Universidad de las Ciencias Informáticas

Facultad 3

Módulo para la administración de usuarios, roles, permisos, trazas y nomencladores en el Sistema de Gestión de Índices de Precios de la Oficina Nacional de Estadística e Información

---

Trabajo de diploma para optar por el título de
Ingeniero en CienciasInformáticas

Autor:

Rene Sánchez Linares

Tutores:

Ing. Aldis Joan Abreu Medina
Ing. Luis Javier Rodríguez Castro

La Habana, 2020

Declaración de Autoría

Declaro ser el único autor de este trabajoy concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de este, con carácter exclusivo.

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

_____

Firma del autor
Rene Sánchez Linares

_____

Firma del tutor
Ing. Aldis Joan Abreu Medina

_____

Firma del tutor
Ing. Luis Javier RodríguezCastro

Agradecimientos

Agradecimientos

A un paso de cumplir uno de mis grandes sueños que es ser ingeniero, antes que todo quisiera agradecer de forma especial a mis padres y a mi hermana que han estado al lado mío en estos largos 5 años, por ellos me gradué y a ellos le debo todo lo que soy, este título es más por ellos que por mí, a mi familia que siempre estuvo apoyándome en todo lo que me hiciera falta, también a ellos les debo mucho, agradecido con mis tutores Aldis y Luis Javier, que me ayudaron en todo el proceso, a los profesores Yordanis y Zenel que me guiaron en esta etapa, darles mis más profundos agradecimientos a Sandy que a pesar de no ser mi tutor siempre estuvo al pie del cañón conmigo preocupándose de que mi trabajo estuviera de la mejor forma posible, junto con mi compañera de año Yuni que desde que empezó este proceso siempre estuvo cerca de mi jalándome las orejas y nunca me dejo de apoyar ni de estar a mí lado. Por último, quisiera agradecerles tanto a mis hermanos de san miguel que siempre me preguntaban cómo estaba en la escuela, de cómo estaba saliendo en las pruebas y cuando me iba a graduar, a ellos muchas gracias por todo. Estos hermanos que me pusieron el sobre apodo de La Uci con todo cariño. Que decir de la gran familia que me dio la UCI, ellos tienen tanto valor como mi título, son 5 años de fiestas, padrino, gallega, 45,jodederas en la residencia y en el comedor, festivales, delicias, pinos, guarapera, piscina, wilfredolam, canchas a las 5, parque de la leche y pizzería. Ni hablar de todo lo que formábamos en el docente y en el aula, solo el que compartió aula conmigo sabe que era una hora y media de profundo relajo, pero con orden, esos asientos al final del aula donde se sentaban los jodedores era lo mejor sin duda de todo el turno de clases. Viene lo bueno, los Juegos Mella, hablo de esto y se me eriza la piel, los mejores de recuerdos como universitario los viví en estos juegos, participe en la lucha, beisbol 5, pero lo que se quedó grabado en mi corazón para siempre fue el fútbol sala donde obtuve la gran ansiada medalla de oro por mí y por mi facultad 3, que partido en la final, 1 a 1 y a penales, un partido no apto para cardíacos, y por último el Futbol 11, mi primer gran deporte en los mella, donde a día de hoy tengo la gran responsabilidad de ser el capitán, 4 años donde obtuve dos medallas de plata y dos de oro, lo mejor de estos juegos no fueron las medallas obtenidas sino las amistades que hice ahí, a todos los llevo en el corazón. Mi gran familia UCI sin orden de prioridad, César, Fundichely,Choco,Leonel Acosta, Fernando, Yanislaidi, Raudel, Damián, Nelsito, Silvio, Rodney, Mauro Sergio, Obel, Yahima, Ray, Cristian, Naisel, Rafael Rubén, Kaler, Dary, Chumi, Rachel, Picachu, Reinier, AdriánZaldívar, Ariel, Dayana Oñate, Ana Laura, Ana

Agradecimientos

Karla, Miguel Hurtado, Ángel, Mauro CC, Dayana Almaguer; Daniel el puerta, Cristofer, Félix, Rolando Ramírez, Loandry, Rafael David, Luis Daniel, Yoanderley, Eilen, Yonatan, Leonardo, YasmanySánchez, Wendy, Yai, Eloy, Annet, Oslén, Alexis, Yanarys, Raimer, Rafael Aguilera, Sergio, Marlon, Raúl, Eider, Cuco, Yadira, Manuel, Aldo, Yamilka, Yusnaby, Naila, Jorge, Raykof, Juampa, Drake, Kende, Nuñez, Carlos David, Grabiel, Anett, Rainer, Miguel Diaz, Yulian, Yannier, José Ernesto, Leovanis, Yanniel, Miraida, Miguel Ángel, Ediel, Raynel, Addiel, Astencio, Guido, Colorao , Aponte, Carlos Pardo, Enrico, Noel, Josué, Lucky, Yohao, Elena, Alejandro Varadero, Juan, Yasmany Lazo, Culito, Ronal, Yordan, Martín, Alfredo, Dacho, Liliana, Cristian Mbappe, Lobaina, Carlos Ragnar, Nelson el Cabra, Alejandro Guerra, Pichi, Eddy, Yinet, Felix LM, Betty, Rainel, Omar, Jimagua, David Lemuel, El Moro, Yaili, Sael, Yasser, Ali, Adilson, Piola, Cuza, Albert, Amanda, Anayanci, Asley, Camilo, Isla, Carlos CA, Carlos Ocampo, Chris, José Eduardo, Faustino, Luis Joel, Rachel Ávila, Ramsés, Yoandry, Yusnavi, Luis Miguel, Camilo, Yosvani, Miguel Ulloa, Yoyi, Víctor, Wendy, Adriana, Susel, Rolando Ramírez, Orel, Naile, Karla, Mónica, Miguel Barber, Leo Locutor Marquito, María, Marco Rivas, Marco el Puerta , Leote, Nacho, Frank, Favián, Eliany, Sharawy, Abdiel,Yoisbel, Yisel y a los míos del Big Team Marsan, Daniel, Patricia, Tania, Amalia, Daniel David, Ivan, Jennifer, Karla, Freddy, Chris, Shabely, José Enrique, Javier, Adrián, a ustedes que de gran cariño me dicen la bandera de la uci, siempre los llevare en mi corazón. A todas estas personas las quiero con la vida, mil gracias por todo.

Dedicatoria

Dedico este trabajo a mis padres, hermana, familia, hermanos de san miguel, y a mi gran familia de la UCI.

Resumen

Los índices de precios constituyen una medida estadística que se calcula sobre los precios de los productos de consumo masivo en un determinado período.La Oficina Nacional de Estadística e Información tiene como propósitoinformatizar los procesos que realiza la Dirección de Índices de Precioen esta oficina. En la actualidad el desarrollo de soluciones informáticas se caracteriza por el uso de arquitecturas que permiten el consumo de servicios web. Entre las tecnologías que brindan la posibilidad de desarrollar este tipo de soluciones se encuentran transferencia de estado representacional y GraphQL. Estas permiten consultar recursos desde clientes web a través de servicios. Sin embargo, GraphQL se define como un lenguaje de consulta que supera a la transferencia de estado representacional. La presente investigación tiene como objetivo desarrollar un módulopara la administración de usuarios, roles, permisos y trazas. El proceso de desarrollo de software está guiado por el uso de la metodología Proceso Unificado Ágil variación UCI. En la implementación del módulo se utilizan tecnologías de código abierto. El resultado de la investigación fue validado utilizando una estrategia de pruebas en los niveles de unidad y aceptación. Además, se establecen un conjunto de criterios que permiten evaluar a través de un antes y un después la relación causa-efecto de las variables de la investigación.

Palabras claves: gestionar, índice, informático, precios, sistema

ÍNDICE DE CONTENIDO

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los sistemas informáticos son tipos de sistemas que se organizan en torno al manejo de datos de diversa naturaleza, aunque no todos los sistemas de información sean informáticos. Esto es, no todos son digitales, ni automatizados, ni electrónicos. Estos sistemasocupan en el mundo contemporáneo un lugar clave para la organización humanade sus procesos productivos y de otras naturalezas. Es una herramienta poderosa para el intercambio de información y la construcción de redes informáticas que superan la dificultad de las distancias(Uriarte, 2020).

Los índices de precios se pueden clasificar en dos grandes grupos: simples y complejos(Sánchez, 2017):

Índices simples:

Los índices simples se calculan como el cociente entre el precio en el periodo concreto y el precio en el momento que tomamos de base. Sin embargo, en la práctica los índices que se usan son los complejos.

Índices complejos:

Los índices complejos utilizan más magnitudes para dotar al índice de realismo. Usan la media de la evolución de los precios (o de cantidades o de índices), en vez de coger el valor del precio en un momento concreto, como en los índices simples.

Los índices de precios más utilizados son los siguientes:

- Índice de precios al consumo (IPC): Es un indicador que expresa la evolución de los precios de una cesta de bienes y servicios representativa del consumo por parte de las familias. Es el índice de precios más utilizado de todos yen ocasiones, muchas medidas gubernamentales se toman en función del IPC (por ejemplo, subida de salarios, de pensiones, etc.)
- Índice de precios al consumo armonizado (IPCA): Es un indicador similar al IPC, pero se calcula a partir de una cesta de bienes y servicios común para todos los países de la zona euro. Esto facilita la comparación del IPCA de varios países, al usar todos ellos la misma metodología.
- Deflactor del PIB: Es otra forma de calcular el incremento de precios (es decir, la inflación), mediante el cociente entre el PIBnominal y el PIB real.
- Índice de precios industriales (IPRI): Este índice mide la variación de los precios de productos industriales, excluyendo la construcción. Entre los productos industriales se incluyen la industria manufacturera, el suministro de luz, gas y agua o la industria extractiva.

- Índice de precios percibidos y pagados por los agricultores: Este índice mide la evolución de los precios de los productos del sector agrario.

La Oficina Nacional de Estadística e Información (ONEI), fue creada a partir de lo definido en el artículo 31 del Decreto Ley No. 281 del 2 de febrero de 2011, como resultado de la organización del Sistema de Información del Gobierno. Esta oficina contribuye a satisfacer las necesidades informativas relacionadas con los objetivos y planes del Gobierno en todos los niveles de dirección, en los ámbitos; económico, social, demográficoy medioambiental.

La ONEI tiene entre sus objetivos avanzar en las mediciones en los ámbitos económico, social, demográfico y medioambiental. Para el caso del análisis de temas económicos, la oficina cuenta con la Dirección de Índices de Precios, en la cual se gestionan los siguientes índices:

- Índice de precios al consumidor.
- Índice de precios de la construcción.
- Índice de precios de la producción de la industria manufacturera.
- Índice de precios de la comercialización mayorista de la industria manufacturera.

En la actualidad esta dirección cuenta con sistemas informáticos que gestionan de forma independiente los cuatro índices antes mencionados. Cada uno de estos sistemas son diferentes tanto en el negocio, como en la tecnología con la cual fueron implementados. Lo anterior provoca que procesos importantes como la gestión de usuarios no se realice de la misma forma en cada sistema. En el sistema destinado a la gestión del índice de precios al consumidor, la gestión de usuario se encuentra estructurada a través de niveles. Esto permite que cada usuario tenga privilegio para acceder solo a la información que le corresponde. Sin embargo, en el resto de los sistemas que gestionan los otros tresíndices, todo el proceso se realiza con un usuario único.

En la actualidad el departamento de índices de la ONEI cuenta con un sistema informático que gestiona los índices de precios de la construcción. Este software constituye una aplicación de escritorio con base de dato local, que dificulta el logro de una interconexión entre cada una de las sedes de la ONEI a nivel de país, que permita mantener una actualización constante de la información que seprocesa. Además, el sistema está implementado en las tecnologías Visual Basic y sistema gestor de base de datos Access, las cuales constituyen tecnologías privativas que no se corresponden con la política de soberanía tecnológica que aplica el país.

La forma en que la ONEI gestiona cada uno de los índices antes mencionados, tiene como limitante que el trabajo se realiza de forma independiente, faltando una estandarización que homogenice el proceso. Todo lo anterior, dificulta a los especialistas de la Dirección de Índices de Precios, gestionar de forma centralizada los usuarios, nomencladores, roles, permisos y trazas

comunes para el trabajo con cadaíndice. Esto trae como consecuencia que en ocasiones exista duplicidad de información y esfuerzo.

A partir de la problemática antes descrita se genera la necesidad de resolver el siguiente **problema de investigación**:¿cómo gestionar los usuarios, nomencladores, roles, permisos y trazas relacionados con la gestión de índices de precios en la ONEI, de forma tal que se contribuya a la estandarización en el proceso, evitando duplicidad de información?

Teniendo en cuenta el problema que se describe anteriormente se define como**objeto de estudio:** aplicaciones informáticas de gestión.

Determinándose como **objetivo general:**desarrollar un módulo para la administración de usuarios, roles, permisos, trazas y nomencladores en el Sistema de Gestión de Índices de Precios de la ONEI, de forma tal que se contribuya a la estandarización en el proceso, evitando duplicidad de información.

Del objetivo general, se desglosan los siguientes**objetivos específicos:**

- Elaborar el marco teórico de la investigación mediante un estudio de los referentes teóricos sobre el desarrollo de soluciones informáticas dirigidas a la gestión de usuarios, roles, permisos, trazas y nomencladores de una aplicación informática de gestión.
- Realizar la identificación de los requisitos, análisis, diseño e implementación del módulopropuesto.
- Validar el diseño y el funcionamiento del módulopropuesto, aplicando métricas y pruebas de software respectivamente.
- Verificar el cumplimiento de la relación causa efecto de la variable independiente sobre las variables dependientes de la investigación.

Se identifica como**campo de acción:** la gestión de usuarios, roles, permisos, trazas y nomencladores en lasaplicaciones informáticas de gestión.

Definiéndose como**idea a defender:** sisedesarrolla elmódulopara la administración de usuarios, roles, permisos, trazas y nomencladores en el Sistema de Gestión de Índices de Precios de la Oficina Nacional de Estadística e Información, se contribuirá a la estandarización en el proceso, evitando duplicidad de información.

Métodos teóricos:

- **Histórico-lógico**: permitiórealizar un estudio sobre la evolución y el desarrollo de aplicaciones informáticas de gestión paraadquirir conocimientos sobre los sistemas informáticos que gestionan los índices de precios en la Oficina Nacional de Estadística e Información.

- **Analítico-sintético**: posibilitó la realización del estudio teórico de la investigación haciendo más sencillo el proceso de análisis de los documentos y la extracción de los elementos fundamentales a tener en cuenta para desarrollar el módulo.
- **Modelación:** se empleó para el diseño de los prototipos de interfaz de usuario del componente y la realización del modelo de diseño.

Métodos empíricos:

- **Entrevista:** se utilizópara recoger el conjuntode información referente al cliente y al negocio, así como todo lo que se necesita para la comprensión de la totalidaddel proceso en general.

El presente trabajo de diploma está estructurado en 3 capítulos de la siguiente forma:

Capítulo 1: Fundamentación teórica:
En el presente capítulo se describen los principales conceptos relacionados con la investigación, fundamentándose el objeto de estudio y el campo de acción.Se realiza un análisis de sistemassimilares que existen en Cuba y en otros países del mundo. Además, se describen las principales características de la metodologíade desarrollo de software utilizada, junto a los lenguajes y herramientas.

Capítulo 2: Descripción de la propuesta de solución:
En este capítulo se realiza una descripción de la propuesta de solución. A partir de la metodología definida, se obtienen en la disciplina de requisitos los requisitos funcionales y no funcionales del módulo que se propone. En la disciplina de análisis y diseño se aplica el patrón arquitectónico, los patrones de diseño, los diagramas de clases del diseño y el modelo de base de datos. Y por último los estándares de codificación que se utilizan como conjunto de reglas no formales.

Capítulo 3: En este capítulo se muestran los resultados obtenidos en la validación del diseño de la solución aplicando las métricas Tamaño Operacional de Clases y Relaciones entre Clases. El capítulo concluye con la evaluación de la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación, a partir de la definición de un conjunto de indicadores que permiten chequear el comportamiento de esta relación antes y después del desarrollo del módulo propuesto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se describen los principales conceptos relacionados con la investigación, fundamentándose el objeto de estudio y el campo de acción. Se realiza un análisis de sistemas similares que existen en Cuba y en otros países de mundo. Además, se describen las principales características de la metodología de desarrollo de software utilizada, junto a los lenguajes y herramientas.

1.1 Conceptos principales relacionados con el tema

Para una mejor comprensión del tema, es necesario conocer los conceptos que a continuación seenuncian:

Índice de precios:es un número índice, una medida estadística que se calcula sobre los precios de los productos de consumo masivo en un determinado periodo (Uchoa, 2014).

Gestionar:se refiere a ocuparse de la administración, organización, coordinación, y funcionamiento de una empresa o compañía y de sus recursos humanos y económicos, con la finalidad de lograr un conjunto de objetivos concretos (Gestionar, 2016).

Sistema Informático: es un sistema automatizado de almacenamiento, procesamiento y recuperación de datos, que aprovecha las herramientas de la computación y la electrónica para llevar a cabo su serie compleja de procesos y operaciones. En otras palabras, un sistema informático es un computador de alguna índole(Uriarte, 2020).

1.2 Valoración de sistemas homólogos

Con el objetivo de obtener información y realizar un mejor análisis del tema se realizó un estudio de institucionesque se caracterizan por la gestión de información estadística a escala nacional e internacional. Esta información se obtuvo de los siguientes sitios: Departamento Administrativo Nacional de Estadística (DANE), Instituto Nacional de Estadística (INE), Instituto Nacional de Estadística de Bolivia (INE), y la Oficina Nacional de Estadística e Información (ONEI).

Departamento Administrativo Nacional de Estadística (DANE): En octubre de 1951 mediante el Decreto 2240, se separa la Oficina Nacional de Estadística de la Contraloría General de la República, es así como se crea la Dirección Nacional de Estadística, dependencia directa de la Presidencia de la República. Tiene como misión Planear,

implementar y evaluar procesos rigurosos de producción y comunicación de información estadística a nivel nacional, que cumplan con estándares internacionales y se valgan de la innovación y la tecnología, que soporten la comprensión y solución de las problemáticas sociales, económicas y ambientales del país, sirvan de base para la toma de decisiones públicas y privadas y contribuyan a la consolidación de un Estado Social de Derecho equitativo, productivo y legal.

Instituto Nacional de Estadística (INE): El Instituto Nacional de Estadística es un organismo autónomo de carácter administrativo, con personalidad jurídica y patrimonio propio, adscrito al Ministerio de Economía y Empresa a través de la Secretaría de Estado de Economía y Apoyo a la Empresa. Se rige, básicamente, por la Ley 12/1989, de 9 de mayo, de la Función Estadística Pública (LFEP), que regula la actividad estadística para fines estatales la cual es competencia exclusiva del Estado, y por el Estatuto aprobado por Real Decreto 508/2001 de 11 de mayo. También, la ley atribuye al INE las siguientes funciones: la formulación del Proyecto

Estadístico Nacional con la colaboración de los Departamentos Ministeriales y del Banco de España; la propuesta de normas comunes sobre conceptos, unidades estadísticas, clasificaciones y códigos; y las relaciones en materia estadística con los Organismos Internacionales especializados y, en particular, con la Oficina de Estadística de la Unión Europea (EUROSTAT).

Instituto Nacional de Estadística de Bolivia (INE): Es el órgano ejecutivo del Sistema Nacional de Información Estadística de Bolivia, tiene las funciones de relevar, clasificar, codificar, compilar y difundir con carácter oficial, la información estadística del país. El Decreto Ley No 14100 es el instrumento legal que norma el funcionamiento actual del INE. Es la entidad técnica encargada de producir, analizar y difundir información estadística oficial y de calidad, así como normar, coordinar y promover el sistema nacional estadístico para el desarrollo del país.

Oficina Nacional de Estadística e Información (ONEI): En Cuba se encuentra la Oficina Nacional de Estadística e Información (ONEI) que no presenta un software paragestionar de forma centralizada los usuarios, nomencladores, roles, permisos y trazas comunes para el trabajo con cada tipo de índice, faltando una estandarización que homogenice el proceso.

Conclusiones de la valoración de sistemas homólogos:

Al realizar un profundo análisis y estudio se arribó a la conclusión de que se necesita desarrollar un componente que dé solución al problema de investigación dando cumplimiento al objetivo general.

1.3 Metodología de desarrollo de software

Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, se las puede dividir en Ágiles y Tradicionales o Robustas(Castillo, 2018).

Metodologías agiles:son aquellas metodologías de gestión que permiten adaptar la forma de trabajo al contexto y naturaleza de un proyecto, basándose en la flexibilidad y la inmediatez, y teniendo en cuenta las exigencias del mercado y los clientes. Los pilares fundamentales de las metodologías ágiles son el trabajo colaborativo y en equipo (Kezmo, 2017).
Metodologías robustas:Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada(Castillo, 2018).

El Proceso Unificado Ágil(AUP por sus siglas en inglés de *AgileUnified Process*), constituye una versión simplificada del Proceso Unificado Racional (RUP por sus siglas en inglés de*RationalUnified Process*), desarrollada por Scott Ambler. Esta metodología combinaprocesos propios del concepto unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad. Permitiendo así describir de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio. AUP aplica técnicas ágiles, entre las que se incluyen(Rodríguez, 2018):

- El desarrollo dirigido por pruebas.
- El modelado ágil.
- La gestión de cambios ágil.

- La refactorización de bases de datos para mejorar laproductividad.

Variación de AUP para la UCI:

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se utiliza una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Surgiendo la variación de AUP para la UCI(Rodríguez, 2018).

Descripción de las Fases:

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la que se nombra Ejecución y se agrega la fase de Cierre (Rodriguez, 2015).

Inicio: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

Ejecución: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Cierre: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Disciplinas:

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y

Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Rodriguez, 2015).

Escenarios para la disciplina Requisitos:

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN),Diagrama de Proceso del Negocio(DPN)y Modelo Conceptual(MC)). Existen tres formas de encapsular los requisitos (Caso de Uso del Sistema(CUS),Historias de Usuarios(HU),Diagrama de Requisitos por Proceso(DRP)). Surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

- **Escenario No 1:** proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.
- **Escenario No 2:** proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS**.**Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no sea necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.
- **Escenario No 3:** proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

- **Escenario No 4:** proyectos que no modelen negocio solo pueden modelarel sistema con HU. Se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información (Rodríguez, 2018).

El escenario a utilizar es el No.4, ya que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido, estas características se adaptan a la solución propuesta.

1.4 Lenguajes y herramientas

A continuación, se describen los siguientes lenguajes y herramientas que se utilizan.

1.4.1 Spring Boot 2.0.6

Como marco de trabajo para el lado del backendse decide utilizar Spring Boot, es una herramienta que nace con la finalidad de simplificar aún más el desarrollo de aplicaciones basadas en el ya popular marco de trabajo Spring Core. Spring Boot busca que el desarrollador solo se centre en el desarrollo de la solución, olvidándose por completo de la compleja configuración que actualmente tiene Spring Core para poder funcionar(oblancarte, 2019).

Spring Boot, publicado en 2012, es una solución para el marco de trabajoSpring de Java que sigue el principio de “convención sobre configuración” y reduce la complejidad del desarrollo de nuevos proyectos basados en Spring. Para ello, Spring Boot proporciona la estructura básica configurada del proyecto, que incluye las pautas para usar el marco y todas las bibliotecas de terceros relevantes para la aplicación, lo que nos allana el camino para comenzar a desarrollarla lo más rápidamente posible. De esta manera se simplifica mucho la creación de aplicaciones independientes y reproducibles, por lo que la mayoría de las nuevas aplicaciones basadas en Springse desarrollan con Spring Boot(Boot, 2019)**.** Spring Boot centra su éxito en las siguientes características que lo hacen extremadamente fácil de utilizar:

- **Configuración:** Spring Boot cuenta con un complejo módulo que autoconfigura todos los aspectos de nuestra aplicación para poder simplemente ejecutar la aplicación, sin tener que definir absolutamente nada.

- **Resolución de dependencias:**con Spring Boot solo hay que determinar qué tipo de proyecto estaremos utilizando y él se encarga de resolver todas las librerías/dependencias para que la aplicación funcione.
- **Despliegue:** Spring Boot se puede ejecutar como una aplicación Stand-alone, pero también es posible ejecutar aplicaciones web, ya que es posible desplegar las aplicaciones mediante un servidor web integrado, como es el caso de Tomcat, Jetty o Undertow.
- **Métricas:**por defecto, Spring Boot cuenta con servicios que permite consultar el estado de salud de la aplicación, permitiendo saber si la aplicación está prendida o apagada, memoria utilizada y disponible, número y detalle de los Bean's creado por la aplicación, controles para el prendido y apagado, etc.
- **Extensible:** Spring Boot permite la creación de complementos, los cuales ayudan a que la comunidad de Software Libre cree nuevos módulos que faciliten aún más el desarrollo (oblancarte, 2019).

1.4.2 JavaScript ECM 6

Como lenguaje de programación del lado del frontendse seleccionó JavaScript, es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (javascript, 2006-2020).

1.4.3 Java 1.8.0

Como lenguaje de programación del lado del backendse seleccionó Java, es un lenguaje de programación de propósito general orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible permitiendo a desarrolladores escribir un programa y ejecutarlo en cualquier tipo de dispositivo sin tener que compilarlo una y otra vez.

Como un lenguaje de programación orientado a objetos (POO) el programador puede generar fragmentos de código autónomo, que puedan interactuar con otros objetos para resolver un problema ofreciendo soporte para diferentes tecnologías. De hecho, es común referirse también Java como un conjunto de tecnologías en referencia a los diferentes productos y versiones que componen su familia (Robledano, 2019).

Es un lenguaje con una curva de aprendizaje baja (se puede decir que es fácil de aprender) y que dispone de una gran funcionalidad de base (incrementada por la gran cantidad de código de terceros existente). Java, como lenguaje de programación, ofrece un código robusto, que ofrece un manejo automático de la memoria, lo que reduce el número de errores (java, 2019).

1.4.4 VueJs 2.1.8

Es un marco de trabajo que se va a utilizar del lado del frontend, a diferencia de otros marcos de trabajomonolíticos, Vue está diseñado desde cero para ser utilizado incrementalmente. La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar sofisticadasaplicaciones de página única(SPA por sus siglas en inglés de*Single-Page Applications*) cuando se utiliza en combinación con herramientas modernas y librerías de apoyo(vue, 2014-2020).
El hecho de ser un marco de trabajoprogresivo no es la única característica interesante de Vue.js, entre otras también destacan:

- Es un marco de trabajointuitivo, todos sus componentes están justo donde deben estar.
- Es muy fácil de aprender, si ya tienes conocimientos en otros marcos de trabajote costará poco aprender Vue, verás que incluso es más sencillo que otras herramientas de este tipo. Y si no has usado un marco de trabajoantes igual te será fácil aprender. Así que la experiencia previa no es una limitante.
- Incluye plugins que permiten hacer peticiones http, validar formularios, y otros.
- El código fuente es muy claro, sin complicaciones, por lo tanto es fácil de entender y mantener.
- Es un marco de trabajoque destaca por su velocidad, su biblioteca pesa poco, además a diferencia de otros marcos de trabajonunca realiza comprobaciones, por lo que optimizar requiere mínimos esfuerzos.

- Vue permite crear archivos con extensión vue que incluyen tanto el HTML como el CSS y el JavaScript. Todo lo necesario en un solo paquete.
- Es fácil de integrar con otras librerías.
- El core pesa muy poco, apenas unos Kb, lo cual reduce el tiempo de carga de las aplicaciones web desarrolladas.
- Puede implementarse en el HTML de cualquier proyecto sin requerir grandes cambios.
- Dispone de una extensión para el navegador Chrome llamada Vue.js devtools que permite entre otras cosas ver el proceso de renderizado del árbol de componentes o cómo se está comportando el estado global de la aplicación. En general con la extensión podemos descubrir y solucionar errores con facilidad.
- La documentación es abúndate. Hay disponible una gran cantidad de manuales, tutoriales y otros recursos que te ayudarán no solo a aprender a usar Vue sino a realizar casi cualquier cosa que quieras.
- Es un proyecto open source, gestionado, desarrollado, evolucionado y planteado por y para la comunidad.
- Hay una comunidad en continua evolución y aprendizaje, de hecho, es una de las comunidades más dinámicas de este tipo de herramientas, y ha creado una gran cantidad de recursos para el aprendizaje (ArturoJS, 2018).

1.4.5 Vuetifyjs 1.5

El marco de trabajo para interfaces de usuario Vuetifycombinala potencia del popular VueJs con la estética de Material Design. Permite acelerar el desarrollo de aplicaciones web complejas, incorporando una gran cantidad de componentes "listos para usar". Vuetify se basa en el habitual sistema tipo "grid" para la ordenación del layout de la página. Dispone de una enorme librería de componentes que incluye desde elementos de formulario sencillos como botones, combobox, inputs, sliders, a componentes más avanzados típicos en aplicaciones Android como "cards" o "snackbars" (LLamas, 2019).
Esta librería permite realizar componentes en Vue de una forma muy sencilla y expresiva. Esto significa que las interfaces van a tener un aspecto profesional sin necesidad de tener un conocimiento avanzado de CSS. Por lo tanto, recomiendo encarecidamente para utilizarla junto a VueJs (Llobet, 2019).

1.4.6 IntelliJ IDEA 2018.3.5

Es un entorno de desarrollo integrado(IDE por sus siglas en inglés de *IntegratedDevelopmentEnvironment*)multiplataforma para Java. El IDE puede extenderse mediante numerosos complementos que harán de un programa todavía más completo. El usuario se dará cuenta de que cada aspecto de IntelliJ IDEA está diseñado para maximizar la productividad del desarrollador. Tanto el potente análisis de código estático como el diseño ergonómico del programa hacen que el desarrollo no sólo sea productivo (Amoedo, 2017).

Intellij IDEA intenta construir un IDE con un poder similar al de Eclipse, pero con un acabadoen la parte superior. Los desarrolladores tendrían la ventaja de usar IDEA debido a las muchas herramientas y ganchos que tiene para ahorrar tiempo en todos los proyectos. La finalización inteligente del código, la integración de la prueba de la unidad nativa y la administración nativa de Gradle son solo algunos de los aspectos más destacados del IDE Java de Jetbrain (intellij-idea).

1.4.7 WebStorm 2018.3.5

Es un entorno de desarrollo integradoligero pero potente, perfectamente equipado para el desarrollo complejo del lado del cliente y el desarrollo del lado del servidor, es multiplataforma y funciona en Windows, Mac OS X y Linux. WebStorm ofrece soporte avanzado para JavaScript , HTML , CSS y sus sucesores modernos, así como para marcos como AngularJS o React , depuración e integraciónconel VCS y varias herramientas de desarrollo web. Además, proporciona un entorno de desarrollo local configurado y listo para usar, que incluye soporte para Node.js , Meteor , CoffeeScript , TypeScript , Dart , Sass y más (webstorm).

1.4.8 PostgreSQL 9.5

Es el sistema de Gestor de Base de Datos a utilizar en esta investigación, es un servidor de basede datos objeto relacional libre, ya que incluye característicasde la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional, liberado bajo la licencia BSD. Como muchos otros proyectos de código abierto el desarrollo de PostgreSQL no es manejado por una sola compañía, sino que es dirigido por una comunidad de desarrolladores

(PGDG por sus siglas en inglés de *PostgreSQL Global Development Group*) y organizaciones comerciales las cuales trabajan en su desarrollo (Ibarra, 2008).
Entre sus ventajas están:

- Ampliamente popular - Ideal para tecnologías Web.
- Fácil de Administrar.
- Su sintaxis SQL es estándar y fácil de aprender.
- Footprint bajo de memoria, bastante poderoso con una configuración adecuada.
- Multiplataforma.
- Capacidades de replicación de datos.
- Soporte empresarial disponible (postgressql, 2012).

1.4.9GraphQL

Es la tecnología que se va a utilizar para conectar el backend con el frontend, GraphQL es un protocolo de consulta de datos originalmente elaborado por Facebook para uso interno. Desde 2015 existe en una versión open source que ha impulsado su desarrollo y que ha generado muchas ventajas y beneficios prácticos. Si está familiarizado con el funcionamiento de las aplicaciones, sabrá que los protocolos de consulta de datos son esenciales para obtener toda clase de reportes e información. REST es el mecanismo más utilizado, pero ahora GraphQL ha surgido como una alternativa que pretende adaptarse mejor a las necesidades actuales.El objetivo esencial de GraphQL es ofrecer a los clientes una forma más directa, sencilla y eficiente para obtener exactamente los datos que requieren, a través de un protocolo potente y dinámico(aplyca, 2018).

1.5 Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema en particular del diseño dentro de un contexto específico y entre fuerzas que afectan la manera en que se aplica y en la que se utiliza dicho patrón (Pressman, 2010).

Los patrones de diseño se asocian usualmente con el diseño orientado a objetos. Los patrones publicados se suelen apoyar en características de objetos como herencia y polimorfismo para dar generalidad. Sin embargo, el principio universal de encapsularla experiencia en un patrón es igualmente aplicable a cualquier tipo de diseño de software. Los patrones son una forma de reutilizar el conocimiento y la experiencia de otros diseñadores(Sommerville, 2011).

Los patrones de diseño se pueden agrupar en dos grandes grupos; los GRASP (por sus siglas en inglés *General ResponsibilityAssignment Software Patterns*), que son patrones generales de software para asignación de responsabilidades y los GOF (por sus siglas en inglés *GangofFour*), encargados de la inicialización, agrupación y comunicación de los objetos. En el Capítulo 2 de la presente investigación, se describen los patrones de diseño que fueron empleados en el desarrollo del componente propuesto.

Patrones GOF:

Los autores del GoF proponen dos principios básicos(cleformacion, 2017):

- Programar orientado a una interfaz y no a una implementación.

Este principio busca que el código se utilice como una herramienta que no revele su funcionamiento. La clásica metáfora de la “caja negra” implementada a través del principio de la encapsulación. De este modo, los usuarios del código se centran en la firma abstracta, lo que puede hacer y no puede hacer el código, en vez de en el cómo lo hace. Se establece una separación fundamental entre el programador del código -la persona que lo crea y lo mantiene- y el usuario del código -entendido como el desarrollador que utiliza la librería sin modificar su funcionamiento-. Dicho de otro modo, los patrones GoF buscan proporcionarnos piezas con instrucciones claras sobre con qué se pueden conectar y qué hacen, pero que no nos dicen nada sobre su funcionamiento interno.

- Favorecer la composición antes que la herencia.

En las arquitecturas orientadas a objetos, la composición y la herencia pueden ser muy próximas funcionalmente, pero conceptualmente son distintas. Ambas permiten la agregación de funcionalidades cuando diseñamos una clase y evitan la redundancia de código, pero tienen una diferencia fundamental: la herencia expone el funcionamiento de una clase, mientras que la agregación no lo hace (si trata con clases bien encapsuladas). Siguiendo el principio anterior, es lógico que los patrones GoF favorezcan la composición, ya que la herencia no garantiza la encapsulación. Los autores del GoF llegan a afirmar que la herencia no es compatible con la encapsulación.

Patrones GRASP:

Los patrones GRASP son principios o guías que ayudan a entender el diseño orientado a objetos y a aplicar dicho diseño de una manera metódica y racional. Ayudan en la asignación de responsabilidades de clases y objetos.

Dentro de los patrones GRASP se encuentran:

Experto:es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a la cosa real que representa, por lo que ofrece una analogía con el mundo real (Torre, 2017).

Creador: este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (Torre, 2017).

Bajo Acoplamiento: el bajo acoplamiento es un principio que se debe tener siempre en cuenta durante las decisiones de diseño. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Este patrón estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad (Torre, 2017).

Alta Cohesión: el patrón Alta Cohesión es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase de alta cohesión posee un número relativamente pequeño, con una importante funcionalidad relacionada y poco trabajo que hacer. Colabora con otros objetos para compartir el esfuerzo si la tarea es grande (Torre, 2017).

Controlador: La mayor parte de los Sistemas reciben eventos de entrada externa, los cuales generalmente incluyen una interfaz gráfica para el usuario operado por una persona. Otros medios de entrada son los mensajes externos o las señales procedentes de sensores como sucede en los sistemas de control de procesos. En todos los casos, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen

esos eventos de entrada. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La misma clase controlador debería utilizarse con todos los eventos sistémicos de un caso de uso, de modo que se pueda conservar la información referente al estado del caso (Torre, 2017).

1.6 Pruebas

Las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa. El proceso de prueba tiene dos metas distintas:

- Demostrar al desarrollador y al cliente que el software cumple con los requerimientos. Para el software personalizado, esto significa que en el documento de requerimientos debe haber, por lo menos, una prueba por cada requerimiento. Para los productos de software genérico, esto quiere decir que tiene que haber pruebas para todas las características del sistema, junto con combinaciones de dichas características que se incorporarán en la liberación del producto.
- Encontrar situaciones donde el comportamiento del software sea incorrecto, indeseable o no esté de acuerdo con su especificación. Tales situaciones son consecuencia de defectos del software. La prueba de defectos tiene la finalidad de erradicar el comportamiento indeseable del sistema, como caídas del sistema, interacciones indeseadas con otros sistemas, cálculos incorrectos y corrupción de datos (Sommerville, 2011).

1.7Conclusiones del capítulo

- ✓ El estudio de los principales conceptos relacionados con el tema deinvestigaciónpermitió una mejor comprensióndel trabajo.
- ✓ El análisis de las características de la variación de la metodología *AUP* para la UCI, permitió comprender las disciplinas a utilizar en la organización del desarrollo del módulo propuesto, en correspondencia con las características del mismo.
- ✓ El estudio de las herramientas y lenguajes posibilitóun mejor entendimiento de las tecnologías de implementación que se utilizarán para realizar el módulo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 2: DESCRIPCIÓNDELAPROPUESTADESOLUCIÓN

2.1 Introducción

En este capítulo se realiza una descripción de la propuesta de solución. A partir de la metodología definida, se obtienen en la disciplina de requisitos los requisitos funcionales y no funcionales del componente que se propone. En la disciplina de análisis y diseño se aplicanlos patrones arquitectónicos y de diseño, mientras quelos diagramas de clases del diseño, y el modelo de base de datosson artefactos resultantes de esta disciplina. Y por último los estándares de codificación que se utilizan como conjunto de reglas no formales.

2.2 Descripción de la propuesta de solución

La solución propuesta se centra en desarrollar el módulo de administración del Sistema de Gestión de Índice de Precios. El mismo comprende la gestión de usuarios, roles, permisos, trazas y nomencladores. Para la implementación de la solución se utiliza el marco de trabajo Spring Boot del lado del backend, Vue.js como el marco de trabajo del lado del frontend y GraphQL como la tecnología que garantiza la comunicación entre ambos. El módulo tiene como objetivo garantizar la correcta gestión de los usuarios, roles, permisos, trazas y nomencladores del sistema.

2.3 Disciplina de requisitos

Esta disciplina explica cómo obtener las solicitudes de los interesados y transformarlas en un conjunto de productos de trabajo de los requisitos que cubran el ámbito del sistema que va a crearse y proporcionen requisitos detallados sobre lo que el sistema debe hacer.

2.3.1 Técnicas para la identificación de requisitos

Las técnicas agrupadas como generales son las que permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle con el apoyo de técnicas más específicas. Estas técnicas son más abiertas y requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar, es importante que para sacar el mayor provecho de estas técnicas se debe tener un dialogo lo más abierto posible entre las organizaciones de desarrollo de software y las empresas cliente.Para la captura de los requisitos del módulo propuesto se utilizan las siguientes técnicas:

- **Entrevista:**Las entrevistas formales o informales con participantes del sistema son una parte de la mayoría de los procesos de ingeniería de requerimientos. En estas entrevistas, el equipo de ingeniería de requerimientos formula preguntas a los participantes sobre el sistema que actualmente usan y el sistema que se va a desarrollar. Los requerimientos se derivan de las respuestas a dichas preguntas (Sommerville, 2011).Esta técnica fue aplicada a los trabajadores de la ONEI mediante un cuestionario de preguntas realizadas para el descubrimiento de todas las necesidades que existen en dicha en la presente entidad.
- **Tormenta de ideas:** Esta técnica consiste en que varios individuos comiencen a enunciar propuestas para resolver un asunto. La técnica pretende explotar la creatividad de cada persona, fomentándola a través de las interacciones en el seno del grupo (Porto, 2016). Esta técnica se ve reflejada durante las reuniones con el cliente, donde después de discutir todo lo relacionado con las necesidades de este se llegaron a acuerdos como resultado de estas conversaciones.

2.3.2 Especificación de requisitos

La especificación de requerimientos es el proceso de escribir, en un documento de requerimientos, los requerimientos del usuario y del sistema. De manera ideal, los requerimientos del usuario y del sistema deben ser claros, sin ambigüedades, fáciles de entender, completos y consistentes. Esto en la práctica es difícil de lograr, pues los participantes interpretan los requerimientos de formas diferentes y con frecuencia en los requerimientos hay conflictos e inconsistencias inherentes.Los requerimientos funcionales para un sistema refieren lo que el sistema debe hacer.Los requerimientos no funcionales, como indica su nombre, son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios (Sommerville, 2011).

Al realizar las técnicas para la obtención de requisitos se identificaron 22 requisitos funcionales, a continuación, se presentan estos requisitos con sus descripciones.

CAPÍTULO 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

Tabla 1Requisitos Funcionales

Nº	**Nombre**	**Descripción**
1.	Listar usuarios	Permite buscar un usuario existente en el sistema, siguiendo un conjunto de filtros de búsqueda.
2.	Registrar usuario	Permite registrar un nuevo usuario en el sistema.
3.	Modificar usuario	Permite modificar un usuario existente en el sistema.
4.	Visualizar usuario	Permite visualizar los datos de un usuario seleccionado.
5.	Desactivar usuario	Permite desactivar un usuario existente en el sistema con estado activado.
6.	Activar usuario	Permite activar un usuario existente en el sistema con estado desactivado.
7.	Cambiar contraseña	Permite cambiar la contraseña de un usuario específico.
8.	Listar roles	Permite buscar un rol existente en el sistema, siguiendo un conjunto de filtros de búsqueda.
9.	Registrar rol	Permite registrar un nuevo rol en el sistema.
10.	Modificar rol	Permite modificar un rol existente en el sistema.

11.	Visualizar rol	Permite visualizar los datos de un rol existente en el sistema.
12.	Eliminar rol	Permite eliminar un rol existente en el sistema.
13.	Adicionar permisos	Permite adicionar permisos al usuario, además de los que tiene asignado por su rol.
14.	Listar nomencladores	Permite buscar un nomenclador existente en el sistema, siguiendo un conjunto de filtros de búsqueda.
15.	Registrar nomenclador	Permite registrar un nuevo nomenclador en el sistema.
16.	Modificar nomenclador	Permite modificar un nomenclador existente en el sistema.
17.	Visualizar nomenclador	Permite visualizar los datos de un nomenclador existente en el sistema
18.	Desactivar nomenclador	Permite desactivar un nomenclador existente en el sistema con estado activado.
19.	Activar nomenclador	Permite activar un nomenclador existente en el sistema con estado desactivado.
20.	Listar trazas	Permite buscar las trazas del sistema, siguiendo un conjunto de filtros de búsqueda.
21.	Autenticar usuario	Permite la autenticación en el sistema.

22.	Cerrar sesión	Permite cerrar la sesión del usuario autenticado.

En el caso de los requisitos no funcionales se definieron un total de 6, los cuales son clasificados y descritos a continuación.

Seguridad

RnF1.El sistema debe ser capaz de asociar a los usuarios autenticados los permisos especificados.

Confiabilidad

RnF2. El marco de trabajo permite cancelar las instancias de los procesos y las tareas.

RnF3. El sistema debe ser capaz de crear salvas cada un intervalo de tiempo y darle al usuario la posibilidad de restablecer, de esta forma si hay falta de fluido eléctrico se garantiza que no se pierda la información introducida por el usuario.

Usabilidad

RnF4. Para llegar a la funcionalidad debe estar a no más de 3 clics.

RnF5. El sistema provee manuales de usuario. La ayuda es sensible al contexto e incluye información sobre los flujos de trabajo.

Eficiencia

Rn6. El sistema de ser capaz de ofrecer tiempos de repuestas mínimos o iguales a 5segundos.

2.3.3 Validación de requisitos

Con el objetivo de garantizar que el componente a desarrollarse corresponde con las necesidades del cliente, cada uno de los requisitos identificados fueron validados antes de llegar a la disciplina de análisis y diseño.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 la validación de estos se utilizaron las siguientes técnicas:

- **Construcción de prototipos de interfaz de usuario:** esta técnica permite hacer simulacionesdel componente implementado y brinda la posibilidad a los especialistas de tener una idea de cómo serán lasinterfaces del componente una vez desarrollado.En la HU se muestra el prototipo no funcional correspondiente al RF 8: registrar nomenclador.

- **Revisiones formales de los requisitos:** se realizaron revisiones formales de cada requisito, por parte del cliente y el equipo de desarrollo, quedando validado que la interpretación de cada una de las descripciones no es ambigua, ni presenta omisiones o errores que hagan que la descripción del requisito no se corresponda con las necesidades del cliente.

Métrica Calidad de la especificación: para medir la calidad de la especificación de los requisitos de software se aplicó la métrica Calidad de la Especificación (CE). El empleo de esta métrica permite obtener un alto nivel de entendimiento y precisión de los requisitos, para llegar a ello, se debe primeramente calcular el total de requisitos de software como se muestra a continuación:

Nr = Nf + Nnf

Nr: total de requisitos de software.
Nf: cantidad de requisitos funcionales.
Nnf: cantidad de requisitos no funcionales.

Sustituyendo los valores en la ecuación, se obtiene:

Nr = 22 + 6

Nr = 28

Para calcular la especificidad de los requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

Q1 = Nui / Nr

Nui: número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

Para sustituir los valores de las variables en la ecuación se tuvo en cuenta que, de los requisitos especificados para el desarrollo del componente,tres de ellos causaron

contradicción en sus interpretaciones (RF 5, RF 17 y RF 19). Por tanto, la variable Q1 obtiene el siguiente valor:

Q1 = 25 / 28

Q1 = 0.89

Es importante aclarar que mientras más cerca de 1 está el valor de Q1, menor es la ambigüedad. Teniendo en cuenta el resultado anterior, igual a 0.89, se concluye que el 89% de los requisitos es entendible. Los 3 requisitos identificados como ambiguos fueron modificados y validados para garantizar su correcta comprensión, llegando al resultado ideal de Q1=1. Este resultado demuestra que el 100 % de los requisitos son fáciles de comprender.

2.3.4 Historias de Usuario

Las historias de usuario son descripciones cortas y simples de una característica contada desde la perspectiva de la persona que desea la nueva capacidad, generalmente un usuario o cliente del sistema. Por lo general, siguen una plantilla simple (Scrum, 2018):

- Como **<Usuario>**
- Quiero **<algún objetivo>**
- Para que **<motivo>**

Para el desarrollo del móduloque se propuso se definieron un total de 22 HU, una por cada RF. En estas historias de usuariose muestran todos los detalles que posibilitan una mejor comprensión. A continuación, se presenta una de las HU que se diseñó para el desarrollo de estos módulos.

Tabla 2 Historia de usuario

Historia de usuario	
Número: 1	**Requisito:** Registrar nomenclador
Programador: Rene Sánchez Linares	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo Estimado:**8h
Riesgo en Desarrollo:	**Tiempo Real:**8h

Descripción: Permite registrar un nomenclador en el sistema.

Campos:

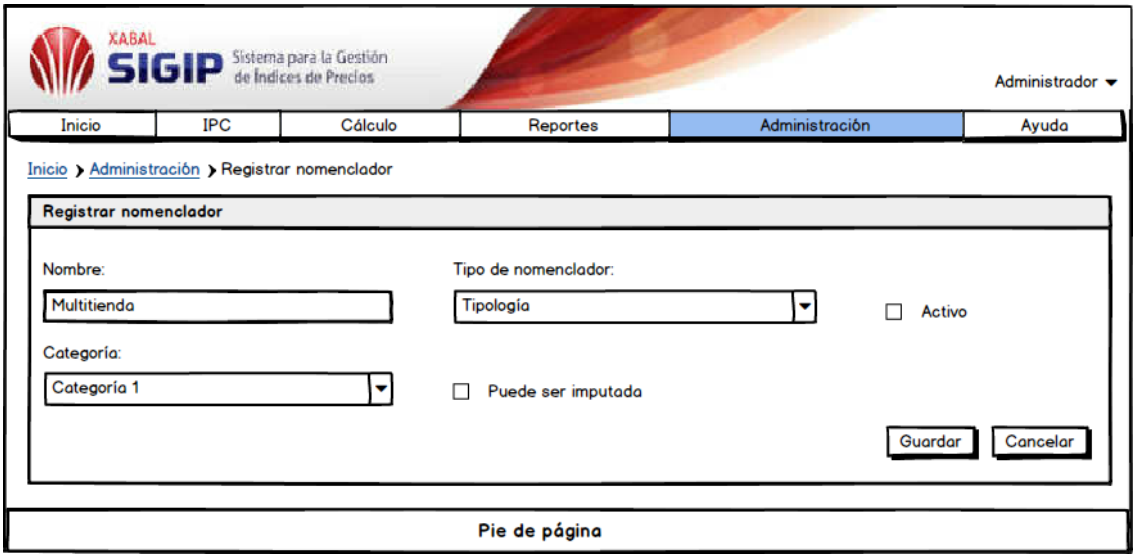
- Nombre (nombre del nuevo nomenclador que se está registrando)
- Tipo de nomenclador (permite seleccionar uno de los tipos de nomencladores definidos en el sistema: Categoría, Tipología, Observación, Incidencia y Mercado)
- Activo (permite definir si el nuevo nomenclador iniciará activado o no)
- Siglas (campo que se muestra solo cuando se seleccionan uno de los tipos de nomencladores Incidencia u observación)
- Categoría (campo que se muestra solo cuando se selecciona el tipo de nomenclador Tipología, lista todas las categorías registradas en el sistema, el campo es opcional)
- Puede ser imputada (campo que se muestra cuando se selecciona el tipo de nomenclador Tipología)

Opción:

- Guardar (permite registrar los datos del nomenclador seleccionado)
- Cancelar (regresa a la interfaz principal)

Observaciones:

Prototipo de interfaz gráfica de usuario:



2.4 Disciplina de análisis y diseño

En esta disciplina se trabajan los requerimientos para transformarlos enespecificaciones que describirán cómo debe funcionar el sistema.Losobjetivos del análisis y diseño son: transformar los requerimientos en diseño y desarrollar la arquitectura primaria del sistema (CUMP, 2018).

2.4.1 Diseño arquitectónico

El diseño arquitectónico se interesa por entender cómo debe organizarse un sistema y cómo tiene que diseñarse la estructura global de ese sistema. En el modelo del proceso de desarrollo de software, el diseño arquitectónico es la primera etapa en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos. La salida del proceso de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación (Pressman, 2010).

La solución propuesta se estructura con la API *GraphQL* delmóduloa desarrollar, dividiendo la arquitectura en dos partes esenciales *fronted* y *backend.* El *fronted* interactúa con los usuarios y el *backend* procesa las entradas desde el *frontend.*Teniendo en cuenta lo anterior, GraphQL utiliza el patrón arquitectónico Vuex para gestionar el estado de la aplicación. También se utiliza la biblioteca de *apollo-client* para tener un manejo de los datos. En la figura 1 se muestra la arquitectura definida para el*frontend*del módulo propuesto.

En el diagrama de la Figura 1 se representa la arquitectura del frontend*,*elcual se utiliza para definir los componentes visuales para los roles, trazas, permisos, usuarios y nomencladores. En la gestión del estado de la aplicación se emplea la biblioteca Vuex, la cual define una acción que lanza una mutación correspondiente a la misma y esta modifica el estado deseado, que a su vez es reflejado en los componentes visuales. Por otra parte, se utiliza la biblioteca *apollo-client*para la gestión del estado de los datos. Esta biblioteca define mutaciones, consultas y suscripciones. Además de un *store* que guarda el estado de los datos para ser utilizados en los componentes visuales.

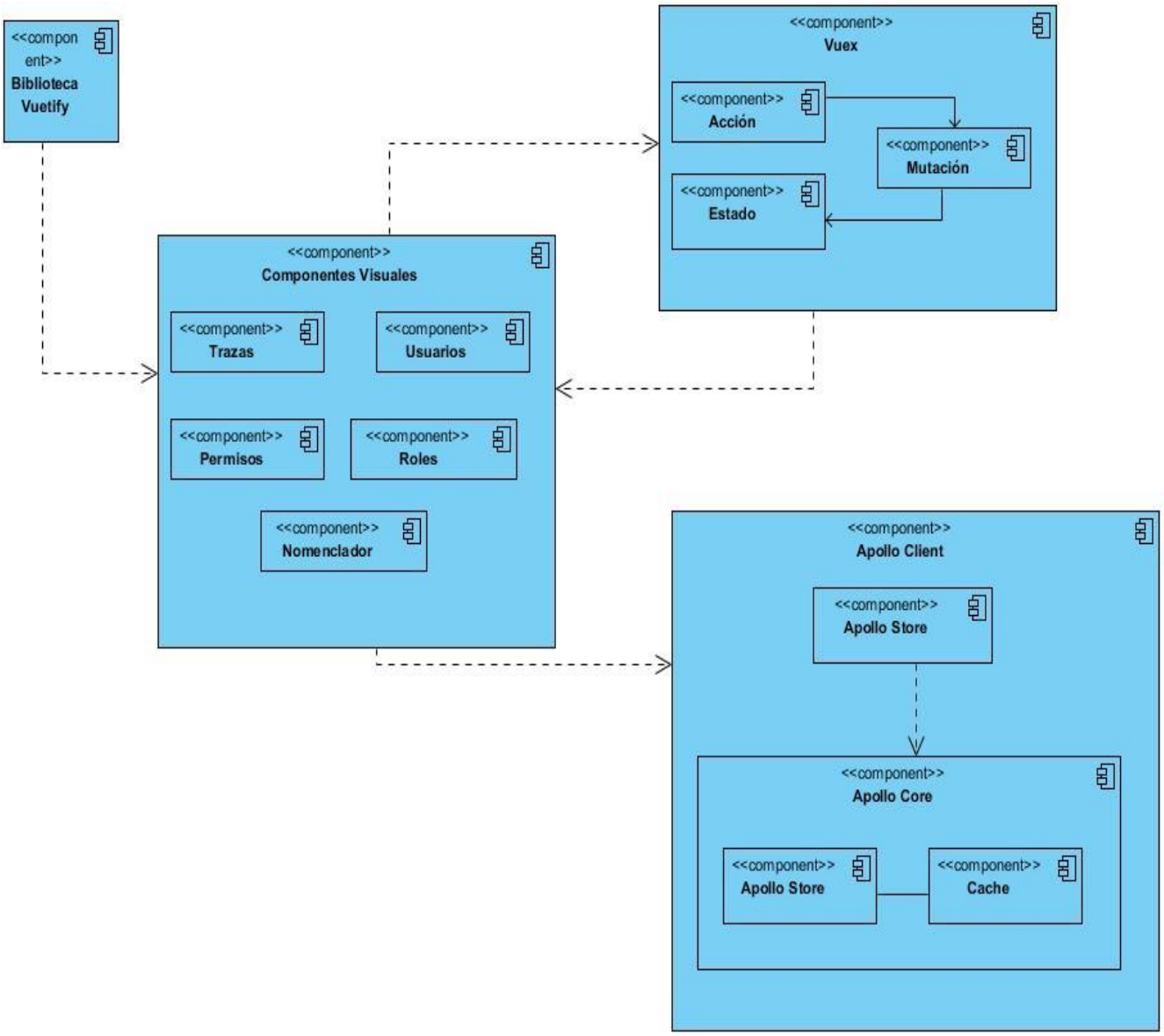


Figura 1: Arquitectura del Cliente-Graphql

Fuente:elaboración propia

Por otra parte, en la API GraphQLdel componente propuesto se aplica el patrón arquitectónico N-Capas, definiéndose las capas web, infraestructura y dominio.

En el componente propuesto se aplica el patrón arquitectónico N-Capas, definiéndose las capas web, infraestructura y dominio.

La arquitectura basada en N-Capas permite que un número ilimitado de programas corra simultáneamente, enviando información de uno a otro, utilizando diferentes protocolos para comunicarse e interactuar simultáneamente; esto permite crear aplicaciones más poderosas que proporcionen diversos servicios a varios clientes (Gonzaga, 2006).

En la Figura 2se muestrala arquitectura del backendde la capa Web que es donde se encuentra el único punto de acceso de Graphql. En la capa Dominio se encuentran las clases de persistencia, y los componentes repositorios los cuales son los encargados de la interacción con la base de datos. En la capa Infraestructura se encuentran las implementaciones de los servicios que definen las funciones necesarias para devolver los datos solicitados por el cliente GraphQL. Esta capa facilita además todas las interfaces de los servicios, las cuales son utilizadas por *GraphQLResolver.*

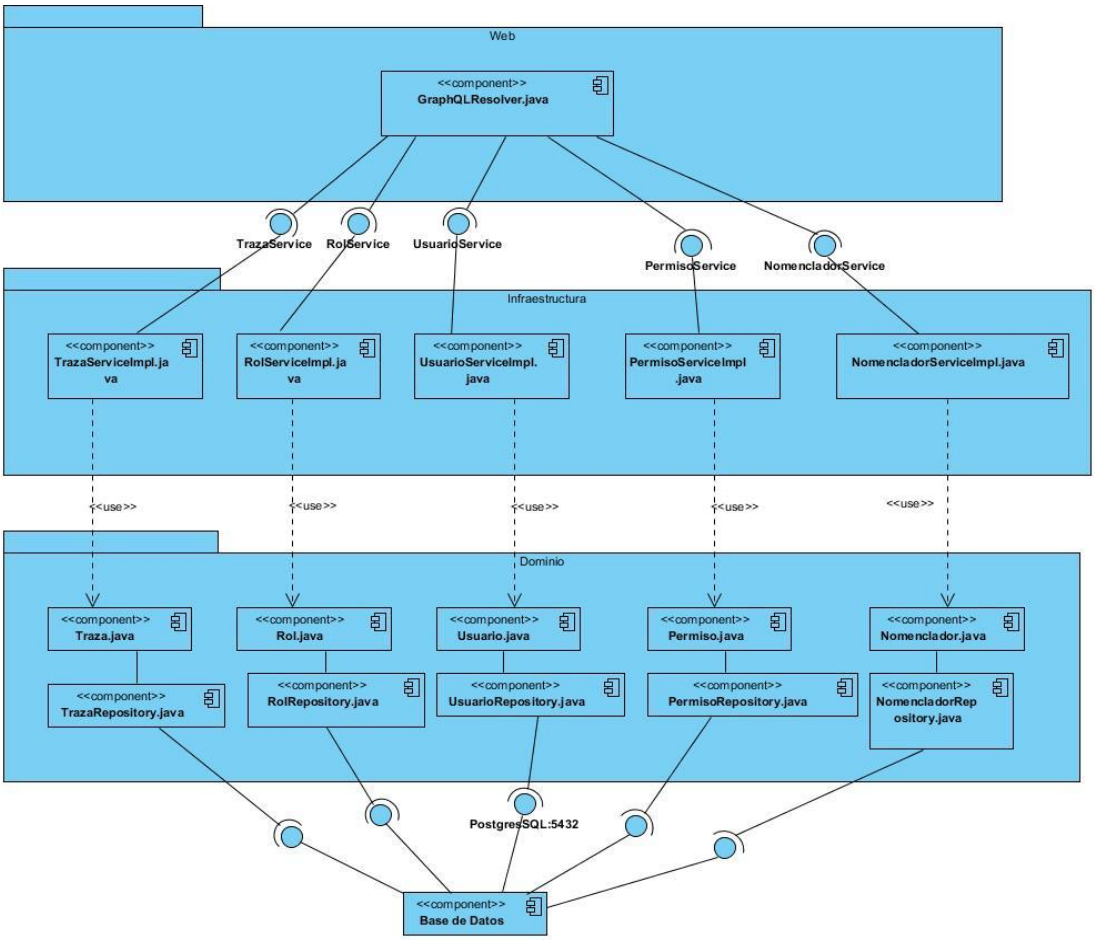


Figura 2: Arquitectura del Backend

Fuente: elaboración propia

2.4.2 Patrones de diseño

Para diseñar el componente se emplearon un conjunto de patrones de diseño, los cuales son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. A continuación, se describen los patrones empleados:

Patrones GRASP:

Experto:Es utilizado para que cada objeto realice la funcionalidad de acuerdo a la información que domina. Se evidencia en las clases *Service*,*ServiceImpl* y en los *Repository*, las cuales agrupan funcionalidades y toman decisiones sobre una entidad determinada. Por ejemplo, las clases *NomencladorService*, *NomencladorServiceImpl* y *NomencladorRepository* solo toman decisiones relacionadas con la Entidad Nomenclador.

```java
package cu.uci.cegel.onei.sigipipc.persistence;

import ...
@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "dnomenclador")
public class Nomenclador {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name = "dnomencladorId")
    private Long id;

    @NonNull
    @Column(name = "activo")
    private Boolean activo;

    @NonNull
    @Column(name = "descripcion")
    private String description;

    @NonNull
    @ManyToOne
    @JoinColumn(name = "dtiponomencladorId")
    private TipoNomenclador tiponomenclador;


}
```

Figura 3: Aplicación del patrón Experto en la clase Nomenclador

Fuente:elaboración propia

Creador: el uso de este patrón se evidencia en las clases *ServiceImpl*, que son las que tienen la responsabilidad de instanciar objetos para cumplir sus funciones. PorEjemplo, la clase *NomencladorServiceImpl*,que es donde se creanlos objetos de TipoNomencladorpara realizar las diferentes funcionalidades.

Bajo acoplamiento: Al lograr una alta cohesión en el diseño de las clases del sistema, se garantiza un bajo acoplamiento ya que permite tener clases más independientes, donde la realización de cambios sea más sencillay a su vez permitan la reutilización. Este patrón se observa en las clases *Service*, *ServiceImpl*y los *Repository.*

Alta cohesión: Se evidencia en el diseño de cada una de las clases de los módulos garantizando que solo existan en ellas las características y funcionalidades necesarias. Entre estas clases se encuentran las clases *Service, ServiceImpl* y los *Repository*.

Entre los patrones GoF se encuentran:

Instancia única (*Singleton*):Teniendo en cuenta que para el desarrollo de los módulos propuestos se utiliza el marco de trabajo SpringBoot, se asegura que todas las clases del componente mantengan una sola instancia y proporcionan un punto de acceso global a ellas durante la ejecución.

Fachada:Proporcionar una interfaz para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas. El patrón se utilizó al diseñar interfaces con funcionalidades bien definidas que van a ser implementadas por todas las clases que las necesiten. Se evidencia en las clases *Service* que son interfaces que posteriormente son implementadas, ejemplo *NomencladorService* es una interfaz que posteriormente es implementada en la clase *NomencladorServiceImpl*.

Iterador:Proporciona una forma de acceder secuencialmente a los elementos de un objeto compuesto por agregación sin necesidad de desvelar su representación interna. Se utiliza en las clases *ServiceImpl*para recorrer las estructuras de datos utilizadas. Este patrón permite garantizar una integridad en la información.

2.4.3 Diagrama de clases del diseño

Los diagramas de clases son diagramas estructurales del lenguaje unificado de modelado(UML por sus siglas en inglés de *UnifiedModelingLanguage*). Este lenguaje de modelado visual es un estándar de la Organización Internacional de Normalización o Estandarización (ISO por sus siglas en inglés de *International OrganizationforStandardization*.) ISO para visualizar los sistemas de la programación orientada a objetos, aunque también permite visualizar procesos de negocio. Con ayuda de elementos gráficos, UML muestra los estados de los sistemas y describe las interacciones entre los elementos que lo componen. Para poder hacerlo, la notación UML define las formas y las líneas para 14 tipos de diagrama(Cillero, 2009).

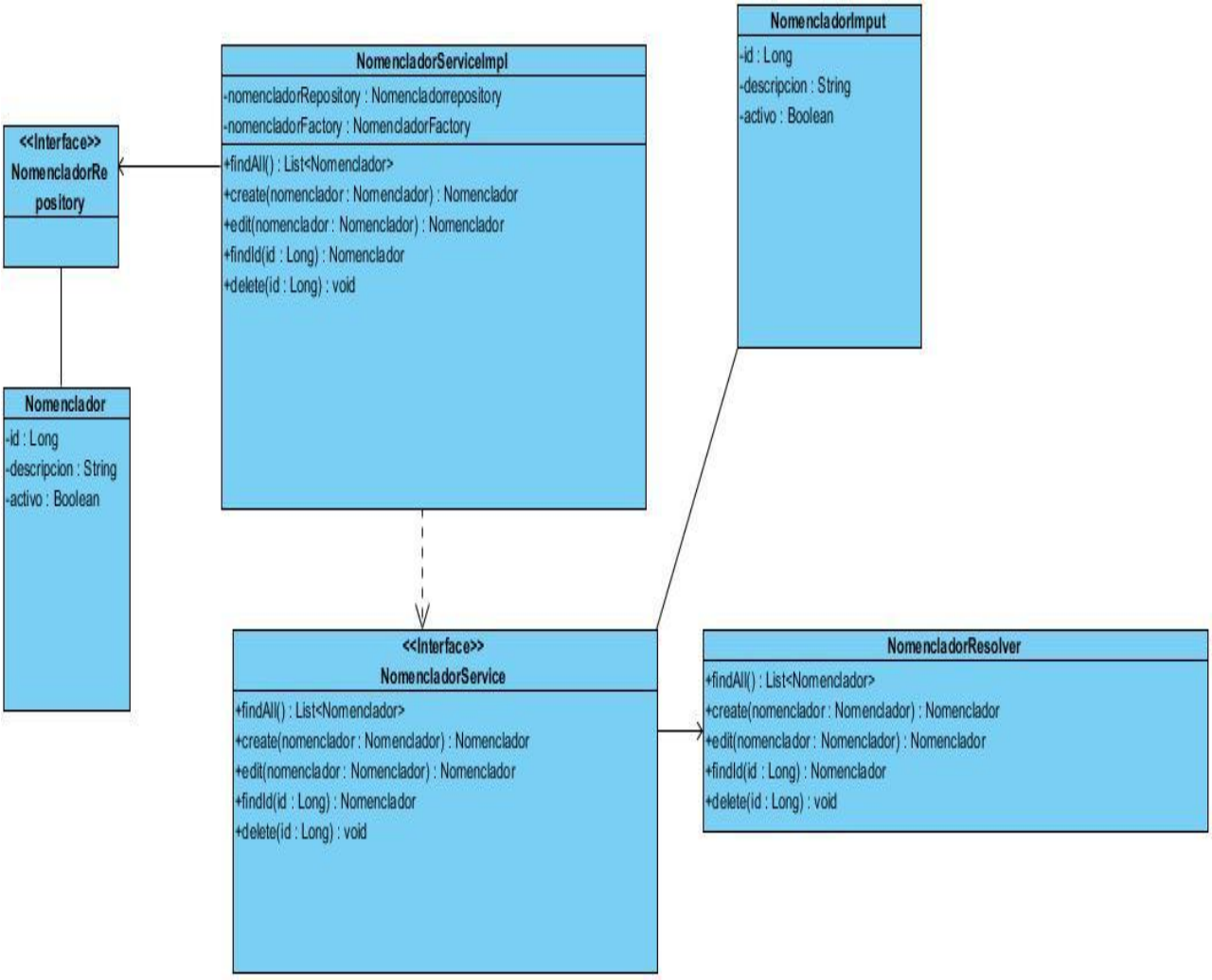


Figura 4*:* Diagrama de clases del diseño

Fuente:elaboración propia

- Clase Nomenclador: es una clase de persistencia que representa los datos que contiene un nomenclador.
- Clase *NomencladorImput*: representa los datos que interactúan con el frontend desde el Bakend.
- Interfaz Nomenclador*Service*: define todas las abstracciones necesarias para la gestión de los nomencladores.
- Interfaz *NomencladorRepository*: es la interfaz encargada del acceso a datos, la cual permite la comunicación entre la clase *NomencladorServiceImpl* de los módulos y la base de datos, la misma presenta una relación de asociación bidireccional con la clase *Nomenclador*.
- Clase *NomencladorServiceImpl*: implementa las abstracciones definidas en la clase Nomenclador*Service*y permite la gestión de los nomencladores, la misma presenta una relación de uso con la interfaz Nomenclador*Service*, además presenta una relación de asociación unidireccional con la interfaz *NomencladorResolver*.
- *NomencladorResolver.: es donde se representan las query y las mutacion de GraphQL relacionadas con la entidad.*

2.4.4 Modelo de base de datos

La Figura 3 semuestra el modelo de la base de datos relacional PostgreSQL del módulo propuesto. Este presenta cada una de las tablas, así como sus atributos y relaciones, las cuales son de mucho a mucho, representados en un diagrama Entidad-Relación. El modelo de datos representado en la figura 3 cuenta con un total de 9 tablas, dentro del cual se gestionan tanto los elementos del negocio como los de la arquitectura. Este diagrama también es el encargado de agrupar un conjunto de entidades con atributos en común. Las tablas del modelo son las encargadas de gestionar conceptos específicos de los módulos en general, por ejemplo, en la tabla dtrazas es donde se guarda el id de dtrazas, el ip, nombreusuario, fecha y la descripción.

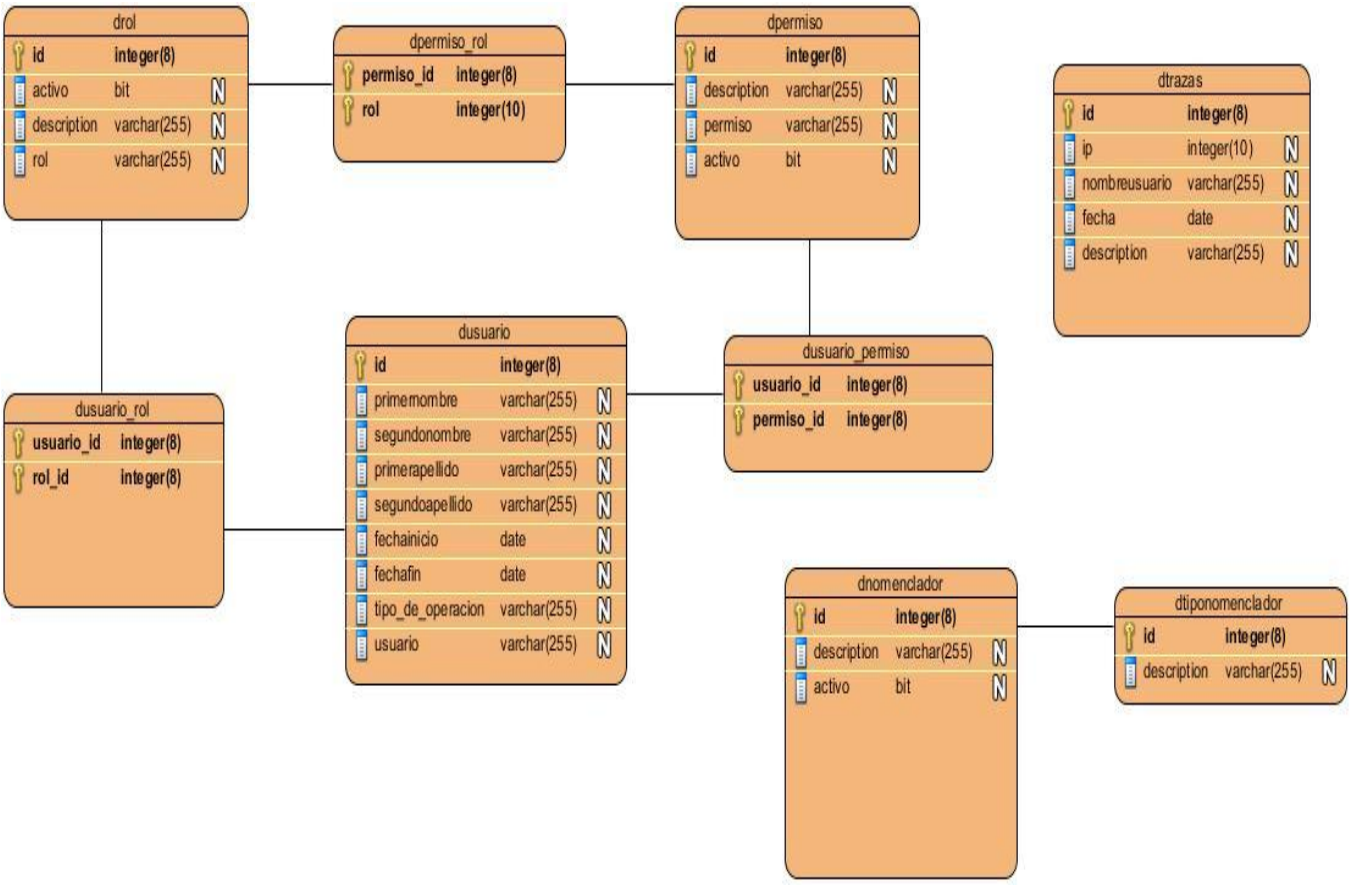


FIGURA 6: Modelo de datos de la solución propuesta
Fuente:elaboración propia

2.5 Disciplina de implementación

En esta disciplina a partir de los resultados del Análisis y el diseño se construye la solución que da lugar a la presente investigación, para la cual se aplicaron un conjunto de estándares de codificación definidos por el equipo de desarrollo del SIGIP.

2.5.1Estándares de codificación

- Los nombres de las clases serán con mayúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán de igual forma, ejemplo: *PermisoService*.
- Los nombres de los métodos serán con minúscula, en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula, ejemplo: totalNomencladores.

- Los identificadores para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto las siguientes palabras se escribirán con mayúscula, ejemplo: trazas y *trazasDTO*.
- Los nombres de variables o funciones deben ser lo suficientemente descriptivos, sin exceder de 30 caracteres.
- El nombre de la clase controladora terminará con la palabra *Controller*, ejemplo: *GraphQLResolver*.
- Las interfaces de repositorio comenzarán nombrándose por el nombre de la entidad y seguido la palabra *Repository*, ejemplo: Trazas*Repository*.
- Las interfaces de los servicios comenzarán nombrándose por el nombre de la entidad y seguido la palabra *Service*, ejemplo: *TrazasService*.
- Las implementaciones de las interfaces de servicios comenzarán nombrándose por el nombre de la entidad y seguido la palabra *ServiceImpl*, ejemplo: *TrazasServiceImpl* y *NomencladorServiceImpl.*

2.6 Conclusiones

- ✓ El empleo de la metodología *AUP* en su variación para la UCI en función de describir elproceso de desarrollo del módulo propuesto, permitió organizar el desarrollo de la solución y generar los artefactos necesarios.
- ✓ La aplicación de técnicas para la obtención de requisitos permitió la obtención de los requisitos funcionales y no funcionales.
- ✓ La aplicación del patrón arquitectónico Vuex, para el manejo del estado de la aplicación, junto a la biblioteca *apollo-client*utilizada para el manejo de los datos en la arquitectura del clienteGraphQLy el patrón arquitectónico N-Capas permitió que el módulo propuesto sea flexible al mantenimiento y a la introducción de cambios.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En este capítulo se muestran los resultados obtenidos en la validación del diseño de la solución aplicando las métricas Tamaño Operacional de Clases y Relaciones entre Clases. El capítulo concluye con la evaluación de la relación causa-efecto de la variable independiente sobre las variables dependientes de la investigación, a partir de la definición de un conjunto de indicadores que permiten chequear el comportamiento de esta relación antes y después del desarrollo del módulo propuesto.

3.2Validación del diseño

Para poder verificar la calidad del diseño propuesto se emplearon las métricas de diseño relaciones entre clases (RC) y tamaño operacional de clases (TOC).

3.2.1 Métrica Tamaño Operacional de Clase (TOC)

A cada una de las clases del diseño se le aplico la métrica TOC con el objetivo de medir la calidad con respecto a la de su responsabilidad, complejidad de implementación y reutilización de los datos. Se describe a continuación los pasos seguidos al aplicar esta métrica

- Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee el diseño.
- Calcular el promedio de los umbrales.

En la siguiente tabla se muestran los datos a utilizar para la aplicación de la métrica TOC sobre el diseño de clases de los módulos propuestos.

Tabla 3 Métricas TOC.

Atributos de Calidad	**Clasificación**	**Criterio**
Responsabilidad	Baja	Umbral<= Promedio
	Media	Promedio < Umbral < = 2* Promedio
	Alta	Umbral > 2* promedio
Complejidad de Implementación	Baja	Umbral <= Promedio
	Media	Promedio < Umbral < = 2*

		Promedio
	Alta	Umbral > 2* promedio
Reutilización	Baja	Umbral > 2* promedio
	Media	Promedio < Umbral < = 2* Promedio
	Alta	Umbral <= Promedio

Fuente:(Lorenz, 1994)

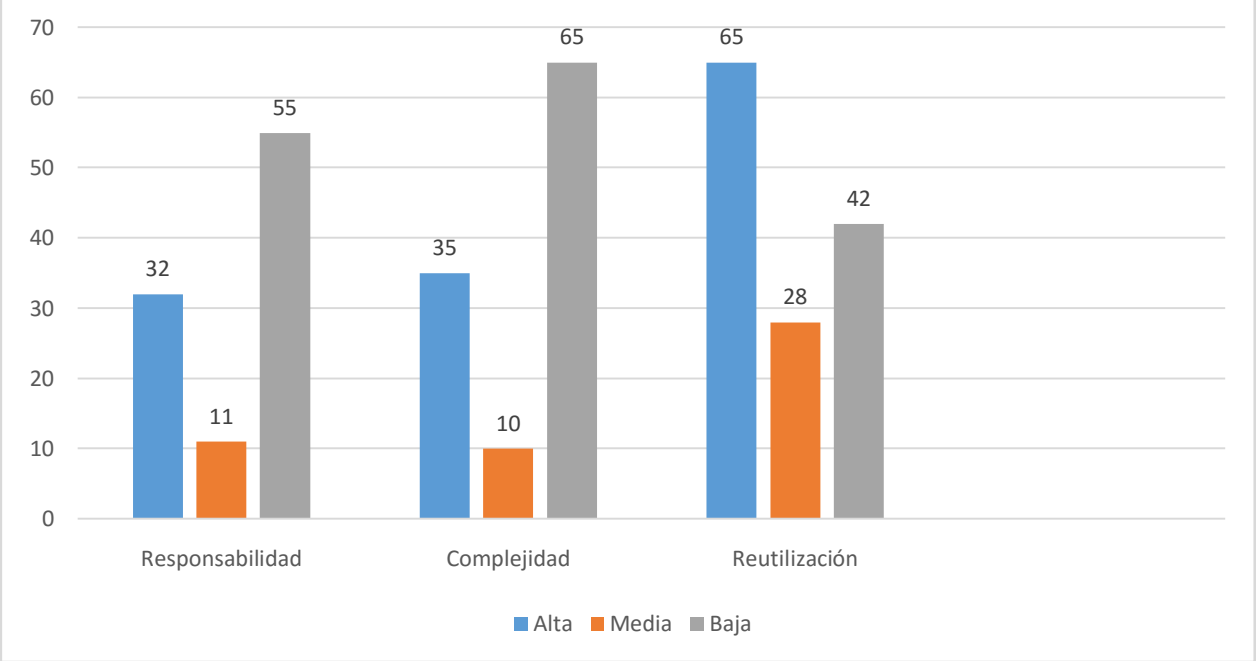


Figura 5 Representación en (%) de los resultados de la aplicación de la métrica TOC.

Fuente: *elaboración propia.*

La figura anterior demuestra que con la aplicación de la métrica TOC se obtuvieron resultados positivos en relación a los siguientes atributos evaluados:

Responsabilidad: los resultados fueron satisfactorios, teniendo en cuenta que el 55 % de las clases tienen una responsabilidad Baja.

Complejidad de implementación: los resultados fueron satisfactorios, pues se demostró que el 65 % de las clases tienen una complejidad baja.

Reutilización: los resultados obtenidos fueron satisfactorios, demostrándose que el 65% de las clases tienen una reutilización alta.

3.2.2. Métrica Relaciones entre clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. El primer paso en la aplicación de esta es evaluar los siguientes atributos de calidad: acoplamiento, complejidad de mantenimiento, reutilización de cada clase y la cantidad de pruebas que cada clase requiere (Pressman, 2010).
Se exponen los pasos que se llevaron a cabo para aplicar la métrica a todas las clases del componente propuesto en la presente investigación:

- Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.
- Calcular el promedio de las CRU.

En la siguiente tabla se muestran los datos a utilizar para la aplicación de la métrica RC sobre el diseño de clases de los módulos propuestos.

Tabla 4: Rango de valores para medir la afectación de los atributos de calidad (RC).

Atributos de Calidad	**Clasificación**		**Criterio**
Acoplamiento	Ninguna		CRU=0
	Baja		CRU=1
	Media		CRU=2
	Alta		CRU>2
Complejidad de Mantenimiento	Baja		CRU <= Promedio
	Media		Promedio < CRU <= 2* Promedio
	Alta		CRU > 2* promedio
Reutilización	Baja		CRU > 2* promedio
	Media		Promedio < CRU <= 2* Promedio
	Alta		CRU <= Promedio
Cantidad de pruebas	Baja		CRU <= Promedio
	Media		Promedio < CRU <= 2*

			Promedio
	Alta		CRU > 2* promedio

Fuente:(Lorenz, 1994)

En la figura 6 se muestran los resultados obtenidos luego de aplicar la métrica RC sobre el diseño de clases del componente propuesto.

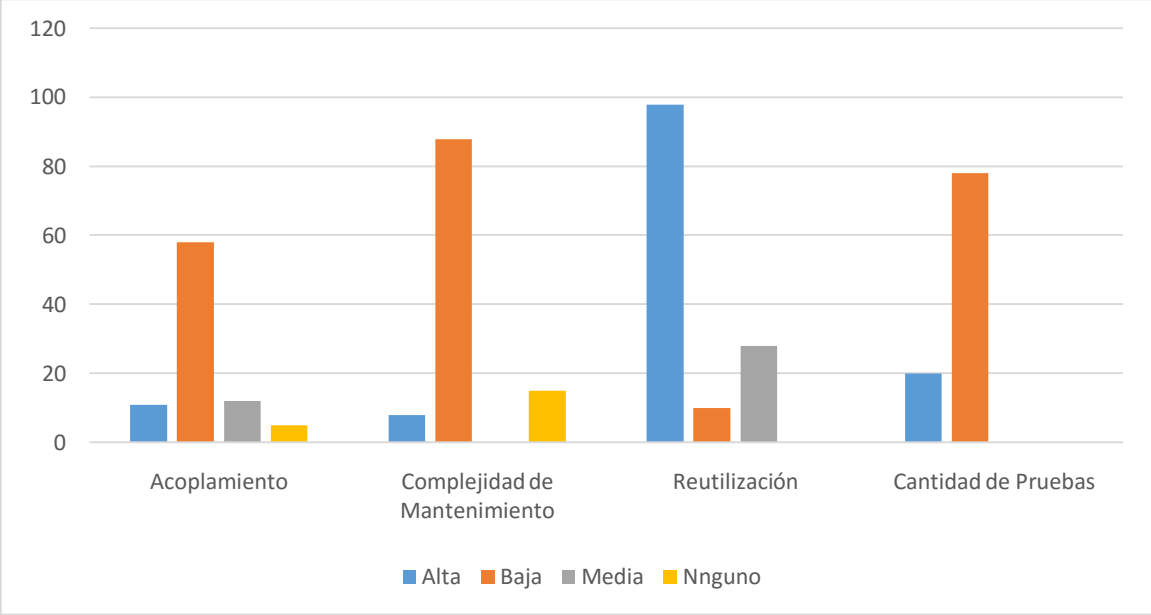


Figura 6Representación en (%) de los resultados de la aplicación de la métrica RC.

Fuente: *elaboración propia.*

La figura anterior demuestra que con la aplicación de la métrica RC se obtuvieron resultados positivos en relación a los siguientes atributos evaluados:

Acoplamiento: los resultados obtenidos son positivos teniendo en cuenta que el 58 % de las clases no poseen acoplamiento.

Complejidad de mantenimiento: los resultados mostrados en la figura anterior, demuestran que el 88 % de las clases del módulo propuesto presentan una complejidad de mantenimiento baja, facilitando así las futuras actividades de soporte sobre el mismo.

Reutilización:los resultados obtenidos fueron satisfactorios, demostrándose que el 98% de las clases tienen una reutilización alta.

Cantidad de pruebas: los resultados demuestran que el 78 % de las clases poseen un bajo grado de esfuerzo a la hora de realizar cambios, rectificaciones o pruebas sobre ellas.

Con los resultados obtenidos una vez aplicada la métrica RC se concluye que el diseño de los módulos propuestos tiene una baja complejidad de mantenimiento y acoplamiento entre sus clases. Además, se requiere de un bajo grado de esfuerzos para realizar cambios sobre la mayoría de las clases y estás a su vez presentan un elevado por ciento de reutilización. Todo lo anterior facilita la obtención de una solución informática escalable, con poca dependencia entre clases, flexible al mantenimiento y a la introducción de cambios.

3.3 Pruebas de Software

Las pruebas de software son un conjunto de actividades que pueden ser planificadas con antelación y ejecutarse sistemáticamente durante la implementación o al finalizar el desarrollo del software. Estas comprenden un conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación. Las pruebas de software se ejecutan a partir de la aplicación de métodos y técnicas. La organización del proceso de pruebas se realiza a través de cuatro niveles: unidad, integración, sistema y aceptación (Pressman, 2010).

Para la validación del módulo propuesto en la presente investigación se define una estrategia de prueba de software a partir de las disciplinas establecidas por la metodología AUP variación UCI. En la estrategia se define que en las disciplinas de pruebas internas y pruebas de liberaciónsolo se valide a nivel de unidad y en la disciplina de pruebas de aceptación se utiliza el nivel de igual nombre, en todos los casos con la aplicación de los respectivos métodos y técnicas. Los niveles de integración y sistema no se aplican, teniendo en cuenta que el alcance de la tesis solo comprende el desarrollo del componente,sin llegar a su integración con alguna aplicación.

3.3.1 Pruebas internas

A continuación, se detalla cómo el equipo de desarrollo ejecutó las pruebas internas a la solución propuesta, organizada en el nivel de unidad, aplicando los correspondientes métodos y tipos de pruebas.

Pruebas a nivel de unidad:

Las pruebas a nivel de unidad se concentran en el esfuerzo de verificación de la unidad más pequeña del diseño, el componente o módulo de software. Tomando como guía la descripción del diseño a nivel de componente, se prueban importantes caminos de control para describir errores dentro de los límites del módulo. El alcance restringido que se ha

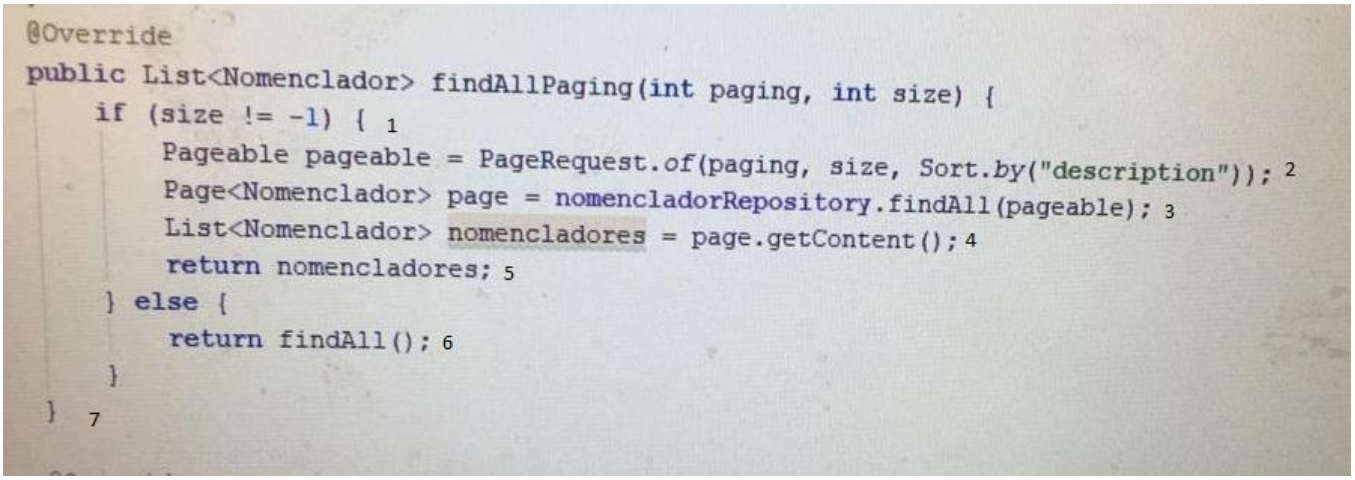
determinado para las pruebas de unidad limita la relativa complejidad de las pruebas y los errores que éstas descubren (Pressman, 2010). En el caso de la presente investigación en este nivel de prueba se decide aplicar los métodos de caja blanca y caja negra.

Métodos de caja blanca:

El método de caja blanca posibilita el desarrollo de casos de prueba que garanticen la ejecución, al menos una vez, de los caminos independientes (Pressman, 2010). En el caso del presente trabajo de diploma el método fue ejecutado aplicando la técnica de ruta básica.

La técnica de ruta básica tiene como objetivo comprobar que cada camino se ejecute independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta técnica debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada camino independiente sea ejecutado por lo menos una vez en el sistema (Pressman, 2010).

En la validación del componente propuesto la técnica de ruta básica se aplicó a todos los métodos de las clases controladoras, teniendo en cuenta que estas agrupan las principales funcionalidades de la solución. A continuación, se describe la aplicación del método de caja blanca sobre la funcionalidad *ListarNomenclador*, perteneciente a la clase *NomencladorServiceImpl* (ver Figura). Se toma como muestra esta funcionalidad teniendo en cuenta que es una de las de mayor prioridad yrespondea uno de los principales requisitos funcionales de la solución.



```
@Override
public List<Nomenclador> findAllPaging(int paging, int size) {
    if (size != -1) { 1
        Pageable pageable = PageRequest.of(paging, size, Sort.by("description")); 2
        Page<Nomenclador> page = nomencladorRepository.findAll(pageable); 3
        List<Nomenclador> nomencladores = page.getContent(); 4
        return nomencladores; 5
    } else {
        return findAll(); 6
    }
} 7
```

Figura 7*:* Funcionalidad ListarNomenclador

***Fuente*:** elaboración propia

Una vez definido el código sobre el cual se aplica el método, los pasos a seguir para desarrollar la técnica de ruta básica son los siguientes:

1) Confeccionar el grafo de flujo, este muestra el flujo de control lógico (Pressman, 2010). Está compuesto por los siguientes elementos:

- Nodos: son círculos que representan una o más sentencias procedimentales.
- Aristas: son flechas que representan el flujo de control y son análogas a las flechas deldiagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

En la figura 8 se muestra el grafo de flujo obtenido con la ejecución del método de caja blanca sobre la funcionalidad *ListarNomenclador*.

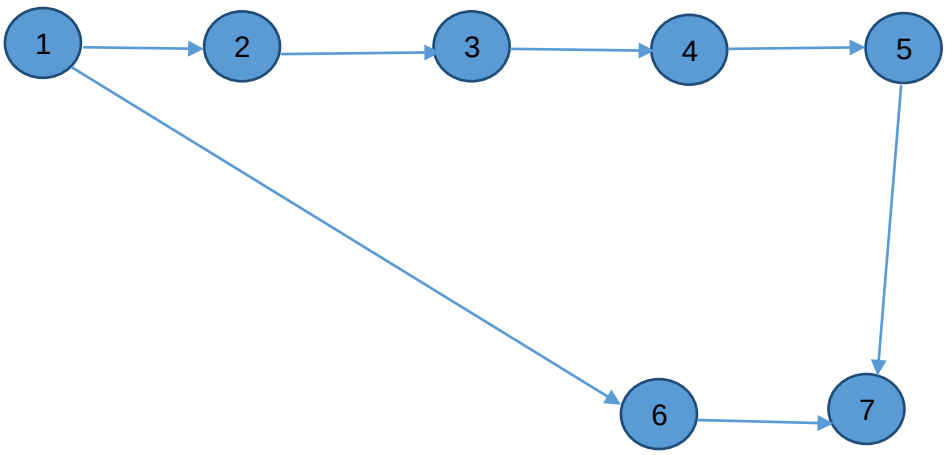


Figura 8: Grafo de camino básico Del método ListarNomenclador.

Fuente: elaboración propia.

2) Calcular la complejidad ciclomática:

El valor calculadopor la complejidad ciclomática define el número de rutas independientes del conjuntobásico de un programa y brinda una cota superior para el número de pruebas que se deben realizar a fin de asegurar que todos los enunciados se ejecutaron al menos unavez(Pressman, 2010).

La complejidad ciclomática se calcula de tres formas diferentes, lascuales deben llegar al mismo resultado para comprobar que el cálculo es elcorrecto (Pressman, 2010).

- El número de regiones del grafo de flujo corresponde a la complejidadciclomática.
- La complejidad ciclomática V(G) para un grafo de flujo G se define como:
  V(G)=E-N+2
  Donde E es el número de aristas del gráfico de flujo y N el número de nodos delgráfico de flujo.
- La complejidad ciclomática V(G) para un grafo de flujo G también se definecomo
  V(G)=P+1
  Donde P es el número de nodos predicado (nodos de donde parten al menos dos aristas) contenidos en el grafo de flujo G.

En el grafo de flujo de la figura 8, la complejidad ciclomática puede calcularse usando cada una de las vías anteriormente descritas:

1. El grafo de flujo tiene 2 regiones, por tanto, V(G) = 2
2. V(G) = (7 aristas - 7nodos) +2 = 2
3. V(G) = 1 nodos predicado + 1= 2

Por tanto, la complejidad ciclomática del grafo de flujo de la figura 8 es dos.

3) Determinar un conjunto básico de caminos linealmente independientes:

El valor de V(G) proporciona la cota superior sobre el número de rutas linealmenteindependientes a través de la estructura de control del programa (Pressman, 2010). En el caso de la funcionalidad *ListarNomenclador*, se definen 2 caminos básicos:

Camino básico #1: 1, 2, 3, 4, 5, 7
Camino básico #2: 1, 6,7
4) Obtención de casos de prueba (CP):

Una vez definidos los caminos, se procede a diseñar los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación, en las tablas yse presentanlos casos de prueba definidospara los caminos obtenidos con la técnica ruta básica.

Tabla 5 : Caso de prueba # 1.

Caso de prueba para la ruta básica # 1	
Descripción: el método recibe como parámetro una cadena de texto, un número de página, un tamaño. En caso de que sea el tamaño distinto de -1 devuelve una lista de nomencladores organizada o no por página	
Entrada:	Una cadena de texto, número de página, un tamaño
Resultados esperados	Una lista organizada por páginas o sin organizar.
Condiciones	

Fuente: elaboración propia.

Tabla 6 : Caso de prueba # 2.

Caso de prueba para la ruta básica # 2
Descripción: el método recibe como parámetro una cadena de texto, un número de página, un tamaño. En caso de que sea el tamaño distinto de -1 devuelve una lista de

nomencladores organizada por descripción pasada por parámetro	
Entrada:	Una cadena de texto, numero de página, un tamaño
Resultados esperados	Una lista organizada por descripción.
Condiciones	

Fuente: elaboración propia.

Una vez ejecutados todos los casos de pruebas obtenidos con la técnica empleada, se concluye que los mismos fueron probados satisfactoriamente, corrigiéndose los hallazgos surgidos en una primera iteración y comprobándose su corrección en una segunda. Al concluir la prueba sedemuestra que todas las funcionalidades de los módulos propuestosse ejecutan satisfactoriamente, quedando libres de código repetido o innecesario.

3.4 Conclusiones parciales

- ✓ La aplicación de técnicas para la validación del diseño certifica la obtención de módulos flexibles al mantenimiento y a la introducción de cambios.
- ✓ La realización de pruebas internas en el nivel de unidad aplicando el método de caja blanca, corrobora la ausencia de código repetido o sentencias innecesarias. .

CONCLUSIONES GENERALES

- ✓ El estudio de los referentes teóricos relacionados a las tecnologíaspermitió a la presente investigación adoptar la decisión de que la creación de unmódulo era muy necesaria.
- ✓ El desarrollo de pruebas de software en los niveles de unidad, organizadas por las disciplinas que estable la metodología AUP variación UCI para la etapa de pruebas, así como los resultados arrojados por las métricas de validación de diseño de software empleadas, corroboran la validez del diseño e implementación del módulo propuesto.
- ✓ El desarrollo de pruebas de internas y de liberación en los niveles de unidad, integración y sistema, corroboran la validez del diseño e implementación del módulo propuesto.
- ✓ La realización de pruebas internas en el nivel de unidad aplicando el método de caja blanca, corrobora la ausencia de código repetido o sentencias innecesarias.

RECOMENDACIONES

RECOMENDACIONES

- ✓ Integrar el módulo desarrollos al SIGIP.
- ✓ Extender el uso de este módulo en otros de los subsistemas pendientes a desarrollar para el SIGIP.

BIBLIOGRAFÍA

3schools.com. 2019. 3schools.com. [En línea] 2019. https://www.w3schools.com/bootstrap4/.

Aguilera González, Sergio Orlando. 2019.*Diseño de casos de prueba. Componente documento .* La Habana : Universidad de las Ciencias Informáticas, 2019.

—. 2018.*Historias de usuario. Componentes documentos y plantillas para el XEJEL.* La Habana : Universidad de las Ciencias Informáticas, 2018.

Aleaga, Yaiset Moreno. 2018.*CEGEL-SITPC-Estándares de Codificación para PHP.* 2018.

Álvarez, Daniel José Salas. 2010. Estándares de codoficacion Java. [En línea] 2010. http://www.aves.edu.co/ovaunicor/recursos/view/265.

Álvarez, Miguel Angel. 2014. desarrolloweb.com. [En línea] 2014. https://desarrolloweb.com/articulos/que-es-mvc.html.

Amoedo, Damián. 2017. [En línea] 2017. https://ubunlog.com/intellij-idea-ide-java/.

Angular. 2018. Angular.io. [En línea] 2018. https://angular.io/guide/architecture.

ArturoJS. 2018. tutorialesenpdf. [En línea] 2018. https://tutorialesenpdf.com/vue-js/.

Bembibre, Victoria. 2009. Definición ABC. [En línea] 2009. https://www.definicionabc.com/tecnologia/usuario.php.

Berciano Alonso, Alonso. 1999. [En línea] 1999. http://platea.pntic.mec.es/~abercian/guiahtml/comienzo.htm.

Boot, Spring. 2019. IONOS. [En línea] 2019. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/spring-boot-tutorial/.

Castillo, Christian Xavier Ocampo. 2018. [En línea] 2018. https://www.studocu.com/gt/document/universidad-nacional-de-loja/procesos-de-

software/resumenes/metodologias-para-el-desarrollo-de-software-tradicionales-y-agiles/4252692/view.

—. 2018. StuDocu. [En línea] 2018. https://www.studocu.com/ec/document/universidad-nacional-de-loja/procesos-de-software/resumenes/metodologias-para-el-desarrollo-de-software-tradicionales-y-agiles/4252692/view.

Castro Morell, Daniel E. y González Guadarrama, José R. 2008.*Sistema para la tramitación de procesos penales.* 2008.

Cavenago, Almeira AS. Pérez. 2007.*Arquitectura de software: Estilos y Patrones.* Argentina : Universidad Nacional de la Patagonia San Juan Bosco, 2007.

Cillero, Manuel. 2009. [En línea] 2009. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-clases/.

—. 2017. Diagrama de componentes. [En línea] 2017. https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/.

Crespo, Álvaro González. 2016. Qué es GitLab. [En línea] 2016. [Citado el: 19 de enero de 2019.] https://es.linkedin.com/learning/gitlab-esencial/que-es-gitlab.

CUMP, ROMINA CUBA. 2018. [En línea] 2018. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjVjOvP6aDqAhXjT98KHUyOAD04ChAWMAZ6BAgIEAE&url=http%3A%2F%2Frepositorio.upch.edu.pe%2Fbitstream%2Fhandle%2Fupch%2F4393%2FAnalisis_CubaCumpa_Romina.pdf%3Fsequence%3D1%2.

DataPrinx. Introducción al diseño de bases de datos. [En línea] http://www.dataprix.com/1-introduccion-diseno-bases-datos.

del Castillo San Félix, Alvaro. 2000. El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. [En línea] 2000. http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.

Doctrine-project.org. 2018. Doctrine project. What is Doctrine? [En línea] 2018. [Citado el: 14 de Noviembre de 2018.] http://docs.doctrine-project.org/en/latest/tutorials/getting-started.html.

Franco Espiño, Beatriz y Pérez Alcázar, Ricard. 2015. Redrta.org. *Redrta.org.* [En línea] 2015. http://mgd.redrta.org/directrices-administracion-de-documentos-electronicos/mgd/2015-01-27/161143.html.

Gamma, Erich, y otros. 1994.*"Design Patterns: Elements of Reusable Object Oriented Sofware".* s.l. : Grady Booch, 1994.

Gestionar. 2016. Significados. [En línea] 2016. https://www.significados.com/gestionar/.

Gonzaga, Elizabeth Acosta. 2006. Arquitecturas en n-Capas: Un Sistema Adaptivo. 2006.

González Ochoa, Darián y Domínguez González, Yosviel. 2018.*SITPC, una herramienta informática cubana para la administración de la justicia.* La Habana : s.n., 2018.

Guerra, Enrique Fernandez. 2016. Desarrolloweb.com. [En línea] 2 de junio de 2016. [Citado el: 30 de noviembre de 2018.] https://desarrolloweb.com/articulos/introduccion-a-typescript.html.

Guevara, Ing.Jymmy. 2018. Análisis y deseño detallado de aplicaciones informáticas de gestión. [En línea] Google Sites, 2018.

Ibarra, Antonio Aliaga. 2008. iessanvicente. [En línea] 2008. https://iessanvicente.com/colaboraciones/postgreSQL.pdf.

Illescas, Eduardo Díaz-Meco. 2016. Tc-abodagos.es. [En línea] 15 de junio de 2016. [Citado el: 30 de noviembre de 2018.] https://tc-abogados.es/que-es-un-documento-electronico/.

intellij-idea. riptutorial.com. [En línea]

ISO/IEC. 2005.*INGENIERÍA DE SOFTWARE - CALIDAD DEL PRODUCTO. PARTE 1: MODELO DE LA CALIDAD.* La Habana : Cuban National Bureau of Standards, 2005.

IT-CGAE. 2013.*Manuel de usuario. LEXNET.* s.l. : Red Abogacía Española, 2013. pág. 25.

java, conoce. 2019. seas. [En línea] 2019. https://www.seas.es/blog/informatica/conoce-el-lenguaje-de-programacion-java/.

javascript. 2006-2020. uniwebsidad. [En línea] 2006-2020.

Jiménez Castela, Pedro . 2017.*Angular 4 desde Cero.* 2017.

Juiz, O. 2009.*INFORME DEL SISTEMA SISECO.* 2009.

Kezmo. 2017. [En línea] 2017. https://blog.kezmo.com/qu%C3%A9-son-las-metodolog%C3%ADas-%C3%A1giles-y-por-qu%C3%A9-debes-implementarlas-en-tu-organizaci%C3%B3n-484a510e5b0?gi=bdb2a3db8b1b.

Larman, Craig. 2016.*UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 3ra . s.l. : Prentice-Hall, 2016.

LLamas, Luis. 2019. luisllamas.es. [En línea] 2019. https://www.luisllamas.es/vuetify-estetica-material-design-para-tus-apps-en-vuejs/.

Llobet, Vicente Javier González. 2019. tecnoxperiencia. [En línea] 2019. https://tecnoxperiencia.com/vuetify-vue-aplicaciones-profesionales/.

—. 2019. tecnoxperiencia. [En línea] 2019. https://tecnoxperiencia.com/vuetify-vue-aplicaciones-profesionales/.

Lorenz, M. y Kidd, J. 1994.*Object-Oriented Software Metrics.* s.l. : Prentice-Hall, 1994.

Lorenz, M., & Kidd, J. 1994.*Object-oriented software metrics: a practical guide.* New Jersey, Prentice-Hall : s.n., 1994.

Lucid Software Inc. 2018. Lucidchart. [En línea] 2018. https://www.lucidchart.com/pages/es/qu%C3%A9-es-la-notaci%C3%B3n-de-modelado-de-procesos-de-negocio.

MJU, Ministerio de Justiica de España. 2012.*EXPEDIENTE JUDICIAL ELECTRÓNICO Informes de Modernización Judicial en España.* España : Ministerio de Justicia, 2012.

Muñoz Serafin, Miguel. 2018.*Introducción al desarrollo de aplicaciones N-Capas con tecnologías Microsoft.* 2018.

Murua, Juan, Fernández, Alejandro y Eguiluz, Javier. 2019.*Pro Git, el libro oficial de Git.* 2019.

oblancarte. 2019. [En línea] 2019. https://www.oscarblancarteblog.com/2018/07/17/spring-boot-relacion-los-microservicios/.

Ochando, Blázquez. 2014. Modelo entidad-relación. Fundamentos y Diseño de Bases de Datos. [En línea] 2014. http://ccdoc-basesdedatos.blogspot.com/2013/02/modelo-entidad-relacion-er.html.

ohmyroot. 2017. Estandares de codificación. [En línea] 12 de enero de 2017. https://www.ohmyroot.com/buenas-practicas-legibilidad-del-codigo/.

Oracle. 2018. Netbeans.org. *NetBeans IDE Features.* [En línea] 2018. [Citado el: 20 de Febrero de 2019.] https://netbeans.org/features/index.html.

Pablo, José. 2017. ¿Qué es y para qué sirve un modelo conceptual? [En línea] 2017. http://www2.chj.gob.es/albufera/01_WEB_ED/01_AV_DSAV/04_GA/01_MC/9-Metodologia/9-4_Fase_3.htm.

Pacheco, Nacho. 2013. Documentación de Symfony. [En línea] 2013. http://gitnacho.github.io/symfony-docs-es/.

Pérez Porto, Julián y Gardey, Ana Gardey. 2018. Definición.de. *Definición.de.* [En línea] 2018. https://definicion.de/estandarizacion/.

Pérez Valdés, Damián. 2007. maestrosdelweb. *maestrosdelweb.* [En línea] 3 de julio de 2007. http://www.maestrosdelweb.com/que-es-javascript/.

Porto, Julián Pérez. 2016. Definición de Tormentas de Ideas. 2016.

postgressql. 2012. postgresql-dbms.blogspot. [En línea] 2012. http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

Pratt, Michael . 2013. Pratт. *Pratт.* [En línea] 16 de abril de 2013. http://www.michael-pratt.com/blog/13/Patrones-de-Diseno-Inyeccion-de-Dependencias/.

Pressman, Roger S, Ph.D. 2010.*Ingenieria de Software. Un enfoque práctico. Septima Edición.* Mexico, D.F : The Me Graw Hill Companies, 2010.

Pressman, Roger S. 2010.*Ingeniería de Software. Un enfoque práctico.* Septima . México DF : McGraw-Hill INTERAMERICA EDITORES, 2010.

Robledano, Ángel. 2019. OpenWebinars.net. [En línea] 2019. https://openwebinars.net/blog/que-es-java/.

Rodríguez Sánchez, Tamara. 2015.*Metodología de desarrollo para la Actividad productiva UCI.* La Habana : s.n., 2015.

—. 2015.*Metodología de desarrollo para la Actividad productiva UCI.* La Habana : s.n., 2015.

Rodriguez, Tamara. 2015. [En línea] 2015.

Rodríguez, Yordanis Rodríguez. 2018. informaticahabana. [En línea] 2018. http://www.informaticahabana.cu/sites/default/files/ponencias2018/GES16.pdf.

rol. 2017. Significado de Rol. [En línea] 2017. https://www.significados.com/rol/.

Sæther Bakken, Stig, Aulbach, Alexander y Schmid, Egon. 2002.*Manual de PHP.* 2002.

Sánchez, Alejandro Donoso. 2017. [En línea] 2017. https://economipedia.com/definiciones/indice-de-precios.html.

Scrum. 2018. Historias de Usuario, Escritura, Definición, Contexto y Ejemplos. 2018.

Sommerville, Ian. 2011.*Ingeniería de Software.* 9na. México : Addison-Wesley, 2011. ISBN: 978-607-32-0603-7.

SRL, E. S. J. 2011. Lex-Doctor, GESTIÓN JURÍDICA. [En línea] 2011. http://www.lexdoctor.com/productos_estudiosjuridicos_info.php.

Tecnologías-Información. 2018. Modelo de datos. [En línea] 2018. https://www.tecnologias-informacion.com/modeladodatos.html.

TENSTEP.INC. 2016. [En línea] 2016. https://www.tenstep.ec/portal/articulos-boletin-tenstep/41-scrum/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento.

Torre, Fernando Alfonso Casas De la. 2017. [En línea] 2017. https://es.slideshare.net/Indiana_1969/patrones-grasp-76283581.

Uchoa, Florencia. 2014. Definición ABC. [En línea] 2014. https://www.definicionabc.com/economia/indice-de-precios.php.

Uriarte, Julia Máxima. 2020. [En línea] 2020. https://www.caracteristicas.co/sistema-informatico/.

Usuario, Definición de, [prod.]. 2019.*Definición de Usuario.* 2019.

Velasco, Rubén. 2017. PHP 7.2, todas las novedades de esta nueva versión de PHP. [En línea] 13 de noviembre de 2017. [Citado el: 20 de septiembre de 2018.] https://www.redeszone.net/2017/11/13/novedades-php-7-2/.

Visual Paradigm. 2019. [En línea] 2019. http://www.visual-paradigm.com/.

Visual Paradigm, International Ltd. 2017.*Visual Paradigm.* 2017.

vue. 2014-2020. es.vuejs.org. [En línea] 2014-2020. https://es.vuejs.org/v2/guide/index.html.

webstorm. riptutoria. [En línea] https://riptutorial.com/es/webstorm/example/9986/instalacion-y-configuracion.