propuesta de solución para comprender mejor las acciones a desarrollar por el componente de gestión de nomencladores.

Con el análisis y diseño de la solución se obtuvieron veinte requisitos funcionales para el componente modelado y se generaron los artefactos definidos para estas fases por el Modelo de desarrollo de CEIGE versión 1.2.

Los artefactos generados, unido al buen uso de los patrones de diseño permitieron sentar las bases para la implementación del software con un alto grado de calidad.

3. CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN

3.1. Introducción

En el presente capítulo se exponen y describen los artefactos de implementación de la solución en cuestión. Se especifican algunos de los estándares de codificación por los que se rigen los programadores en el Departamento de Soluciones para la Aduana así como la descripción del tratamiento de errores realizado. Por último se aplican pruebas al sistema para verificar que el comportamiento del software cumple con la especificación de los requisitos identificados y se valida el diseño y la implementación realizada.

3.2. Disciplina Implementación.

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes de software, es decir, ficheros de código fuente, scripts, ejecutables y similares. (9) Durante esta disciplina también se genera el Diagrama de componentes.

3.2.1. Diagrama de componentes

Un componente representa una parte de un sistema modular, desplegable, y reemplazable, que encapsula la implementación de un conjunto de funcionalidades relacionadas entre sí y que expone un conjunto de interfaces.

Los diagramas de componentes permiten modelar la estructura de un software y las relaciones de dependencia que se establecen entre los componentes que lo integran a partir de la comunicación mediante las interfaces. Estas interfaces indican que un componente utiliza los servicios ofrecidos por otro.

A continuación se muestra el diagrama de componentes de la presente investigación en el cual se evidencian las relaciones de dependencia que se establecen entre el componente Nomencladores y el resto de los componentes de la VUCEC.

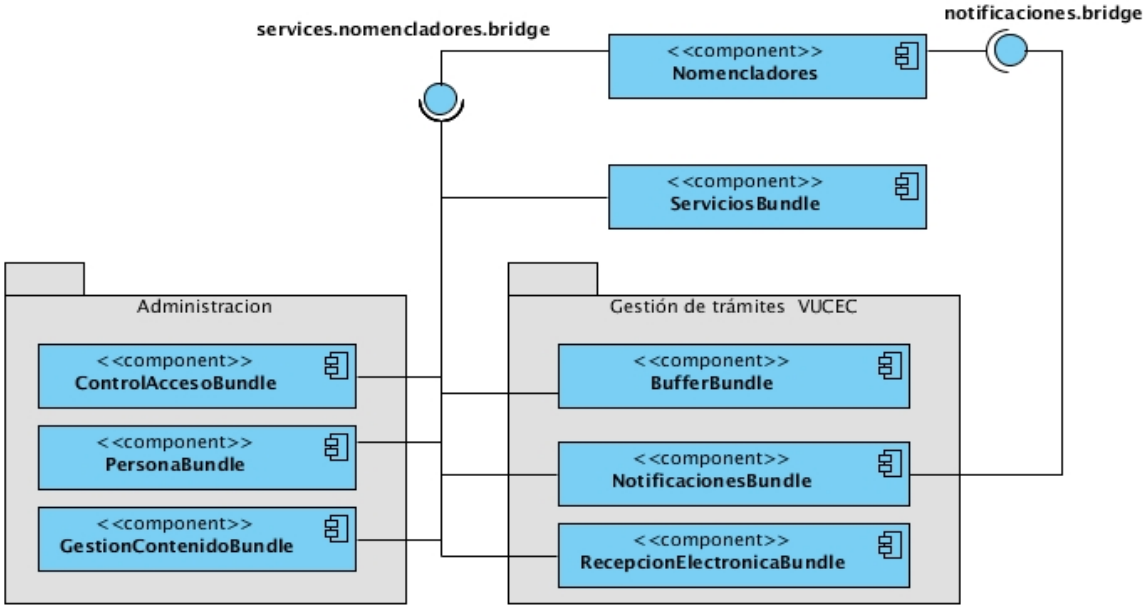


Fig. 13. Diagrama de componentes.

3.2.2. Estándar de codificación

Los estándares de codificación surgen como una necesidad de mantener una mejor comunicación entre los programadores de manera que se favorezca la reutilización y mantenimiento de los sistemas. Permiten identificar de forma sencilla el objetivo y las funcionalidades que brinda cada una de las clases. En el mundo existen diferentes estándares para cada uno de los lenguajes existentes. (23)

En la implementación de la solución se siguió el estándar de codificación del Departamento de Soluciones para la Aduana del CEIGE. Entre los estándares establecidos y debido a la cantidad de veces que se utilizan en la implementación de la solución se destacan los siguientes:

- Los nombres de las clases deben estar expresados en notación UpperCamelCase.[12]
- No se deben utilizar guiones bajos en su nombre "_".
- El nombre de las clases debe expresar con claridad el alcance y la responsabilidad de la misma.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma.

[12] **UpperCamelCase**: Consiste en escribir frases o palabras compuestas eliminando los espacios intermedios y poniendo en mayúscula la primera letra de cada palabra incluyendo la primera letra de la frase.

- Los formularios tendrán el sufijo Type para identificarlos según el estándar establecido por el marco de trabajo Symfony2 para este tipo de clases.

Ejemplo:
NomencladoresType, EstadoNomencladorType, TipoDocumentoType, etc.

- Todas las funciones definidas por los desarrolladores deben seguir la nomenclatura UpperCamelCase, a no ser que para cierto ámbito se especifiquen características específicas.
- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza la misma.

Ejemplo:
buscarPorEstadoAction, obtenerTipoNomencladoresAction, etc.

- Cualquier operación relacionada con una tabla de la Base de Datos debe situarse en el modelo de la clase correspondiente a dicha tabla.
- Los nombres de las variables deben expresar claramente el contenido de la misma.
- En caso de que no se le asigne un valor inicial a las variables se deben inicializar con un valor que indique el tipo de dato al que debe pertenecer.
    - Los tipos de datos cadena son definidos con comillas dobles (“).
    - Los tipos de datos de caracteres se definen con comillas simples („).
    - En caso de que se espere almacenar tipos de datos diversos no se inicializa.

Ejemplo:
```
$index = 0;
$search == "";
$arreglo_temp = array();
$objeto = new Clase;
$mixta
```

- Para la documentación de las variables y las funciones se utilizará un comentario de bloque donde se especifique el objetivo de las mismas, el o los tipos de parámetros y el tipo de retorno si es necesario para la función.

Ejemplo de la documentación de una función:
```
/**
* Comentario u objetivo de la función
```

```
* @param $estadoUso
* @param $entidad
* @return void
*/
public function buscarPorEstadoAction($estadoUso, $entidad){
// TODO
$result = array (); // comentario u objetivo de la función
}
```

3.2.3. Tratamiento de errores

En el desarrollo de software el tratamiento de errores es importante para garantizar el correcto funcionamiento del sistema y que no se afecte la calidad y disponibilidad del mismo. Es fundamental identificar y controlar los problemas que puedan presentarse a la hora de interactuar con el software. En el desarrollo de la solución se pueden apreciar diferentes mecanismos para el tratamiento de errores, los cuales son mencionados a continuación:

Se utiliza JavaScript para depurar los errores de parte del cliente validando los formularios y evitando consultas a la Base de Datos sin sentido o que tengan malas intenciones como son las inyecciones SQL, en cualquiera de estas situaciones el JavaScript funciona como una barrera y no deja pasar estos valores lanzando alertas o excepciones.

Las validaciones del negocio se encargan de controlar el flujo de los datos recibidos en el controlador para evitar inconsistencias en los datos de entrada. Se llevan a cabo en las diferentes clases con el uso de funciones propias del lenguaje. En caso de que exista algún error, es notificado al usuario para que pueda corregirlos.

Otro de los aspectos importantes utilizado en el tratamiento de errores del sistema desarrollado es la utilización de los formularios generados por Symfony2 para insertar, eliminar o modificar valores de la Base de Datos, los validadores son configurados en cada uno de los campos de las clases entidades. La biblioteca de formularios de Symfony2 utiliza internamente el servicio ***validator*** para validar los objetos después de que los valores se han presentado y vinculado.

Ejemplo:

```
$form = $this->createForm(new $direccFormType, $tiponomencladorTmp);
$form->bindRequest($this->getRequest());
```

```
if ($form->isValid()) {……..
```

3.3. Métricas para validar el diseño

Las métricas de software son una medida cuantitativa que permite a los desarrolladores tener una visión profunda de la calidad del proceso de desarrollo del software. (Pressman, 2005). Para validar el diseño de esta investigación se aplicaron las métricas Tamaño operacional de la clase y Relaciones entre clases.

3.3.1. Tamaño operacional de la clase (TOC)

El tamaño operacional de las clases está dado por el número de métodos asignados a una clase. Mediante este número de métodos se calcula el nivel de responsabilidad, la complejidad de implementación y su reutilización a fin de inspeccionar la efectividad del diseño, utilizando estos criterios como un conjunto de atributos que se ven afectados para lograr de forma estadística el tamaño de las operaciones realizables:

Tabla 6. Criterios a medir.

Tamaño operacional de clase (TOC)	
Atributos que afecta:	**Modo en que lo afecta:**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

La aplicación de la métrica parte de identificar el número de procedimientos heredados y privados y calcular el promedio de procedimientos:

Total de clases: **13**

Promedio de procedimientos: **16,23**

A los valores obtenidos anteriormente se le aplican los siguientes criterios para obtener en un rango (alta media o baja) todas las clases clasificadas:

Tabla 7. Rango de valores para la evaluación de los atributos relacionados con la métrica TOC.

	Categoría	Criterios
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad de implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Tabla 8. Resultados.

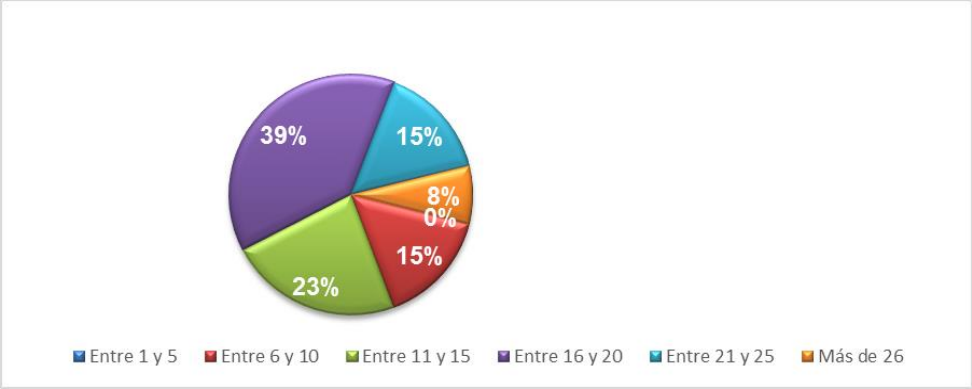
Nombre clase	Cantidad de procedimientos	Responsabilidad	Complejidad	Reutilización
EstadoNomenclador	25	Media	Media	Media
Nomenclador	16	Baja	Baja	Alta
SistemaExterno	24	Media	Media	Media
TipoDocumento	28	Media	Media	Media
TipoDocIdentificacion	14	Baja	Baja	Alta
TipoFoto	16	Baja	Baja	Alta
EstadoCivil	14	Baja	Baja	Alta
EstadoEnvio	16	Baja	Baja	Alta
TipoNotificacion	17	Media	Media	Media
EstadoNotificacion	16	Baja	Baja	Alta
NomencladoresBridge	6	Baja	Baja	Alta
DefaultController	12	Baja	Baja	Alta
TipoNomencladorController	7	Baja	Baja	Alta

Estadísticamente se pueden analizar los promedios de los procedimientos por criterio de cantidad de procedimientos de la siguiente manera:

Tabla 9. Promedios por rango de cantidades de procedimientos.

Criterio	Cantidad de clases	Promedio
Entre 1 y 5	0	0
Entre 6 y 10	2	15,38461538
Entre 11 y 15	3	23,07692308
Entre 16 y 20	5	38,46153846
Entre 21 y 25	2	15,38461538
Más de 26	1	7,692307692
Total	13	100

A continuación se muestra un análisis gráfico de los porcientos de procedimientos por intervalos y otro para cada uno de los atributos o criterios medidos para una mejor apreciación del análisis estadístico realizado:



Gráfica 1. Representación en porciento de los resultados obtenidos agrupados por intervalos.

Tabla 10. Responsabilidad.

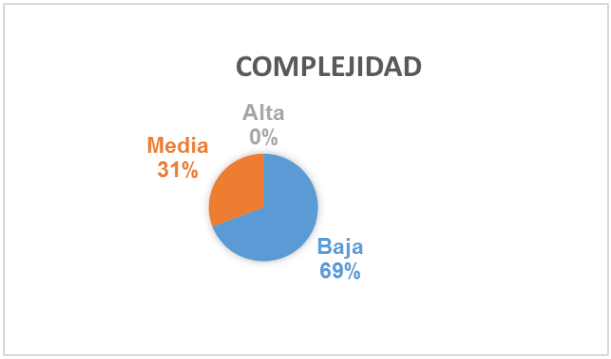
Responsabilidad	Cantidad de clases	Promedio
Baja	9	69,23076923
Media	4	30,76923077

Alta	0	0

Gráfica 2. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad**.**

Tabla 11. Complejidad.

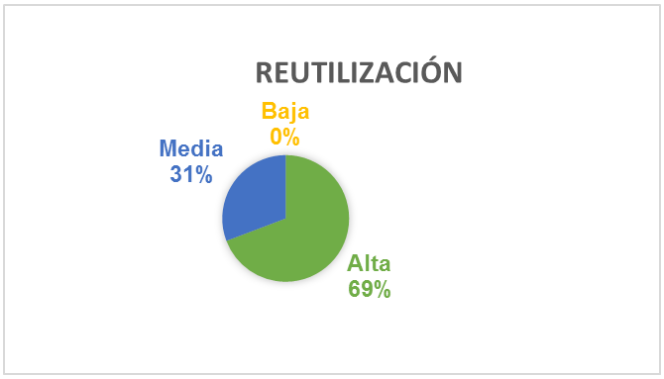
Complejidad	Cantidad de clases	Promedio
Baja	9	69,23076923
Media	4	30,76923077
Alta	0	0



Gráfica 3. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad.

Tabla 12. Reutilización.

Reutilización	Cantidad de clases	Promedio
Alta	9	69,23076923
Media	4	30,76923077
Baja	0	0



Gráfica 4. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Cuando existe un TOC alto se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

Todas las clases que conforman el sistema están dentro de las categorías de baja y media para los atributos Responsabilidad y Complejidad de implementación, lo que demuestra que el diseño propuesto para estos aspectos es positivo. Con relación al atributo Reutilización se obtuvo que el 69 % de las clases presenta un alto grado de reutilización y el 0 % de las clases presenta reutilización baja por lo que se concluye que los resultados obtenidos de manera general según esta métrica son positivos.

3.3.2. Métrica Relaciones entre Clases (RC)

Para el funcionamiento del sistema las clases se relacionan con otras con el objetivo de manipular información que necesitan para su correcto funcionamiento. La métrica Relaciones entre Clases está basada en el número de relaciones de uso de una clase con otra. Esta se aplica con el objetivo de medir atributos de calidad como: reutilización, acoplamiento, complejidad de mantenimiento y cantidad de pruebas.

Para aplicar esta métrica se hizo un conteo de las relaciones de uso que poseen cada una las clases del componente. Luego se pasó a calcular el promedio de las mismas y con ambos valores según los criterios expuestos en la tabla siguiente (Tabla 13) se determinaron los valores para cada atributo medido.

Tabla 13. Rango de valores para medir la afectación de los atributos de calidad (RC).

Atributos de calidad	Clasificación	Criterio

Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

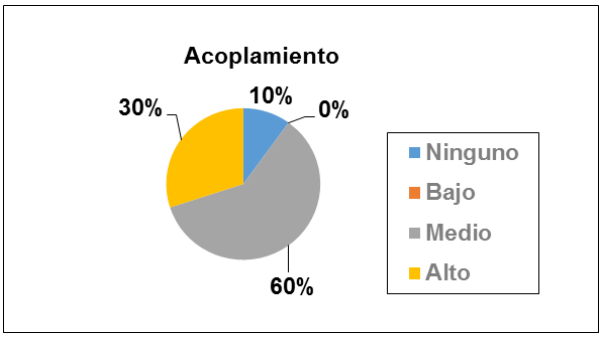
La siguiente tabla muestra los valores obtenidos a partir de la aplicación de la métrica RC a un conjunto de tablas del componente Nomencladores. Se identificó un total de veintiséis relaciones de uso para un promedio de 2.6 por cada clase.

Tabla 14. Valores obtenidos para la métrica RC.

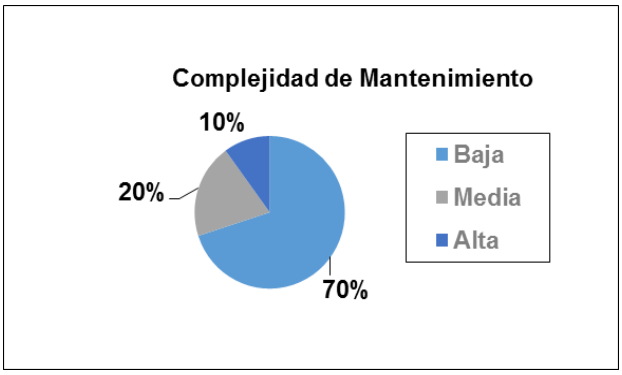
Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mantenimiento.	Reutilización
EstadoNomenclador	8	Alto	Alto	Baja
Nomenclador	0	Ninguno	Baja	Alta
SistemaExterno	3	Alto	Media	Media
TipoDocumento	3	Alto	Media	Media
TipoDocIdentificacion	2	Medio	Baja	Alta
TipoFoto	2	Medio	Baja	Alta
EstadoCivil	2	Medio	Baja	Alta

EstadoEnvio	2	Medio	Baja	Alta
TipoNotificacion	2	Medio	Baja	Alta
EstadoNotificacion	2	Medio	Baja	Alta

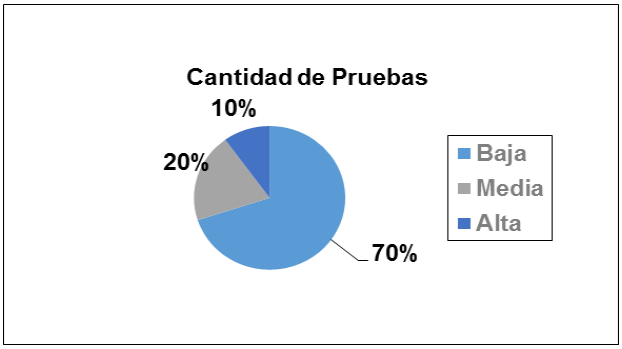
Seguidamente se muestra una gráfica de pastel por cada atributo de calidad según los resultados obtenidos:



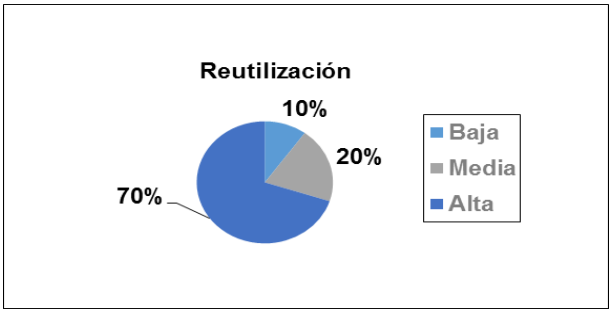
Gráfica 5. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.



Gráfica 6. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Mantenimiento.



Gráfica 7. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.



Gráfica 8. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Como se puede apreciar, los valores obtenidos para los atributos Complejidad de Mantenimiento y Cantidad de pruebas se mantienen bajos en su mayoría por lo que se puede afirmar que el diseño presentado tiene un bajo nivel de complejidad para su mantenimiento y necesita una baja cantidad de pruebas. A su vez el atributo Reutilización concuerda con los resultados obtenidos de la métrica anterior por lo que el sistema tiene un alto grado de reutilización.

Con respecto al atributo Acoplamiento se puede observar según la aplicación de la métrica que el 60 % de las clases presenta un acoplamiento medio y un 30 % del total presenta una acoplamiento alto por lo que los resultados no son los mejores. Las clases que presentan deficiencias fueron analizadas y como conclusión se obtuvo que es imposible minimizar el número de relaciones de uso que establecen estas, debido a que todas las relaciones son necesarias para el correcto funcionamiento del componente.

3.4. Validación de la implementación

La prueba del software es un elemento crítico para la garantía de la calidad del mismo, representa

una revisión final de las especificaciones, el diseño y la codificación, estos elementos juntos componen la aplicación computacional. Con el objetivo de validar el componente y descartar errores en el mismo se realizaron pruebas al software como garantía de su calidad. Con estas pruebas se puede detectar todo posible mal funcionamiento de la aplicación, se considera una prueba exitosa si se demuestran deficiencias en el software.

3.4.1. Pruebas de Caja blanca o estructurales

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que: garanticen que se ejerciten por lo menos una vez todo los caminos independientes de cada método; se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez.(23)

Camino básico.

Para la solución desarrollada la prueba de Caja Blanca aplicada fue la del camino básico. Esta técnica de prueba permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Antes de realizar el método del camino básico se realiza una representación del flujo de control, denominada Grafo de Flujo el cual permite identificar mejor los caminos del programa. Para ello se representa el flujo de control lógico mediante la notación del grafo de flujo mostrada a continuación:

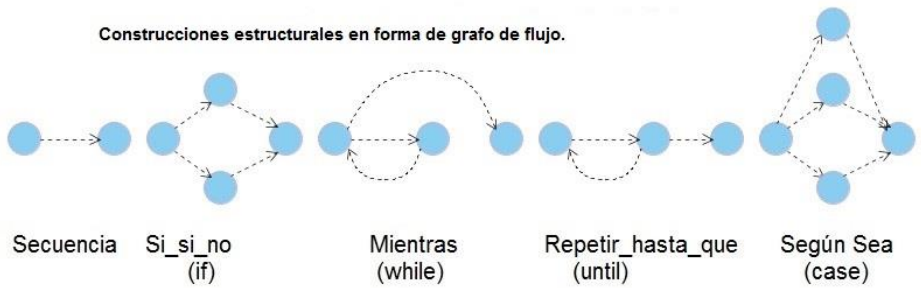


Fig. 14. Construcciones estructurales en forma de grafo de flujo.

Uno de los pasos para la aplicación de esta técnica es el cálculo de la complejidad ciclomática, que es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

A continuación se muestra el código del método eliminarNomencladorAction del requisito Gestionar nomencladores para aplicarle la prueba:

```
public function eliminarNomencladoresAction() {
   $em = $this->getDoctrine()->getManager();
    $uri= $this->getRequest()->get('uriForm');
    $tiponomencladoresEliminar = json_decode($this->getRequest()->get("id_tiponomenclador"), true);
    $errores = array();
    foreach ($tiponomencladoresEliminar as $tiponomenclador) {
        try {
           $em->remove($em->getRepository($uri)->find($tiponomenclador));
           $em->flush();
        }
        catch (\Exception $exc) {
            $errores[] = $exc->getMessage();
            $respuesta = array("success" => false,
            "data" => array("head" => "Se han encontrado errores:","errors" => $errores)); }
    }
    if (count($errores) == 0) {
        if (count($tiponomencladoresEliminar) == 1) {
            $respuesta = array("success" => true,"data" => array(
                    "textResponse" => "TipoNomenclador eliminado satisfactoriamente."));
        }
        else { $respuesta = array("success" => true,"data" => array(
                    "textResponse" => "TipoNomencladores eliminados satisfactoriamente."));
        }
    }
    return new Response(json_encode($respuesta));
}
```

Fig. 15. Código fuente del método eliminarNomencladores de la clase DefaultController.

Para el cálculo de la complejidad ciclomática es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la figura siguiente:

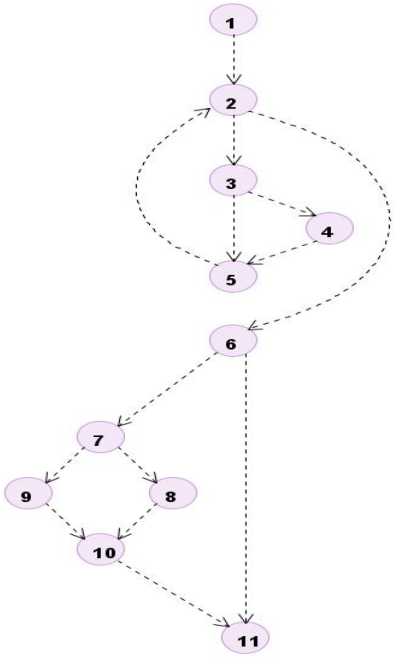


Fig. 16. Grafo de flujo correspondiente a la funcionalidad eliminarNomencladores**.**

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmula 1 .V (G) = (A – N) + 2

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

Resultado.

Siendo: A=14	N=11.

V (G)= (14 -11) + 2 = 5.

Fórmula 2 .V (G) = P + 1.

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

Resultado

Siendo: P= 1.	V (G) =4+1=5.

Fórmula 3 .V (G) = R.

Siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado.

Siendo R=5.	V (G)=5

A partir de los resultados obtenidos del cálculo de la complejidad ciclomática se obtiene que 5 es el límite superior de pruebas que se deben realizar para asegurar que se ejecuta al menos una vez cada sentencia. Los casos de prueba se construyen a partir de los caminos generados a continuación, de forma tal que con los parámetros introducidos se recorran cada uno de los nodos del camino.

Camino 1: 1-2-3-5-2-6-11
Camino 2: 1-2-3-5-2-6-7-8-10-11
Camino 3: 1-2-3-5-2-6-7-8-9-11
Camino 4: 1-2-3-4-5-2-6-11
Camino 5: 1-2-6-11

Pruebas unitarias

La prueba de unidad o pruebas unitarias son la primera fase de las pruebas dinámicas y se realizan sobre cada módulo del software de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional independiente de un programa, está correctamente codificado. (24)

Las pruebas unitarias permiten validar que una porción del código de un sistema funciona apropiadamente. Cuando se programa orientado a objetos la menor unidad es la clase, debido a esto las pruebas unitarias en los sistemas orientados a objetos están enfocadas a probar que cada método de la clase responde de manera correcta dado una entrada determinada.

Para la realización de este tipo de prueba se utilizó la biblioteca PHPUnit, la cual es integrada con el marco de trabajo Symfony2 combinando la potencia de la primera con las facilidades del segundo. (25) Symfony2 establece como estándar que para cada prueba unitaria que se realice, se hace necesario crear una clase PHP en el subdirectorio Tests/ de los paquetes con su nombre terminado en Test.php.

Entre las clases escogidas para aplicar las pruebas se encuentran las clases Entidades que gestiona el componente, comprobando que la validación de los atributos de cada clase funciona correctamente lo cual asegura que ningún nomenclador sea registrado con valores inválidos. También se le aplicaron las pruebas a la clase NomencladoresBridge la cual presenta todos los servicios que brinda el componente Nomencladores para el consumo de los demás componentes de la VUCEC.

A continuación se muestra un fragmento de las pruebas aplicadas a las funcionalidades pertenecientes a la clase NomencladoresBridge y el resultado arrojado:

```php
public function testAdicionarNomenclador($result, $valores)
{
    $res = $this->get('services.nomencladores.bridge')
                ->adicionarNomenclador($valores[0], $valores[1]);
    $this->assertEquals($result, $res);
}
public function addProvider()
{
    $uno = array("EstadoEnvio", array('codigo' => '00043', 'descripcion' => 'pruebauno',
                'createdAt' => '10-10-2014', 'deletedAt' => '10-10-2028'));
    $dos = array("EstadoEnvio", array('codigo' => 'dasd', 'descripcion' => 'pruebados',
                'createdAt' => '10-10-2014', 'deletedAt' => '10-10-2028'));
    $tres = array("EstadoEnvio", array(...));
    $cuatro = array("EstadoEnvio", array(...));
    $cinco = array("EstadoEnvio", array(...));
    return array(
        array(true, $uno)
    , array(array("El campo código debe tener el formato 000 seguido de números"), $dos),
        array("ya existe un nomenclador con ese código o descripción", $tres),
        array("ya existe un nomenclador con ese código o descripción", $cuatro),
        array(...)
    );
}
```

Fig. 17. Código fuente del testAdicionarNomenclador de la clase testNomencladoresBridge.

Seguidamente se muestra una tabla que arroja los resultados obtenidos en la prueba unitaria a la funcionalidad adicionarNomenclador correspondiente al requisito funcional "Actualización de nomencladores con el GINA":

Tabla 15. Prueba unitaria para el servicio AdicionarNomenclador de la clase NomencladoresBridge.

Prueba: Unitaria		
Nombre Prueba: NomencladoresBridge. AdicionarNomenclador		
Estado: Satisfactoria	Tipo: Regresión	Ultima ejecución : 26/05/2014
Ejecutado por: Daniel Arturo Casals Amat	Verificado por: Daniel Arturo Casals Amat	
Descripción: Se comprueba que la funcionalidad funcione correctamente, realizando las validaciones correspondientes, almacenando el nomenclador si el juego de datos es correcto y retornando "true".		
Entrada: parametro1: $result(resultado esperado) parametro2: $valores(conjunto de valores para evaluar el método adicionarNomenclador)		
Criterio de aceptación: Almacena un nomenclador con los datos especificados por parámetro. En caso de existir errores, devuelve un mensaje con el o los errores encontrados.		

Resultado: Almacena el nuevo nomenclador.

Se realizaron dos iteraciones de las pruebas unitarias a las funcionalidades de la clase NomencladoresBrige y las clases entidades corrigiéndose los errores detectados en la primera iteración. En la segunda iteración el 100 % de las funcionalidades arrojaron resultados satisfactorios para esta prueba.

3.4.2. Pruebas de Caja negra

Las pruebas de caja negra, se centran en los requisitos funcionales del software. Permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

La prueba de caja negra intenta encontrar errores de las siguientes categorías: (1) funciones incorrectas o ausentes, (2) errores de interfaz, (3) errores en estructuras de datos o en accesos a Bases de Datos externas, (4) errores de rendimiento y (5) errores de inicialización y de terminación.

El componente fue sometido a dos iteraciones de pruebas de caja negra, para estas se utilizaron los casos de pruebas generados (ver Anexos: Tabla 31 y 32) a partir del resultado de la aplicación de la prueba de camino básico. En la primera iteración se detectaron doce no conformidades:

Funciones incorrectas: 7

Errores de interfaz: 4

Errores de acceso a Base de Datos: 1

Todas estas no conformidades fueron resueltas dando paso a la segunda iteración en la que no se detectaron no conformidades, obteniendo resultados satisfactorios.

3.4.3. Validación del Rendimiento

Uno de los objetivos de la presente investigación es que la solución propuesta contribuya a mejorar el rendimiento del sistema a partir de la optimización del mecanismo empleado para la gestión de los nomencladores. En vista a comprobar el cumplimiento del mismo se realizó una comparación del rendimiento del sistema antes y después de ser usados los servicios que brinda el componente, para esto se utilizó la herramienta Apache JMeter 2.3.

Se implementó un controlador de prueba con los siguientes métodos:

1. testGetTipoDocumentoSinCache: obtiene todos los nomencladores de tipo TipoDocumento a partir del uso de una consulta sin uso de la cache (mecanismo utilizado anteriormente)

2. testGetTipoDocumentoConCache: obtiene todos los nomencladores de tipo TipoDocumento utilizando cache apc.
3. testGetTipoDocumentoSerivicioVigencia: obtiene todos los nomencladores de tipo TipoDocumento utilizando el servicio serviceGetNomencladoresByNombreNomenclador.

Las pruebas realizadas se organizaron de la siguiente forma:

- Se probaron los métodos anteriores realizando veinticinco iteraciones con dos segundos de separación entre cada petición y sin repeticiones para verificar el rendimiento del sistema con una carga normal y trescientas iteraciones con un segundo de separación entre cada petición para verificar el rendimiento en una carga extrema. Esta prueba se realizó con el sistema de cacheo APC desactivado.(ver enla figura 18 la gráfica de la izquierda)
- Se realizó el procedimiento anterior, esta vez con el sistema de cacheo APC activado. (ver Fig.18 la gráfica de la derecha)

Se puede observar en las gráficas generadas que los métodos que consumen los servicios del componente para las consultas (los cuales están configurados para el uso de la caché de APC) tienen mejor rendimiento que el método 1 utilizado actualmente por el sistema, esto se produce una vez que se ha configurado el sistema para el uso de APC caché.

Los tiempos obtenidos en estas pruebas son elevados como se puede observar, los indicadores que afectan este rendimiento están asociados a que las consultas realizadas se ejecutaron para la tabla TipoDocumento, a la misma se le insertaron más de 1000 tuplas para la prueba; las pruebas se realizaron en el entorno de desarrollo del marco de trabajo Symfony2, el mismo es más lento que el entorno de producción que es el utilizado para el despliegue de la aplicación, para las pruebas se utilizó una estación de trabajo con un procesador dual-core a 3.00 GHz con 3 GB de Memoria RAM utilizables para la prueba, el servidor utilizado para el despliegue de la aplicación debe poseer mayores prestaciones.

Por las razones antes expuestas se estima que el rendimiento de la aplicación debe ser mucho mejor que el que arrojaron las pruebas por lo que se evalúa como exitoso el resultado de las mismas.

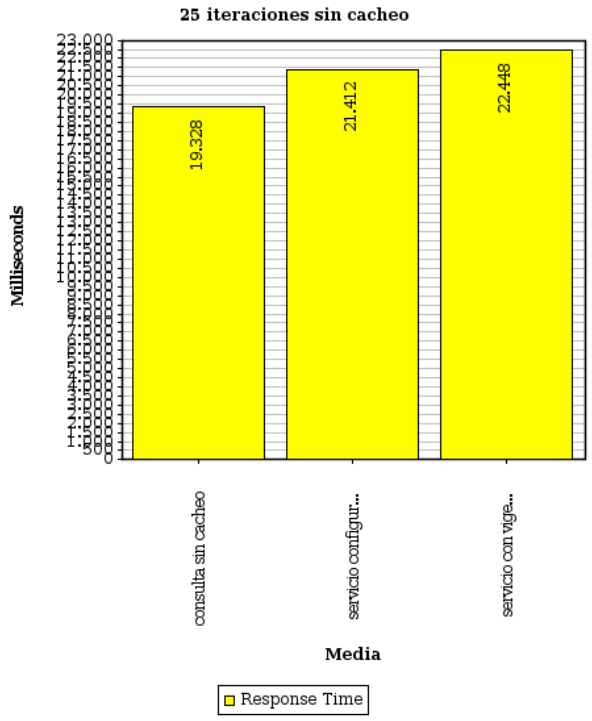


Fig. 18 Media en milisegundos para 25 iteraciones sin cacheo APC.

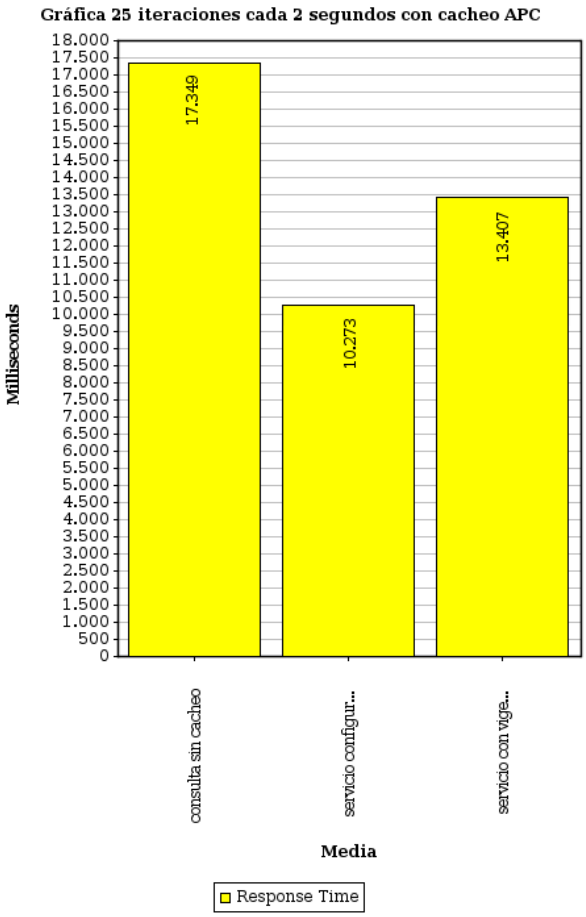


Fig. 18 Media en milisegundos para 25 iteraciones con cacheo APC

3.4.4. Validación de la Consistencia

Para validar la consistencia de los datos asociados a los nomencladores se comprobaron cada una de las deficiencias encontradas. A continuación se describe el mecanismo empleado para comprobar cada una de estas.

El hecho de no contar con un componente que se encargara de la gestión y control de los nomencladores de la VUCEC no permitía que se implementara algún mecanismo de seguridad para evitar que personal no autorizado con o sin intenciones afectara la consistencia de los datos asociados a los nomencladores. Para resolver esta deficiencia se incorporó el rol “ROLE_NOMENCLADOR” el cual deben poseer los usuarios competentes para acceder a las funcionalidades que brinda el componente Nomencladores.

Las tuplas repetidas son un claro ejemplo de inconsistencia de los datos almacenados en una Base de Datos. Esta deficiencia se corrigió a partir de la definición de una serie de campos que deben ser únicos para cada nomenclador(los campos código y descripción además del identificador correspondiente a cada tabla). Este factor se validó a partir de las pruebas de caja negra y las pruebas unitarias realizadas a las agrupaciones de requisitos Gestionar Nomencladores y Actualización de nomencladores con el GINA respectivamente por lo cual se concluye que no existen valores repetidos.

La integridad de entidad establece que la llave primaria debe tener un valor único para cada fila de la tabla. Esta se comprobó a partir de la revisión de que cada tabla utilice una secuencia para definir el valor de su llave primaria en cada inserción.

La Integridad Referencial garantiza que una entidad (fila o registro) siempre se relacione con otras entidades válidas y que no existan relaciones mal resueltas. Aun cuando el Sistema Gestor de Base de Datos garantiza este aspecto, es necesario verificarlo teniendo en cuenta que la información se gestiona a partir de las entidades de Symfony2 las cuales son generadas y configuradas manualmente, incluyendo sus relaciones con otras tablas. A partir de lo anterior se comprobó que cada entidad tiene configurado correctamente los campos de relaciones.

Se comprobaron las restricciones propias de cada atributo de las entidades a partir de las validaciones definidas como anotaciones en las mismas y las definidas en los formularios del marco de trabajo. Para la validación de las entidades, sus atributos y restricciones se aplicaron pruebas unitarias de software cuyos resultados se expusieron con anterioridad

3.5. Conclusiones del capítulo

En este capítulo se detallaron las principales técnicas utilizadas en la fase de implementación y pruebas.

A partir del uso de un estándar de codificación se logró que el código fuente generado pueda ser entendido y por consiguiente facilita las tareas de mantenimiento que pueda necesitar posteriormente.

El tratamiento de los errores realizado limita los problemas que puedan surgir en tiempo de ejecución, brindando una información más descriptiva de la solución de estos.

Con la aplicación de las pruebas de caja blanca, las de caja negra y las validaciones del diseño, implementación y variables que condicionan el origen del problema se comprobó que la solución cumple con los objetivos propuestos.

CONCLUSIONES

Las operaciones llevadas a cabo en la VUCEC requieren de la utilización de nomencladores los cuales favorecen la flexibilidad del sistema y posibilitan una actualización rápida del mismo. La investigación desarrollada parte de la identificación de algunas deficiencias en cuanto al trabajo con estos nomencladores entre las que se encuentran que la gestión actual de los mismos afecta el rendimiento del sistema y limita la consistencia de los datos de los nomencladores registrados.

El estudio de las soluciones informáticas arrojó que ninguna resuelve en su totalidad el problema de la presente investigación, además permitió identificar las principales tendencias y estándares utilizables para la definición e implementación de las tablas nomencladoras del componente desarrollado. También posibilitó identificar el comportamiento de Doctrine Softdeleteable como la solución para el control de la vigencia de los nomencladores de la VUCEC.

A partir de la aplicación de la disciplina Requisitos del modelo de desarrollo establecido para el CEIGE se definieron los requisitos que recogen las funcionalidades necesarias para garantizar que el componente satisfaga las necesidades del cliente.

Los artefactos generados durante la disciplina Análisis y Diseño conjuntamente con el buen uso de patrones de diseño sentaron las bases para la implementación del software con un alto grado de calidad.

Con la implementación se logró obtener un componente que permite gestionar los nomencladores de la VUCEC a través de una interfaz única. Este controla la vigencia de los mismos, emitiendo alertas y mostrando reportes a los administradores para el mantenimiento del sistema y que contribuye con el rendimiento del mismo, garantiza además la consistencia de los datos asociados a los nomencladores.

Con la aplicación de las pruebas de caja blanca, las pruebas de caja negra, la validación del diseño y de las variables que condicionan el origen del problema se comprobó que la solución cumple con los objetivos propuestos.

Al finalizar la investigación se obtuvo como resultado el componente de gestionar nomencladores de la VUCEC.

RECOMENDACIONES

Los objetivos trazados en el presente trabajo se cumplieron satisfactoriamente, sin embargo se recomienda:

- Implementar la opción de importar y exportar nomencladores a través de archivos.
- Implementar la opción de crear o generar las entidades y formularios dinámicamente.

REFERENCIAS BIBLIOGRÁFICAS

1. VUCE - Ventanilla Única de Comercio Exterior, [no date]. [online], [Accessed 8 February 2014]. Available from: https://www.vuce.gob.pe/resena.html
2. ARENCIBIA MORALES, Ing Annia and PÉREZ MALLEA, Msc Iván. Propuesta para disminuir el tiempo de desarrollo en aplicaciones informáticas que gestionen información poco variable en el tiempo. [online]. 29 October 2013. [Accessed 11 June 2014]. Available from: http://repositorio_institucional.uci.cu//jspui/handle/ident/7947
3. REDK - Openbravo ERP - Gestión de clientes - Proveedores - Productos - Datos maestros, [no date]. [online], [Accessed 9 February 2014]. Available from: http://www.redk.net/tecnologias/openbravoerp/funcionalidad/gestion-de-datos-maestros.html
4. Developers Manual/Introduction/es - OpenbravoWiki, [no date]. [online], [Accessed 9 February 2014]. Available from: http://wiki.openbravo.com/wiki/Developers_Manual/Introduction/es
5. Leandro Perez–Borroto Vivero, 2009, Procesos de Configuración y Carga Inicial del Sistema Integral de Gestión Cedrux. [online]. Habana/Cuba: Universidad de las Ciencias Informáticas. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_1959_09/1/TD_1959_09.pdf
6. Martin Belfiori, [no date], Que es SAP CO - SAP CO For Dummies. [online]. [Accessed 10 February 2014]. Available from: http://www.cvosoft.com/sistemas_sap_abap/recursos_tecnicos_abap/que-es-sap-co.php
7. Yaniris Blanco Zamora, 2009, Solución Informática para el Módulo Estructura y Composición del sistema Cedrux. [online]. Habana/Cuba: Universidad de las Ciencias Informáticas. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_2211_09/1/TD_2211_09.pdf
8. LETELIER, Patricio and CARMEN, Penadés. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *www.cyta.com.ar/ta0502/v5n2a1.htm* [online]. 15 April 2006. [Accessed 5 June 2014]. Available from: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm
9. William González Obregón. CEIGE-MODELO DE DESARROLLO DE SOFTWARE v1.2.
10. FAVRE, Liliana María, LEONARDI, María Carmen. Integración de técnicas orientadas al cliente y técnicas formales en el desarrollo de software con UML y RUP. In : V Workshop de Investigadores en Ciencias de la Computación [online]. 2003. [Accessed 5 June 2014]. Available from: http://hdl.handle.net/10915/21493

11. Qué es PHP. In: [online]. [Accessed 11 January 2014]. Available from: http://www.desarrolloweb.com/articulos/392.php.
12. MIGUEL ANGEL ALVAREZ. Qué es Javascript. In: [online]. [Accessed 22 January 2014]. Available from:http://www.desarrolloweb.com/articulos/25.php.
13. NetBeans IDE - NetBeans Rich-Client Platform Development (RCP). In: [online]. [Accessed 3 January 2014]. Available from: https://netbeans.org/features/platform/index.html.
14. ¿Qué es Symfony?, [no date]. symfony.es [online], [Accessed 7 March 2014]. Available from: http://symfony.es/que-es-symfony.
15. Servidor Apache HTTP. In: [online]. [Accessed 28 January 2014]. Available from: http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-httpd.html.
16. ¿Qué es Subversion? In: [online]. [Accessed 28 January 2014]. Available from: http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html.
17. Centro de las Naciones Unidas para Facilitación Del Comercio y las Transacciones Electrónicas (CEFACT/ONU), 2005, Recomendación y Directrices para el establecimiento de una Ventanilla Única. 2005.
18. Ian Sommerville, 2005, Requerimientos funcionales y no funcionales. In : Ingeniería de Software [online]. Séptima edición. Available from: http://eva.uci.cu/file.php/158/Documentos/Bibliografia_general/Textos_Complementarios/Ediciones_del_Sommerville/Sommerville_7ma_edicion/Sommerville_Parte_II_Requerimientos.pdf
19. Jenni Manso Martínez. Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE. [online]. Universidad de las Ciencias Informáticas, [no date]. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_03304_10/1/TD_03304_10.pdf
20. Eugenia Bahit, [no date], El paradigma de la Programación Orientada a Objetos en PHP y el patrón de arquitectura de Software MVC [online]. Available from: http://sunshine.prod.uci.cu/gridfs/sunshine/books/MVC_con_PHP.pdf.
21. Jason E. Sweat, 2005, php|architect's Guide to PHP Design Patterns A Practical Approach to Design Patterns for the PHP 4 and PHP 5 Developer.
22. UML_clase_05_UML_paquetes.pdf. In: [online]. [Accessed 23 April 2014]. Available from: http://www.codecompiling.net/files/slides/UML_clase_05_UML_paquetes.pdf.

23. Roger S. Pressman, Ingeniería de Software, un enfoque práctico. 2005.
24. SIRA. JURISTO, NATALIA, MORENO, ANA M. Y VEGAS. Técnicas de Evaluación de Software. 2005.
25. Javier Eguiluz. Desarrollo web ágil con Symfony2. 2012.
26. MIGUEL ANGEL ALVAREZ. Manual de jQuery [online]. [no date]. Available from: http://www.desarrolloweb.com/manuales/manual-jquery.html.
27. Alvaro Fontela Sanchez. ¿Que es Bootstrap? [online]. Available from: http://openwebcms.es/2013/que-es-bootstrap/
28. Craig Larman. Uml y patrones. [no date]. 2da. ISBN 8420534382.
29. Ismail Vega Mena. Diseño e Implementación del Módulo Despacho de los Medios de Transporte Marítimos para el Sistema de Gestión Integral de la Aduana [online]. Universidad de las Ciencias Informáticas, 2014. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04247_11/1/TD_04247_11.pdf
30. ENRIQUE ROBERTO POMPA TORRES, Lianet Cabrera González. Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información. 15 October 2012. P. 11.
31. GGARAY. Openbravo: Ejemplo de Patrón de Diseño MDD | Init Developers. [online]. [Accessed 13 June 2014]. Available from: http://blog.theinit.com/2011/05/10/openbravo-ejemplo-de-patron-de-diseno-mdd/
32. Soft Delete Model - Orm Package - FuelPHP Documentation. [online]. Available from: http://fuelphp.com/docs/packages/orm/model/soft.html
33. Query. Wikipedia [online]. [Accessed 16 June 2014]. Available from: http://es.wikipedia.org/wiki/JQuery
34. Maidely Calderón Montero. Propuesta de procedimiento para la captura de requisitos en los proyectos de Portales de la UCI [online]. 2007. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_0929_07/1/TD_0929_07.pdf
35. Patrones de diseño de bases de datos [online]. Available from: http://eva.uci.cu/file.php/180/2._Clases/Tema_1/Materiales_basicos/4.Patrones_de_diseno_de_BD.pdf