Universidad de las Ciencias Informáticas

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

Título:

Componente de validación del Estándar de arquitectura de datos en MySQL para la Empresa de Tecnologías de la Información para la Defensa

Autora: Arlety Vitón Leyva
Tutores: Ing. Mauricio Guanche Cañizares
Ing. Reinier Fernández Coello

La Habana, 2020

DECLARACIÓN DE AUTORÍA:

Declaro ser autor de la presente tesis y reconozco a la XETID, Empresa de Tecnologías de la Información para la Defensa, los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma de Autor

Arlety Vitón Leyva

_____

Firma de Tutor

Ing. Mauricio Guanche Cañizares

_____

Firma de Tutor

Ing. Reinier Fernández Coello

Dedicatoria

A mi mamá Yaida, por su amor, dedicación y sacrificio.

A mi papá Pape, por estar siempre ahí.

A mi hermano Jessier, por las peleas y las escondidas de mamá.

A mis abuelos Marcos y Puchita, por su amor incondicional.

A Yaimet Y Bolita por su regalo y el apoyo a mi carrera.

A las mamis, la familia que escogí.

A mi tía Isabel, por ser mi paradigma a seguir.

A mi amigo Josué, por su apoyo y ayuda en estos 5 años.

A mis tías, tíos y primos.

A mis tutores.

Resumen

El uso de las Tecnologías de la Información y las Comunicaciones (TIC) en todas las esferas de la sociedad le ha permitido al mundo alcanzar un desarrollo tecnológico significativo. Este desarrollo va en aumento y en constante perfeccionamiento revolucionario. Buscando de manera incisiva la informatización de la sociedad y una mayor calidad en todo lo que se hace. Cuba no esta ajena a estas metas y se sabe que el empleo de las TIC son un factor decisivo para el desarrollo de las empresas, la economía y la sociedad cubana. La Empresa de Tecnologías de la Información para la Defensa (XETID) se encarga de desarrollar aplicaciones informáticas para diferentes organismos cubanos en busca de la informatización de los procesos y la calidad de los mismos. La XETID cuenta con el Centro de Calidad Estándares y Seguridad (CCES) que se encarga de certificar y evaluar técnicamente productos informáticos garantizando la calidad en los productos que desarrolla la empresa. Actualmente, para el proceso de revisión de las bases de datos de estos productos se implementa como solución final un componente que permita la validación del Estándar de Arquitectura de Datos para las bases de datos implementadas en MySQL de forma automática. Garantizando el cumplimiento y la correcta utilización de este estándar en la nomenclatura de las bases de datos. Este componente facilitará el trabajo de los aseguradores de la calidad, disminuyendo el margen de errores cometidos y aumentando la calidad de los productos que ofrece la empresa al mercado.

Palabras clave: calidad, estándar, base de datos, validación

Índice general

Índice de figuras

Índice de tablas

Introducción

En la actualidad, gran parte del desarrollo alcanzado a nivel mundial se debe al incremento del empleo de las tecnologías en casi todas las esferas de la sociedad. Con el objetivo de elevar el nivel tecnológico de las empresas, organizaciones e instituciones se han puesto en práctica aplicaciones informáticas que ayudan a tener un control eficiente sobre los recursos que manejan. Las aplicaciones de este tipo deben contar con una adecuada Arquitectura de Datos (AD) que se encarga de la gestión, el manejo y control de los datos.

Una AD generalmente debe estar conformada por modelos de datos estandarizados, modelos conceptuales, modelos físicos, clasificación de datos, estándares de codificación SQL (*Structured Query Language*) y gestión de calidad de datos (Espinoza Callo 2016a).

Las empresas que le dan importancia a la calidad de sus datos pueden obtener beneficios claves para agregarle valor al negocio y diferenciarse del resto de sus competidores. Esto le da la posibilidad de minimizar los riesgos en sus proyectos, ahorro de tiempo y recursos, haciendo un mejor uso de la infraestructura tecnológica y sistemas para explotar su información. Además, permite la toma de decisiones de negocios oportunas en base a información confiable, validada y limpia, mejora la confianza, buenas relaciones e imagen de la empresa ante sus clientes frente a la competencia. Es por estas razones que se hace necesario realizar pruebas a estas aplicaciones, que permitan medir la calidad y correcta implementación de una AD (PowerData 2014).

El Dr. Kaoru Ishikawa (1988), a su vez, considera que: "*En su interpretación más estrecha, calidad significa calidad del producto, pero en su interpretación más amplia significa calidad del trabajo, calidad del servicio, calidad de la información, calidad del proceso, calidad de la dirección y calidad de la empresa*" (Ishikawa, Kaoru;1988).

En Cuba existen empresas que se encargan de desarrollar aplicaciones para diferentes entidades que les permitan una mejor gestión de sus procesos. Por la importancia que tiene la calidad para las empresas que se dedican a la industria del software estas utilizan estándares y otras herramientas. Todo esto con el fin de asegurar que se cumpla con el objetivo de brindar al público productos y servicios de calidad. Tal es el caso de la Empresa de Tecnologías de la Información para la Defensa (XETID), en la cual se encuentra el Centro de Calidad Estándares y Seguridad (CCES). El mismo se encarga de certificar y evaluar técnicamente productos informáticos de producción nacional o importada. Todo esto según normas nacionales e internacionales para la industria del software en la Fuerzas Armadas Revolucionarias (FAR).

Actualmente para el proceso de revisión de las bases de datos (BD) de estos productos el CCES cuenta con un Estándar de Arquitectura de Datos definido para los gestores de base de datos. Además, posee un sistema llamado CEAD que permite la revisión automática de la BD en PostgreSQL para validar que se cumpla este estándar. Con el fin de garantizar que se cumpla con los estándares de la XETID y la calidad requerida todos los productos informáticos deben estar certificados por el CCES. Sin embargo, para todos los proyectos que se realizan en el Centro de Audiovisuales se hace difícil cumplir esta tarea. Esto se debe a que no se cuenta con un sistema que permita la revisión automática del Estándar de Arquitectura de Datos en la BD debido a que este centro utiliza como gestor de base de datos MySQL. Por estas razones las pruebas que se efectúan a estas bases de datos se ejecutan de forma manual. Esto trae consigo que se haga más compleja la revisión, provocando demoras en las pruebas de calidad, dando cabida a que exista un mayor margen de errores cometidos. Además, no se logra estandarizar el desarrollo, ni generalizar las herramientas, nomenclaturas y otros elementos de las bases de datos. No se puede asegurar la legibilidad del modelo de datos, ni se facilita la portabilidad entre motores de bases de datos, plataformas, aplicaciones y la tarea de los programadores en el desarrollo de los sistemas. De manera general se dificulta el proceso de revisión en el centro y por lo tanto la calidad de los productos que ofrece la empresa.

A partir de la situación problemática anteriormente expuesta se identifica el siguiente **problema a resolver:** ¿Cómo contribuir a la validación del Estándar de la Arquitectura de Datos en MySQL para la XETID?

Definiendo como **objeto de estudio:** Proceso de validación del Estándar de Arquitectura de Datos de la XETID.

El **campo de acción** lo conforma las herramientas de validación del Estándar de Arquitectura de Datos en MySQL para la XETID.

El **objetivo general** de este trabajo es desarrollar un componente de validación del Estándar de Arquitectura de Datos en MySQL para la XETID.

Para el correcto cumplimiento del objetivo general se desglosan los siguientes **objetivos específicos:**

1. Sistematizar los aspectos teóricos asociados al proceso para la validación del Estándar de Arquitectura de Datos de la XETID.

2. Implementar la solución propuesta para la validación del Estándar de la Arquitectura de Datos en MySQL en la XETID.
3. Ejecutar las pruebas al componente implementado para asegurar la calidad de la solución propuesta.

Teniendo como guía para el cumplimiento de los objetivos las siguientes **tareas de investigación**:

1 Elaboración del marco teórico de la investigación referente al objeto de estudio del proyecto.
2 Descripción de los requerimientos funcionales y no funcionales para el desarrollo del componente en cuestión.
3 Diseño de las interfaces para el componente CEAD MySQL.
4 Implementación del componente a partir de la arquitectura definida.
5 Integración del componente al sistema CEAD en el marco de trabajo de la XETID.
6 Validación del componente elaborado con una revisión e inspección al código fuente.

Los **métodos de investigación** utilizados para dar solución a la presente investigación son:

Teóricos:

- **Análisis y síntesis**: se utilizó a la hora de analizar los diferentes conceptos asociados a la investigación y en la búsqueda de los elementos más importantes que se relacionen con el desarrollo de aplicaciones web.
- **Histórico-lógico**: se empleó en la investigación realizada a partir del estudio del estado del arte de la problemática analizada, que permitirá conocer la trayectoria y desarrollo de aplicaciones para revisar base de datos. Teniendo en cuenta sus antecedentes históricos y las investigaciones realizadas, lo que dará la posibilidad de seleccionar aquellas herramientas, lenguajes y metodologías lógicas y adecuadas para el desarrollo de la aplicación.
- **Modelación**: se utilizó para representar los procesos definidos por el negocio mediante la construcción de modelos y diagramas a lo largo del desarrollo de la investigación, simplificando la realidad y facilitando la comprensión de los mismos.

Empíricos:

- **Entrevista:** se utilizó en encuentros con el cliente para definir funcionalidades de la aplicación web.

Con el objetivo de lograr una mayor comprensión de este documento, se decide estructurarlo en 3 capítulos. A continuación, se describen brevemente los mismos:

Capítulo 1- Fundamentación teórica: se efectúa un análisis de los principales temas a tener en cuenta para el desarrollo de la investigación. Donde también se incluye el estudio de diversas

herramientas que pudieran permitir el trabajo con base de datos (en este caso una revisión de las mismas) y sus antecedentes. Además, se seleccionan las tecnologías, lenguaje y herramientas de modelado, lenguajes de programación siguiendo el proceso de desarrollo que se define en Prodesoft.

Capítulo 2- Características y diseño del componente: se describen las características del componente a desarrollar. Se plantea la propuesta de solución y se determinan los principales conceptos para entender el problema y comenzar con la modelación del mismo. De acuerdo con el cliente, se toman los requisitos funcionales y no funcionales que serán objetivo de la informatización, así como la descripción de los mismos y el diseño de los prototipos de interfaz de usuario. Además, se determina la arquitectura a utilizar en el componente y se diseñan las clases y la base de datos. Así mismo, se describen los patrones de diseño a utilizar y el patrón arquitectónico que servirá de base y como enfoque para el manejo de algunas características del componente.

Capítulo 3- Implementación y prueba: se describe todo el proceso desarrollado durante la implementación y las correspondientes pruebas a realizarle al componente. Validando así la solución propuesta, utilizando para ello pruebas de caja blanca y caja negra.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se realiza un estudio de los diferentes temas que son de vital importancia para el desarrollo de la investigación. Se argumenta sobre los conceptos a tener en cuenta para dar solución a la problemática planteada. También se da una breve panorámica de las soluciones tecnológicas existentes en el mundo con respecto al objetivo general del trabajo. Por otra parte, se especifica la metodología a implementar y las fases que la componen. Además, se realiza una pequeña descripción sobre las herramientas utilizadas en la aplicación a desarrollar.

1.2 Conceptos asociados al dominio de la investigación

La **calidad de software** es la concordancia con los requisitos funcionales y de rendimiento, con los estándares de desarrollo y con las características implícitas que se espera del software desarrollado profesionalmente (Pressman 2010a). La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, programación y prueba del software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor conformidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software (Aponte et al. 2015).

Los **estándares** definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del Software. Si no se sigue ningún estándar siempre habrá falta de calidad. Todos los estándares tienen un único fin producir software de alta calidad (Cueva Lovelle 1999).

La **Arquitectura de Datos** es un elemento fundamental para que los sistemas de gestión de información y de organización empresarial tengan éxito. En ella se integran los modelos, las políticas y las reglas que rigen qué datos se van a recopilar; cómo van a ser almacenados, clasificados y explotados mediante la infraestructura tecnológica disponible (Cognodata, 2019). El reto de la arquitectura de datos es resolver la integración y la utilización de la información, la construcción de herramientas y procesos para su gestión y administración, contar con un modelo de toma de decisiones basada en los datos recolectados, sistemas inteligentes para ayudar a los gerentes a la toma de decisiones (Espinoza Callo 2016b).

Luego de estudiar los conceptos anteriores se asume que la **Arquitectura de Datos** describe la estructura de los datos físicos y lógicos de la organización y sus modelos de gestión. Es un elemento fundamental en la arquitectura empresarial sentando las bases para la toma de decisiones.

La **validación del Estándar de la Arquitectura de Datos** a la cual se hace referencia en la presente investigación consistirá en comprobar que la nomenclatura utilizada en las bases de datos de cualquier proyecto desarrollado en la XETID concuerde con el estándar de arquitectura de datos definido en la empresa. La importancia que se le concede a esta validación está ligada a las ventajas de tener bases de datos estandarizadas en una empresa maximizando la utilidad de los datos dentro de ellas.

1.3 Soluciones existentes

Con el estudio de las soluciones existentes se identificaron varios sistemas que pudieran cumplir con las necesidades del problema planteado, los cuales cuentan con funcionalidades que permiten realizarle pruebas a una base de datos. A continuación, se describen las soluciones estudiada.

1.3.1. Apache JMeter

Una de las herramientas para pruebas de base de datos que funciona con páginas web, escrito completamente en lenguaje Java y orientado directamente a los programadores. Si se necesita probar la carga masiva de los datos y realizar pruebas de esfuerzo a los servidores se puede utilizar Apache JMeter. No es un navegador web (no ejecuta el JavaScript del código HTML, por ejemplo). También ofrece opciones de registro de resultados en un simple archivo de texto (Claudia 2018a). Apache JMeter no permite validar la utilización de un estándar de arquitectura de datos en la base de datos que revisa. Además, no cuenta con la función de verificar que esta BD tenga una arquitectura bien implementada.

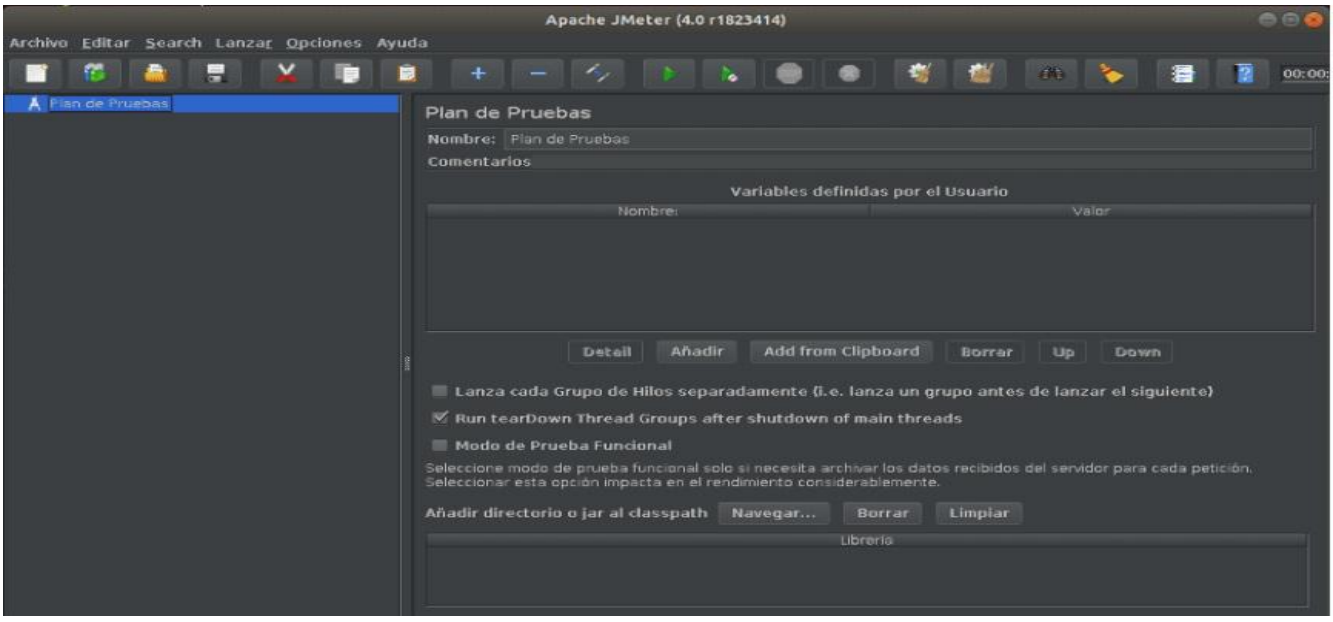


Figura 1: Interfaz gráfica de Apache JMeter

Fuente: (Amoedo 2018)

1.3.2. SQLMap

Es una herramienta de auditoría de bases de datos a la que se le debe proporcionar una url (*Uniform Resource Locator*), para que evalúe distintos tipos de vulnerabilidades para bases de datos como: MySQL, PostgreSQL, Microsoft SQL Server y Oracle. Los que prueba principalmente la herramienta es inyectar el parámetro que se le pasa en la url para obtener acceso a los datos. También se puede incrementar el número de *test* o pruebas que realiza (Khepri 2018). SQLmap no cuenta con una función para comparar una arquitectura de datos y un estándar definido previamente. Tampoco permite verificar por separado que la BD que se prueba cuente con una correcta arquitectura de datos.

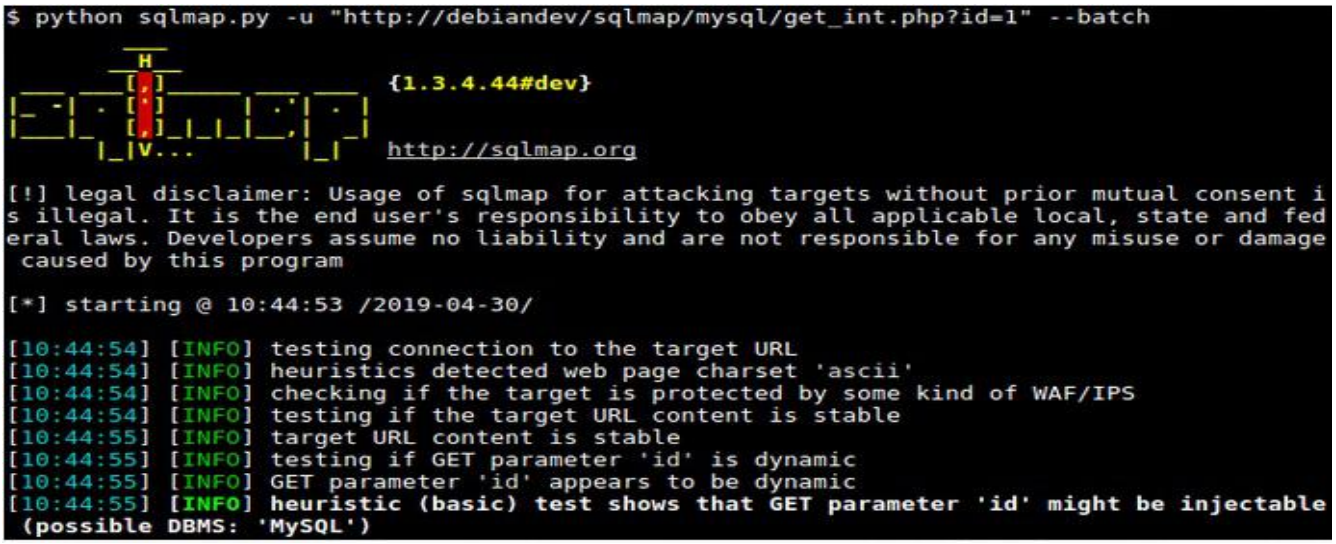


Figura 2: Ejecución de SQLmap

Fuente: (Damele y Stampar 2020)

1.3.3. DbUnit

Es un marco de trabajo (*framework*) de código abierto, permite gestionar el estado de una BD durante las pruebas unitarias y de integración. DbUnit puede trabajar con conjuntos de datos grandes. Acepta verificar que el contenido de la base de datos es igual a determinado conjunto de datos esperado, a nivel de fichero, a nivel de consulta o bien a nivel de tabla. Proporciona un mecanismo basado en XML (*Extensible Markup Language*) para cargar los datos en la BD y para exportarlos desde la BD. Además, sirve como forma de aislar los experimentos en distintos casos de prueba individuales, uno por cada operación (Universidad de Alicante 2014). DbUnit no utiliza estándares para la revisión de las bases de datos ni permite que se utilice uno. Además, no ofrece la posibilidad de comprobar que una BD tenga una correcta arquitectura de datos.

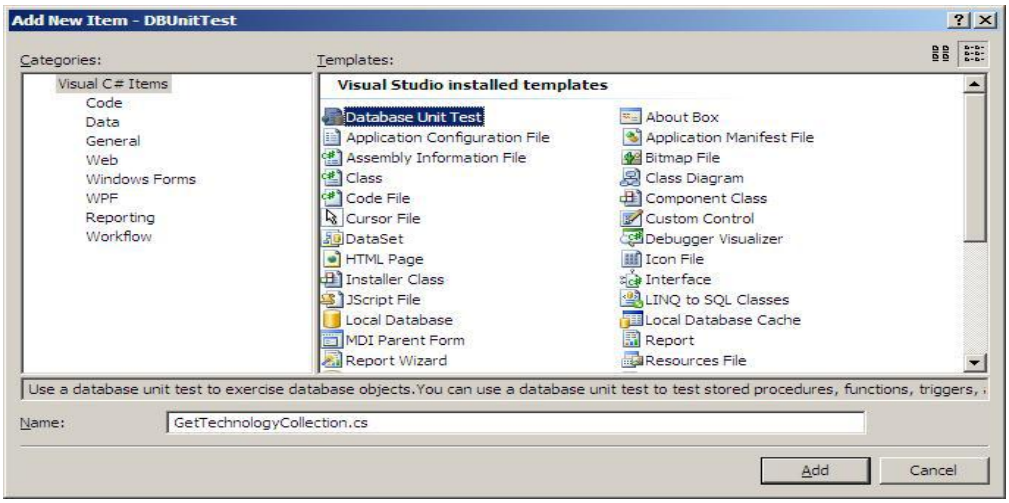
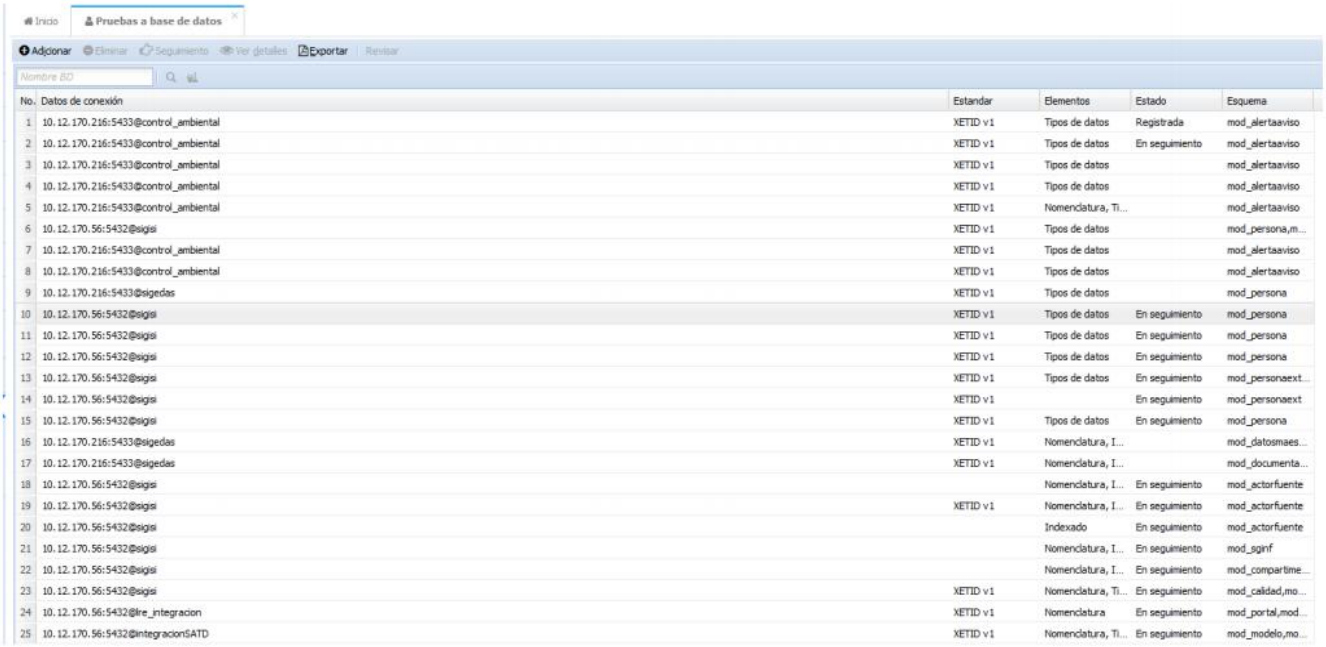


Figura 3: Interfaz de DbUnit

Fuente: (Reyes Pérez 2016b)

1.3.4. Componente de revisión de Arquitectura de Datos para PostgreSQL

Es un componente integrado al marco de trabajo de la XETID Zeolides, el cual permite comprobar una correcta nomenclatura de arquitectura de datos automáticamente, además de que parte de un Estándar conocido. Este componente para todos los controladores de calidad administra una poderosa herramienta que permite disminuir el tiempo de trabajo empleado. Realiza una búsqueda exhaustiva de no conformidades proporcionando un menor margen de error con respecto al método de búsqueda y comparación anterior al mismo (Reyes Pérez 2016a). La herramienta CEAD actualmente permite revisar solo bases de datos desarrolladas con el gestor PostgreSQL, por lo que, a pesar de permitir la validación de un Estándar de Arquitectura de Datos, este no satisface completamente los requerimientos del problema planteado.



No.	Datos de conexión	Estandar	Elementos	Estado	Esquema
1	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos	Registrada	mod_alertaaviso
2	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos	En seguimiento	mod_alertaaviso
3	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos		mod_alertaaviso
4	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos		mod_alertaaviso
5	10.12.170.216:5433@control_ambiental	XETID v1	Nomenclatura, Ti...		mod_alertaaviso
6	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos		mod_persona,m...
7	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos		mod_alertaaviso
8	10.12.170.216:5433@control_ambiental	XETID v1	Tipos de datos		mod_alertaaviso
9	10.12.170.216:5433@sigedas	XETID v1	Tipos de datos		mod_persona
10	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos	En seguimiento	mod_persona
11	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos	En seguimiento	mod_persona
12	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos	En seguimiento	mod_persona
13	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos	En seguimiento	mod_personaext...
14	10.12.170.56:5432@sigisi	XETID v1		En seguimiento	mod_personaext
15	10.12.170.56:5432@sigisi	XETID v1	Tipos de datos	En seguimiento	mod_persona
16	10.12.170.216:5433@sigedas	XETID v1	Nomenclatura, I...		mod_datosmaes...
17	10.12.170.216:5433@sigedas	XETID v1	Nomenclatura, I...		mod_documenta...
18	10.12.170.56:5432@sigisi		Nomenclatura, I...	En seguimiento	mod_actorfuente
19	10.12.170.56:5432@sigisi	XETID v1	Nomenclatura, I...	En seguimiento	mod_actorfuente
20	10.12.170.56:5432@sigisi		Indexado	En seguimiento	mod_actorfuente
21	10.12.170.56:5432@sigisi		Nomenclatura, I...	En seguimiento	mod_sginf
22	10.12.170.56:5432@sigisi		Nomenclatura, I...	En seguimiento	mod_compartime...
23	10.12.170.56:5432@sigisi	XETID v1	Nomenclatura, Ti...	En seguimiento	mod_calidad,mo...
24	10.12.170.56:5432@lre_integracion	XETID v1	Nomenclatura	En seguimiento	mod_portal,mod...
25	10.12.170.56:5432@integracionSATD	XETID v1	Nomenclatura, Ti...	En seguimiento	mod_modelo,mo...

Figura 4: Interfaz de CEAD

Fuente: (Reyes Pérez 2016)

A partir del análisis del estudio elaborado se realiza una comparación entre estas herramientas a partir de los parámetros: lenguaje de programación que utiliza, si es privativo, si permite la validación de un estándar y de una arquitectura de datos, como se muestra en la siguiente tabla:

Tabla 1: Comparación entre herramientas existentes

Herramientas	Lenguaje	Privativo	Validación de estándar	Validación de arquitectura de datos
APACHE JMeter™	Java	No	No	No
dbUnit	Java	No	No	No
sqlmap®	Python	No	No	No
CEAD	PHP	No	Si	Si

Fuente: Elaboración propia

Donde se concluye como resultado del análisis, que ninguna de estas herramientas permite verificar que la arquitectura de la base de datos de un proyecto cumpla con un estándar definido previamente, salvo la herramienta CEAD. De esta última se tomarán funcionalidades y algunos requisitos de diseño para ser utilizados en la propuesta de solución, pues no cumple con todos los requisitos, pero se relaciona con lo que se desea implementar. A partir de lo dicho anteriormente se evidencia la necesidad de una solución informática, que permita la validación del Estándar de Arquitectura de Datos de la XETID. Para que todos los productos informáticos que utilicen una base de datos desarrollada en MySQL, realizados en la empresa, sean correctamente revisados por el CCES. Esta solución informática podría permitir un aumento de la eficiencia para tratar la información y obtener de ella conocimiento. Un modo de maximizar la utilidad de este recurso (los datos) de vital importancia en las organizaciones. Los proyectos después de ser revisados contarían con una fuente de información (base de datos) de mejor calidad, lo cual produciría mejores resultados. Para dar comienzo al desarrollo de la propuesta de solución es necesario utilizar una guía que permita entender y organizar mejor este proceso.

1.4 Proceso de desarrollo de software

En la Empresa de Tecnologías de la Información para la Defensa (XETID), es usada la metodología PRODESOFT para dirigir el proceso de desarrollo de todos los productos informáticos que se desarrollan. Por tanto, es la metodología que se selecciona para guiar el proceso de desarrollo de la propuesta a realizar. PRODESOFT cuenta con 5 fases en el ciclo de vida del software: inicio, modelación, construcción, explotación experimental y despliegue. Con estas fases se logra tener más control sobre el software que se desarrolla en la institución y se garantiza la calidad del mismo (figura 5).

Figura 5: Fases del ciclo de vida de un software

Fuente: (Prodesoft 2014)

Durante la fase de **Inicio** se obtiene una visión previa de la problemática. Se describen los objetivos y alcance del proyecto, se identifican los ejecutores y los involucrados, se estiman las actividades a

realizar durante el desarrollo del proyecto, además de establecer la estrategia a seguir para la modelación del negocio y la obtención de requisitos.

En la fase de **Modelación** se capturan las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requerimientos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación aprobada por los implicados en el proyecto. El hito fundamental de esta fase es la liberación de la arquitectura de sistema, datos y despliegue.

En la fase de **Construcción** se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. Las fases anteriores sólo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye el producto. En esta fase todas las características, componentes, y requerimientos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto.

Durante la fase de **Explotación Experimental** se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto. Esta fase solo se ejecuta cuando se tiene como cliente al Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR), por requerimientos que exige este órgano.

En la fase de **Despliegue** se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas (Prodesoft 2014a).

Después de ser presentado el proceso de desarrollo que servirá de guía para la implementación de la solución propuesta, es necesario conocer las tecnologías que se utilizarán en esta.

1.5 Lenguaje de modelado

El UML (*Unified Modeling Language*) se puede utilizar para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas. La aplicación del lenguaje UML hace necesaria la participación del usuario en la definición de requerimientos y por ende mejora notablemente un sistema según sean las necesidades del usuario. Además, dedicarle el tiempo necesario al diseño provoca que en las etapas de construcción e implementación se mejore el tiempo de desarrollo. Esto debido a que los errores fueron corregidos en las fases de mayor impacto con el sistema (UNAD 2018).

1.6 Herramienta de modelado

La creación del software es un proceso intrínsecamente creativo y la ingeniería del software (IS), trata de sistematizar este proceso con el fin de acotar el riesgo del fracaso en la consecución del objetivo, por medio de diversas técnicas que se han demostrado adecuadas en base a la experiencia previa. En la XETID, los proyectos hacen uso de la herramienta Visual Paradigm para UML, esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos (Pérez y Alcántara 2014).

Visual Paradigm for UML v8.0.

Visual Paradigm for UML soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Galeano 2016).

Es una herramienta para desarrollo de aplicaciones ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en la construcción de sistemas a gran escala que brinden confiabilidad y estabilidad en el desarrollo orientado a objetos. Ofrece potente generador de informes en formato PDF, documentación automática, ambiente visualmente superior de modelado y sofisticado diagramador (Hernández Pérez 2017).

1.7 Lenguajes de programación

Un lenguaje de programación es un sistema de comunicación que permite que los programadores informáticos puedan dar instrucciones a las máquinas. Igual que en el caso de los idiomas, que pueden compartir la manera de estructurar las frases o presentar similitudes de palabras, pero en otros casos siguen un vocabulario totalmente distinto, cada lenguaje de programación sigue unas reglas sintácticas y semánticas determinadas (García 2019).

Para el desarrollo de la solución se utilizará PHP como lenguaje de programación, entre otros, debido a su utilidad para implementar sistemas web.

1.7.1 PHP v5.6

PHP (*Hypertext Preprocessor*) es un lenguaje multiplataforma. Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar el resultado HTML al navegador. Esto hace que la

programación en PHP sea segura y confiable. Cuenta con una capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destacando su conectividad con MySQL y PostgreSQL. Funciona de igual manera para Windows y para Linux. Su gran comunidad hace que el soporte, guías, libros y soluciones de dudas sean mucho más fáciles de consultar en foros y redes sociales. No requiere ningún tipo de licencia y además permite la programación orientada a objetos (Morales 2016).

1.7.2 JavaScript v1.8

Lenguaje de programación que se encuentra presente en prácticamente cualquier aplicación web. Empresas como Google o Facebook han desarrollado incluso sus propias implementaciones basadas en este lenguaje para lograr por ejemplo una experiencia visual del usuario más atractiva. Ha hecho posibles las aplicaciones web modernas, aplicaciones con las que puede interactuar directamente sin hacer una recarga de página para cada acción. JavaScript fue ideado para concederle a la web capacidades interactivas que le ayudarán a dar el salto al siguiente nivel permitiendo crear una interfaz de usuario activa. Por ejemplo, es común usar JavaScript en la validación de formularios para asegurar que la información introducida es válida. Sin necesidad de enviar ninguna información al servidor, el programa realiza los cálculos necesarios ahorrando tiempo y recursos del lado del servidor. Este lenguaje le permite al programador hacer todo lo que quiera, pues es bastante tolerante, relativamente simple de aprender e implementar. Esto hace que se puedan usar diferentes técnicas para hacer el código lo más eficiente posible (Robledano 2019). Además es muy rápido y cualquier función puede ser ejecutada inmediatamente en lugar de tener que contactar con el servidor y esperar una respuesta. Encaja perfectamente con otros lenguajes y puede ser usado en una gran variedad de aplicaciones. Puede insertarse en cualquier página independientemente de la extensión del fichero y ser usado dentro de scripts escritos en otros lenguajes como PHP. Al ejecutarse del lado del cliente reduce la carga en el servidor de la página web (Rawat 2016).

1.7.3 XML

XML (*Extensible Markup Language*): Se puede definir como un metalenguaje formado por etiquetas estructuradas jerárquicamente, de fácil comprensión tanto por máquinas como por personas, y adaptable a los requerimientos necesarios de cada caso. La necesidad de gestionar los contenidos de una manera rápida y organizada, se hace absolutamente necesaria. XML y las especificaciones que derivan de él, ofrecen una amplia gama de soluciones a los diferentes problemas que surgen en este campo. Rápido, flexible, sencillo o extensible son solo algunas de las cualidades que se puede destacar de este metalenguaje. Actualmente las aplicaciones del lenguaje XML son muy extensas, se

ha utilizado tanto para la definición de archivos de configuración, numerosos protocolos de comunicación y otros. Algunos de sus usos más extendidos son:

- ✓ **Intercambio de datos entre sistemas,** uno de los objetivos fundamentales de XML es permitir la posibilidad de intercambiar datos de forma estructurada entre diferentes sistemas. Al tratarse de un formato de texto plano y ser un lenguaje estandarizado, hace que esta transferencia sea muy ágil e independiente de la plataforma utilizada.

- ✓ **Base de datos,** XML permite guardar datos de forma estandarizada para luego poder ser tratados por multitud de lenguajes diferentes. Su manejo es mucho más sencillo que bases de datos como MySQL.

- ✓ **Conversor,** actualmente son muchos los formatos que ofrecen servicios de conversión a XML (Ibermática 2017).

1.8 Sistemas gestores de base de datos

Un Sistema Gestor de Base de Datos (SGBD) permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de información del modo más eficiente posible (Marín 2019). El marco de trabajo de la XETID Zeolides actualmente utiliza como gestor de base de datos PostgreSQL. Debido a que la solución propuesta estará integrada a este marco de trabajo se hace necesario trabajar también con este SGBD.

1.8.1. PostgreSQL v9.4

Sistema que permite la gestión de las bases de datos. Una de las principales razones por la cual PostgreSQL es tan utilizado se debe a que se trata de un sistema de código abierto, totalmente gratis. Es un software que puede correr bajo distintos entornos y sistemas operativos. La facilidad de uso es sin dudas otra de las principales características de este sistema. Una característica extremadamente importante de PostgreSQL es su gran capacidad para el manejo de grandes volúmenes de datos. (Borges 2019).

1.9 Entorno de desarrollo integrado

IDE (*Integrated Development Environment*) entorno donde el programador tiene todas las herramientas de trabajo a su disposición. Los entornos de desarrollo son las herramientas con las cuales los programadores crean aplicaciones. Funciona como editor de código, compilador, depurador y constructor de interfaces gráficas (Moreno Pérez 2018). Para la solución propuesta se decide utilizar el IDE NetBeans ya que es libre y gratuito sin restricciones de uso y es una plataforma

de aplicaciones que permite a los desarrolladores crear rápidamente aplicaciones del tipo web y empresarial.

1.9.1. NetBeans v8.1

NetBeans es un IDE gratuito y de código abierto. Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, como aplicaciones web, o para dispositivos móviles. Da soporte a tecnologías como PHP. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. Es un buen editor de código, multilenguaje, simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda y estructura la visualización de manera ordenada. Cuenta con herramientas para el depurado de errores. Permite la optimización del código y las aplicaciones e intenta hacer que se ejecuten más rápido y con el mínimo uso de memoria. Admite el acceso a base de datos: desde el propio NetBeans se puede conectar a distintos sistemas gestores de bases de datos y ver las tablas, realizar consultas y modificaciones, y todo ello integrado en el propio IDE. Es fácilmente extensible a través de plugins (Calendamaia 2014).

1.10 Marco de trabajo

Marco de trabajo (*framework*) es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta. Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones (Constanzo y Casas 2018). En la XETID se utiliza Zeolides como marco de trabajo, el cual integra una serie de herramientas y componentes de los que podrá hacer uso la propuesta de solución a desarrollar.

1.10.1 Zeolides v2.2.0

Conjunto de componentes de software librerías, herramientas y tecnologías libres integradas que permiten el desarrollo ágil y basado en componentes de aplicaciones web empresariales, centrando el desarrollo en el negocio, los requerimientos y las interfaces de usuario. Este framework presenta una estructura que facilita la comprensión por parte de los desarrolladores que trabajan sobre el mismo. En este ambiente se utiliza Ext JS para implementar la capa de presentación y utiliza el patrón arquitectónico Modelo Vista Controlador (MVC) (Reyes Pérez 2016b).

1.10.2 Ext JS v4.1

Ext JS es un framework que nace para trabajar sobre librerías. Ayuda a mejorar la experiencia del usuario, pues proporciona componentes con funcionalidades avanzadas y de fácil implementación. Es

la más poderosa plataforma de desarrollo de interfaces de escritorio en web con compatibilidad sin igual para todos los navegadores más utilizados. El desarrollo se basa en el patrón de Modelo-Vista-Controlador y se ejecuta en el navegador del usuario. Si se tiene que enumerar las principales ventajas que aporta, se puede hablar de que es orientado a objetos, permite controles de interfaz de usuario, temas y estilos prefijados y cuenta con una completa documentación. Desde el punto de vista del programador permite la generación de componentes, formularios y componentes prefabricados. Es un framework pensado para realizar todo el trabajo en el lado cliente (García 2015).

1.11 Servidor web

Los servidores web son uno de los aspectos más importantes de Internet, ya que se trata de los encargados de enviar las páginas a los usuarios. Sin ellos, Internet como lo conocemos hoy en día simplemente no sería posible (Borges 2019). El componente a desarrollar debe integrarse al marco de trabajo de la XETID Zeolides. Este utiliza entre sus tecnologías el servidor web Apache, por lo tanto, es el servidor con el cual se decide trabajar para implementar el componente como solución web al problema planteado.

1.11.1. Apache v2.4.23

El servidor Apache es uno de los servidores más usados que existen, y de los más flexibles a la hora de configurar para maximizar su utilidad. Es sin dudas una gran opción para un entorno de desarrollo, extremadamente sencillo de instalar independiente del sistema operativo. Se disponen de una gran cantidad de módulos para expandir las funciones y personalizar la instalación. Es multiplataforma y en el caso de Windows se lo puede encontrar empaquetado junto con el servidor MySQL y PHP en un solo instalador que facilita enormemente la instalación. Apache integra funcionalidades para brindar seguridad a aplicaciones gracias a los módulos de Autorización y Autenticación y funciones de Control de Acceso, además de un soporte para cifrado (De León 2019).

1.12 Conclusiones del capítulo

En este capítulo, una vez abordados los elementos teóricos que dan sustento a la propuesta de solución del problema planteado se tiene que el análisis de los conceptos asociados a la investigación permitió una mejor comprensión de los temas relacionados con la problemática a resolver. Así mismo, con el estudio realizado de las características y funcionalidades de las herramientas existentes de pruebas a base de datos se comprobó que no satisfacen los requerimientos del problema, evidenciando la necesidad de un sistema informático que permita darle solución al mismo. Además, se planteó que el proceso de desarrollo del software estará guiado por Prodesoft, utilizado en la XETID para dirigir y estandarizar el desarrollo de sus productos informáticos. También, con el estudio de las tecnologías de desarrollo de software a utilizar en la aplicación permitió la elección de las

herramientas adecuadas para desarrollar dicho sistema. Entre estas se encuentran: Visual Paradigm for UML v8.0 para el modelado, NetBeans v8.1 como entorno de desarrollo, PostgreSQL v9.4 como sistema gestor de base de datos, Apache v2.0 como servidor web. Además, también otras tecnologías como PHP, JavaScript, XML, UML, ExtJS y como marco de trabajo para el desarrollo de aplicaciones web Zeolides v2.0.

Capítulo 2: Características y diseño del componente

2.1 Introducción

En este capítulo se ponen en evidencia las características principales del componente de revisión de la arquitectura de datos para MySQL de la XETID. Se planteará la propuesta de solución al problema con el que se viene trabajando anteriormente, así como las funcionalidades que este deberá cumplir para satisfacer las necesidades del cliente, con su previa descripción y la relación entre los componentes que la integran. Además, se presentan los principios de diseño, modelos y patrones del diseño arquitectónico del componente y los prototipos de interfaz de usuario.

2.2 Propuesta de solución

Partiendo de que los sistemas existentes en el mundo no cuentan con una funcionalidad que les permita validar que la arquitectura de una base de datos diseñada con el SGBD MySQL cumpla con un estándar definido, se toma la decisión de implementar un componente que logre automatizar el proceso de revisión de las bases de datos de los proyectos que son auditados en el centro de calidad de la XETID.

La solución contendrá el componente CEAD MySQL, que estará estructurado de manera visual en dos áreas de trabajo fundamentales. La primera permitirá adicionar, modificar o eliminar una prueba determinada y listar todas las adicionadas anteriormente. Además, también admitirá darle seguimiento a una prueba determinada, así como ver los detalles de esta y exportar el listado de pruebas como informe. El componente también constará con un lenguaje claro y una interfaz sencilla que le permitirá al usuario utilizarlo sin complejidad. La selección de los elementos que se desee revisar en la arquitectura, la obtención de incidencias y detalles de una prueba determinada también será posible para el usuario. Siempre cumpliendo que anteriormente se haya adicionado un estándar a la prueba y se haya establecido una conexión previa con la base de datos.

Una vez realizada una prueba se podrá hacer uso de la segunda área de trabajo la cual le brindará al usuario la posibilidad de visualizar los errores obtenidos al concluir la misma. También podrán ser generadas automáticamente las no conformidades encontradas en el componente No Conformidades.

Una vez descrita la propuesta de solución ya se pueden identificar los conceptos relacionados con el dominio del problema. A partir del análisis de estos conceptos se crea el modelo conceptual que permitirá comprender mejor la situación que se debe resolver.

2.3 Modelo conceptual y sus conceptos

Un modelo conceptual muestra los conceptos presentes en el dominio del problema y las relaciones entre ellos. Un concepto, en este caso, es la representación de cosas del mundo real, no de componentes de software (ABIZTAR 2018). El siguiente modelo conceptual contiene los conceptos considerados como más significativos para el dominio y análisis del problema. Estos conceptos y la relación que se establece entre ellos servirán de guía en esta etapa de diseño y modelación. El Estándar para la modelación de negocio y gestión de requisitos (Ayala 2014) sirvió de guía para el diseño de este modelo.

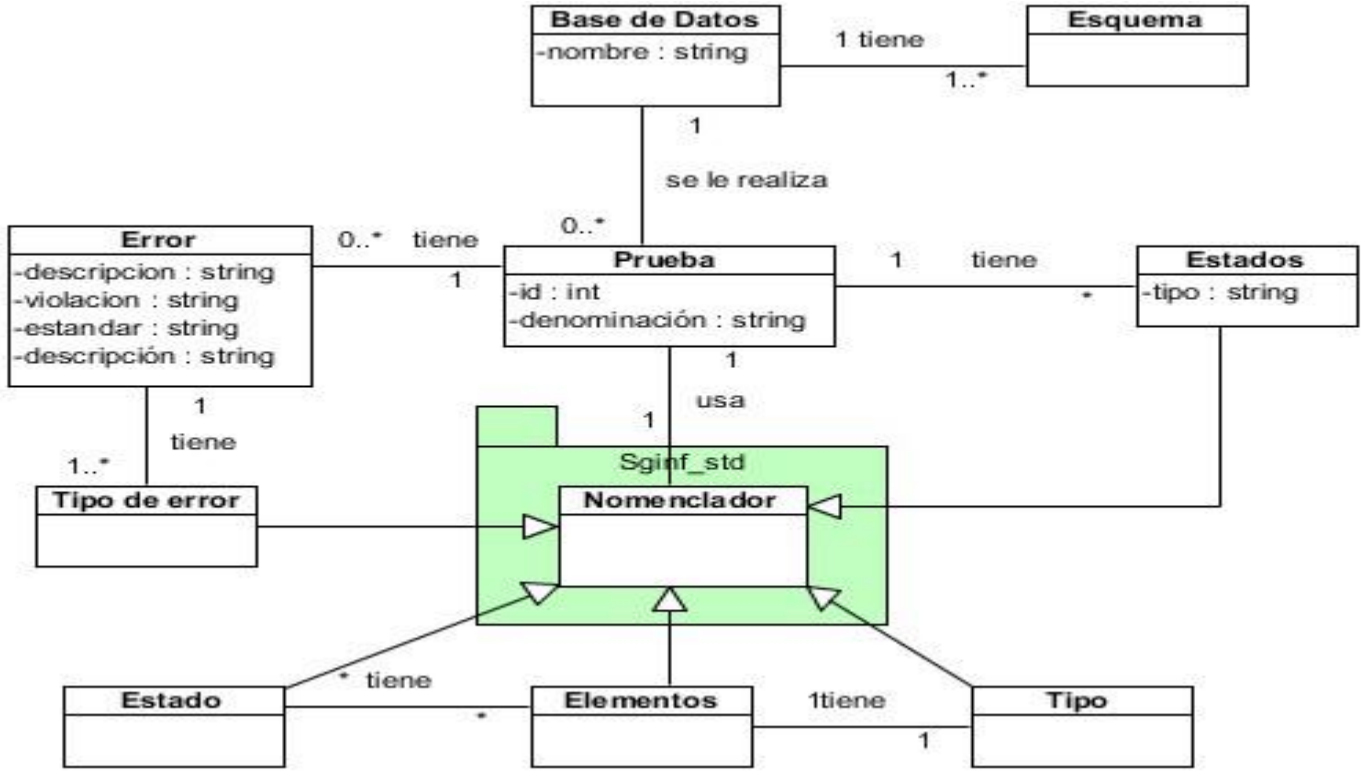


Figura 6: Modelo conceptual de la propuesta de solución

Fuente: Elaboración propia

Conceptos definidos:

Prueba: contiene las pruebas que se le realicen a la base de datos.

Estándar: contendrá un patrón, modelo o punto de referencia con el que deberá cumplir la base de datos que se revise.

Error: describe los errores obtenidos de la arquitectura de datos de la base de datos que se revise.

Tipo de error: los tipos de errores que puede haber son: indexado, tipo de datos y nomenclatura.

Elementos: define los nombres propios de bases de datos, esquemas, tablas, campos, llaves, funciones de dominio y secuencias.

Base de datos: contiene la base de datos a la cual se le realizara la prueba.

Esquemas: son parte de la base de datos, donde se encuentran las tablas con toda la información contenida.

Estados: se refiere al estado en el que se encuentra la prueba. Estos estados pueden ser: en seguimiento, registrada y cerrada.

2.4 Requisitos funcionales

Los requisitos funcionales hacen referencia a la descripción de las actividades y servicios que un sistema debe proveer. Normalmente este tipo de requerimientos están vinculados con las entradas, las salidas de los procesos y los datos a almacenar en el sistema. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Villar 2017).

Los requisitos funcionales son las facilidades que el sistema brindará o deberá brindar al usuario. Deben ser claros, fáciles de probar y específicos. En el caso de CEAD MySQL, se va a dividir en tres áreas de trabajo, una para gestionar estándares, otra para la selección de los elementos que se desean revisar de la arquitectura de datos y otra para la revisión de la arquitectura de datos en sí. Para la obtención de estos requisitos funcionales se realizó una entrevista al cliente y junto con él se determinaron las funcionalidades con las cual debería contar el componente. A continuación, se presenta lo acordado.

RF1: Gestionar prueba a la Base de Datos

RF1.1: Adicionar prueba

RF1.2: Modificar prueba

RF1.3: Eliminar prueba

RF1.4: Listar prueba

RF2: Mostrar seguimiento de la prueba a la Base de Datos

RF3: Mostrar detalles de la prueba

RF4 Realizar revisión de la Base de Datos

RF5 Listar errores de revisión

2.4.1 Descripción de los requisitos funcionales

Para un mayor entendimiento del componente que se propone como solución al problema a resolver planteado en la presente investigación, a continuación, se describen cada uno de los requisitos funcionales del mismo.

Tabla 2: Descripción de los requisitos de la propuesta de solución

Requisito funcional	Descripción
Gestionar prueba a la Base de Datos	Permite Adicionar, Modificar, Eliminar y Listar pruebas de arquitectura de datos a determinadas BD.
Mostrar seguimiento de la prueba a la Base de Datos	Muestra todas las acciones realizadas a la prueba de Base de Datos desde su creación hasta su liberación.
Mostrar detalles de la prueba	Muestra los detalles de las revisiones a la arquitectura de la Base de Datos.
Realizar revisión de la Base de Datos	Realiza una prueba a la arquitectura de datos de la Base de Datos seleccionada.
Listar errores de revisión	Se listan todos los errores encontrados durante la revisión a la Base de Datos.

Fuente: Elaboración propia

2.4.2 Especificación de los requisitos funcionales

Con el objetivo de describir el entendimiento común alcanzado por los involucrados respecto a los requisitos funcionales del componente propuesto, a continuación, se especifican cada uno de estos requisitos:

Especificación del requisito: Adicionar prueba de la Base de Datos

Conceptos tratados	Conceptos	Atributos
	Prueba Estándar Elemento	Id Denominación

	Tipo	
Precondiciones	**Precondiciones**	**Pre-requisito**
	Autenticación del usuario en el rol de asegurador de la calidad.	Autenticar usuario.
Descripción	1. Se selecciona la opción **Inicio/XETID/Calidad/ Prueba a BD.** 2. Se muestra la interfaz Prueba a BD, y se habilita el espacio de trabajo Pruebas a Bases de Datos en MySQL. 3. Se presiona el botón Adicionar de la barra de herramientas del espacio de trabajo Pruebas a Bases de Datos en MySQL. 4. Se muestran los campos a llenar con el botón guardar deshabilitado. 4.1. Si se presiona Cancelar se cancela la prueba. 5. Automáticamente se muestra la ventana Establecer datos de conexión ver Figura 2.2, en la cual se introducirán los datos: • Dirección: IP del servidor de BD. • Puerto: Puerto del servidor de BD. • Usuario: Usuario de la BD • Clave: Contraseña del usuario de la BD • Nombre de la BD: Nombre de la BD a la que se realizará la prueba. 5.1. Si se presiona el botón Cancelar se cierra la ventana. Si se presiona el botón Aceptar, se validan los datos y se mostrará estructurado en la columna Denominación. A continuación se llenan el resto de los campos: • Estándar: Lista de los nomencladores existentes.	

	• Elementos: Tipos de pruebas de BD que se realizarán. • Estados: Flujo que se generará de forma automática. Comenzará primeramente en Registrado. • Esquemas: Esquemas a los cuales se le realizará la revisión. 6. Si se presiona el botón Aceptar, se validan y guardan los datos, y se muestra el mensaje: “La prueba ha sido adicionada satisfactoriamente.”. 6.1 Si se presiona el botón Cancelar se cancela la acción, y se termina el flujo.
Complejidad	Baja.
Validaciones	Para las acciones Aceptar Los botones se habilitarán cuando se haya introducido los valores correctos, en el campo. El sistema valida los datos según lo descrito en el documento Modelo conceptual_CEAD MySQL. Si ocurre un error al intentar acceder a la Base de Datos, el sistema muestra el mensaje: "Los datos de conexión son incorrectos. " Sobre *tooltips*: La opción Adicionar del espacio de trabajo Pruebas a Bases de Datos en MySQL muestra como mensaje tooltips “Adicionar prueba **(Alt+I)**”. La opción Guardar del espacio de trabajo Establecer datos de conexión muestra como mensaje tooltips “Guardar datos”. La opción Cancelar del espacio de trabajo Establecer datos de conexión muestra como mensaje tooltips “Cancelar acción”. El botón Cancelar de la interfaz Pruebas a Bases de Datos en MySQL muestra como mensaje tooltips “Cerrar ventana sin guardar los datos”. El botón Guardar de la interfaz Pruebas a Bases de Datos en MySQL muestra como mensaje tooltips “Guardar prueba”.

	El botón Cancelar de la interfaz Pruebas a Bases de Datos en MySQL muestra como mensaje tooltips “Cancelar acción”. Si existen campos con valores incorrectos se marca en rojo y se muestra un mensaje tooltips de los definidos en el documento Estándar de mensajes para las acciones en dependencia del error. Si un campo no puede ser nulo y se deja vacío se muestra el mensaje: “Este campo es obligatorio”. El campo Dirección de la IU Adicionar datos de la conexión, solo admitirá el formato de número IP 255.255.255.255, si no se cumple dicha condición, el sistema muestra el mensaje: “El formato establecido es 255.255.255.255.”. El campo Puerto de la IU Establecer datos de la conexión, solo admitirá hasta 5 dígitos, si no se cumple dicha condición, el sistema mostrará el mensaje “El tamaño máximo requerido es 5 dígitos”. Los campos usuario y clave solo admitirán hasta 100 caracteres, si no se cumple dicha condición, el sistema muestra el mensaje: “El tamaño máximo requerido es 100 caracteres”.
Post-condiciones	Se ha adicionado la prueba
Post-requisito	No procede.

Figura 7: Prototipo de interfaz de usuario Establecer datos de conexión

Fuente: Elaboración propia

Especificación del requisito: **Realizar revisión de la Base de Datos.**

Conceptos tratados	**Conceptos**	**Atributos**
	Prueba Base de Datos	id denominación
Precondiciones	**Precondiciones**	**Pre-requisito**
	Autenticación del usuario en el rol de Asegurador de la calidad. Debe existir al menos una prueba adicionada.	Autenticar usuario. Adicionar prueba.
Descripción	1. Se selecciona la opción **Inicio/XETID/Calidad/ Prueba de BD.** 2. Se muestra la interfaz Prueba de BD, y se habilita el espacio de trabajo Pruebas a Bases de Datos en MySQL. 3. Se selecciona de la lista de pruebas, la que se le desea realizar la acción. 4. Se presiona el botón Revisar. 5. Se gestionan los errores. 6. Prosigue al caso de uso: Listar errores.	
Complejidad	Alta.	
Validaciones	3.1 La opción Revisar de la barra de herramientas del espacio de trabajo Prueba a Bases de Datos en MySQL se habilitará cuando se seleccione una prueba de la lista. El sistema valida los datos según lo descrito en el documento Modelo conceptual_CEAD MySQL. Si ocurre un error al intentar acceder a la Base de Datos, el sistema muestra el mensaje: "La revisión de la prueba no se pudo realizar debido a	

	<causa>.", mantiene abierta la interfaz Prueba a Bases de Datos en MySQL. Sobre *tooltips*: La opción Revisar del espacio de trabajo Prueba a Bases de Datos en MySQL muestra como mensaje tool-tips “Ejecutar revisión”.
Post-condiciones	Se ha revisado la Prueba
Post-requisito	No procede.

Especificación del requisito: Listar errores

Conceptos tratados	**Conceptos**	**Atributos**
	Prueba Error	Propiedad Violación Estándar Descripción
Precondiciones	**Precondiciones**	**Pre-requisito**
	Autenticación del usuario en el rol de Asegurador de la calidad. Debe haberse realizado una revisión a una prueba.	Autenticar usuario. Realizar revisión.
Descripción	1. Se muestra la interfaz CEAD MySQL, se habilita el espacio de trabajo Prueba a Bases de Datos en MySQL. 2. Se habilita el panel errores encontrados. 3. Se listan los errores generados. Ver figura 2.3.	

Complejidad	Baja.
Validaciones	Si ocurre un error al intentar acceder a la Base de Datos, el sistema muestra el mensaje: "La lista de errores no se pudo generar debido a <causa>." Sobre *tooltips*:
Post-condiciones	Se ha generado una lista con los errores encontrado en la prueba de BD.

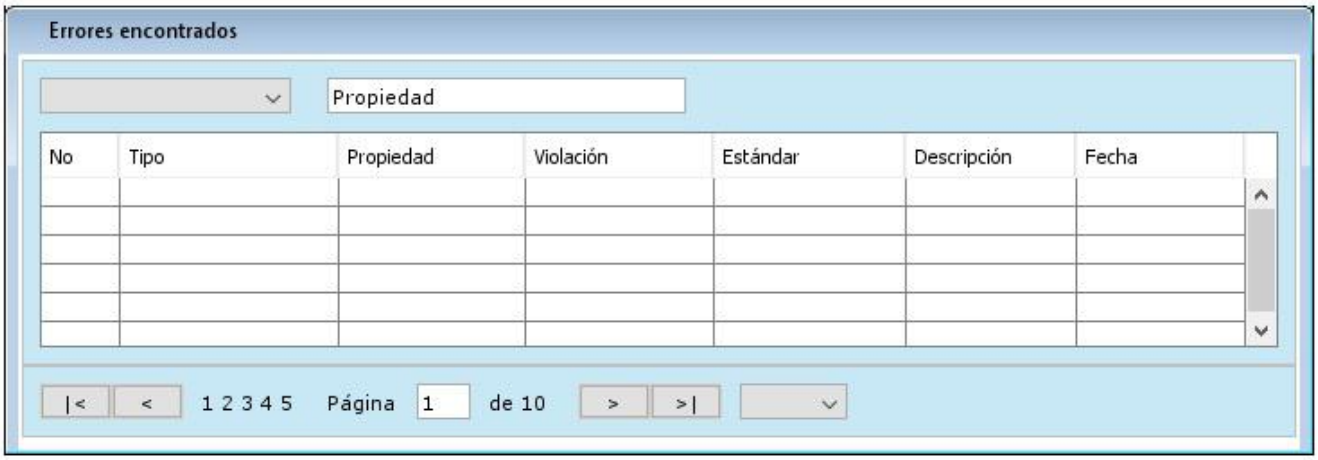


Figura 8: Prototipo de interfaz de usuario Listar errores

Fuente: Elaboración propia

2.6 Requisitos no funcionales del sistema

Los requisitos no funcionales describen otras prestaciones, características y limitaciones que debe tener el sistema para alcanzar el éxito. Los requerimientos no funcionales engloban características como rendimiento, facilidad de uso, presupuestos, tiempo de entrega, documentación, seguridad y auditorías internas. Son restricciones de los servicios o funciones ofrecidos por el sistema (Villar 2017). Como parte de la solución al problema planteado se definen los siguientes Requisitos no funcionales del sistema:

2.6.1 Usabilidad (USB)

RNF01: El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.

RNF02: El sistema hará uso del empleo de teclas calientes y tooltips.

RNF03: El sistema contendrá íconos que representan visualmente la acción para lograr un menor tiempo de aprendizaje.

RNF04: En controles de tipo *button, input, select y text-área*, cambiará el color de fondo indicando que el cursor está situado en él para presionar. Una vez que el cursor deja de estar situado sobre ellos, regresarán a su estado normal.

RNF05: La presentación, composición y formularios del sistema se visualizará sin problemas en Firefox (la versión debe ser mayor de 28).

RNF06: El sistema informa claramente al usuario cuando se produce un error sobre lo ocurrido y de cómo solucionar el problema.

2.6.2 Confiabilidad (CON)

RNF07: Ante cualquier falla en el sistema se deben mostrar los errores sin dar detalles de información que pudiera comprometer la seguridad e integridad del mismo.

RNF08: El sistema deberá mostrar un mensaje de confirmación antes de realizar cualquier acción definitiva o irreversible sobre los datos, como modificación o eliminación.

2.6.3 Rendimiento (REN)

RNF09: Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 10 segundos.

RNF10: El sistema debe de soportar de 20 usuarios concurrentes.

2.6.4 Aplicación de Estándares (EST)

RNF11: Para el desarrollo de la aplicación se seguirán los Estándares de teclas calientes (López 2014d), iconografía (López 2014c), codificación para PHP (Hurtado, Crespo y Terrer 2015), mensajería para las acciones (Rodríguez 2014), diseño de interfaces para las aplicaciones de gestión (López 2014b) y arquitectura de datos (López 2014a).

2.6.5 Interfaz (INU)

RNF12: El sistema debe contar con una interfaz fácil, amigable, sencilla, permitiendo que los usuarios finales del mismo sean capaces de interactuar con este aun teniendo conocimientos básicos.

RNF13: Empleo de imágenes y colores identificados con la XETID.

2.6.6 Portabilidad (POR)

RNF14: El sistema se ejecutará en el Sistema Operativo Linux en cualquiera de sus distribuciones, así como en Windows.

2.6.7 Seguridad (SEG)

RNF15: Las contraseñas introducidas en el sistema serán encriptadas.

RNF16: La información que se maneje en el sistema estará protegida de acceso no autorizado y divulgación, a partir de los diferentes roles de los usuarios que empleen el sistema.

2.6.8 Software (SFT)

RNF17: Para el servidor:

- ✓ Apache v2.2
- ✓ Php v5.3
- ✓ Sistema operativo Debian 7
- ✓ PostgresSQL v9.4

RNF18: Para el cliente:

- ✓ Navegador Firefox v28 en adelante

2.6.9 Hardware (HDW)

RNF19: Podrá ser desplegado en computadoras con un mínimo de 2 núcleos, con microprocesador de 3.0 GHz de velocidad y memoria RAM de 2 GB.

2.7 Arquitectura de software

En una definición tal vez demasiado amplia, David Garlan establece que la Arquitectura de Software constituye un puente entre el requisito y el código, ocupando el lugar que en los modelos antiguos se reservaba para el diseño. Por otra parte, en el documento de IEEE Std 1471-2000 (IEEE 2020), adoptada también por Microsoft, se define la arquitectura como: *… la organización fundamental de un sistema encarnado en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución*.

La Arquitectura enfoca el diseño del software desde varias perspectivas, de la calidad de estos procesos dependerá el éxito del diseño detallado, implementación e integración de la aplicación (Prodesoft 2014b).

2.8 Diseño de la arquitectura del sistema

La Arquitectura de Sistema es una de las disciplinas más complejas dentro de la Arquitectura de Software, responsable de definir correctamente cohesionados, acoplados e interrelacionados los elementos computacionales del producto, las principales interacciones, los conectores y las configuraciones a asumir en función de los elementos del negocio que los mismos abstraen (Prodesoft 2014b).

2.8.1 Patrón arquitectónico

Los patrones arquitectónicos se abocan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica que sirve como base para el diseño de la arquitectura. Los patrones arquitectónicos para el software definen un enfoque específico para el manejo de algunas características del sistema.

Para aplicaciones web los autores sugieren una arquitectura del diseño en tres capas que desacopla la interfaz de la navegación y del comportamiento de la aplicación. Plantean que mantener separadas la interfaz, la aplicación y la navegación, simplifica la implementación y mejora la reutilización.

La arquitectura Modelo Vista Controlador (MVC) es uno de varios modelos sugeridos para la infraestructura de aplicaciones web que desacoplan la interfaz de usuario de sus funciones y contenido informativo (Pressman 2010a).

El diseño del componente está basado en la arquitectura Modelo Vista Controlador (MVC). Este patrón permite separar una aplicación en 3 capas, una forma de organizar y de hacer escalable un proyecto, a continuación, una breve descripción de cada capa.

Modelo: esta capa representa todo lo que tiene que ver con el acceso a datos: guardar, actualizar, obtener datos, además todo el código de la lógica del negocio, básicamente son las clases Java y parte de la lógica de negocio.

Vista: la vista tiene que ver con la presentación de datos del modelo y lo que ve el usuario, por lo general una vista es la representación visual de un modelo.

Controlador: el controlador es el encargado de conectar el modelo con las vistas, funciona como un puente entre la vista y el modelo, el controlador recibe eventos generados por el usuario desde las vistas y se encargar de direccionar al modelo la petición respectiva.(Largo 2016)

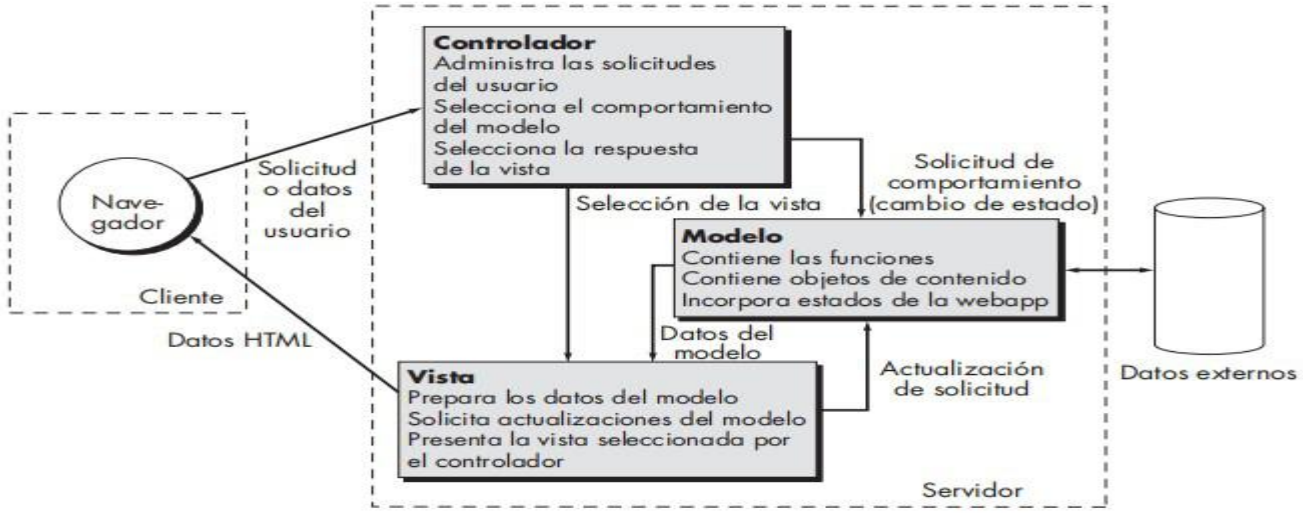


Figura 9: Arquitectura MVC

Fuente: (Pressman 2010a)

2.9 Diagrama de clases del diseño

En el diseño de clases se resume la definición de las clases que se pueden implementar en el software, se visualizan las relaciones entre ellas y se muestra gráficamente la interacción de los objetos para comunicarse entre sí (Prodesoft 2014).

Esta actividad tiene como entrada la especificación de requisitos y el modelo conceptual. A partir de estos artefactos se identifican las clases de diseño. Una vez identificadas, se elabora el Diagrama de clases del diseño que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos (Prodesoft 2014). A continuación, se muestra el diagrama de clases del diseño para el componente a implementar.

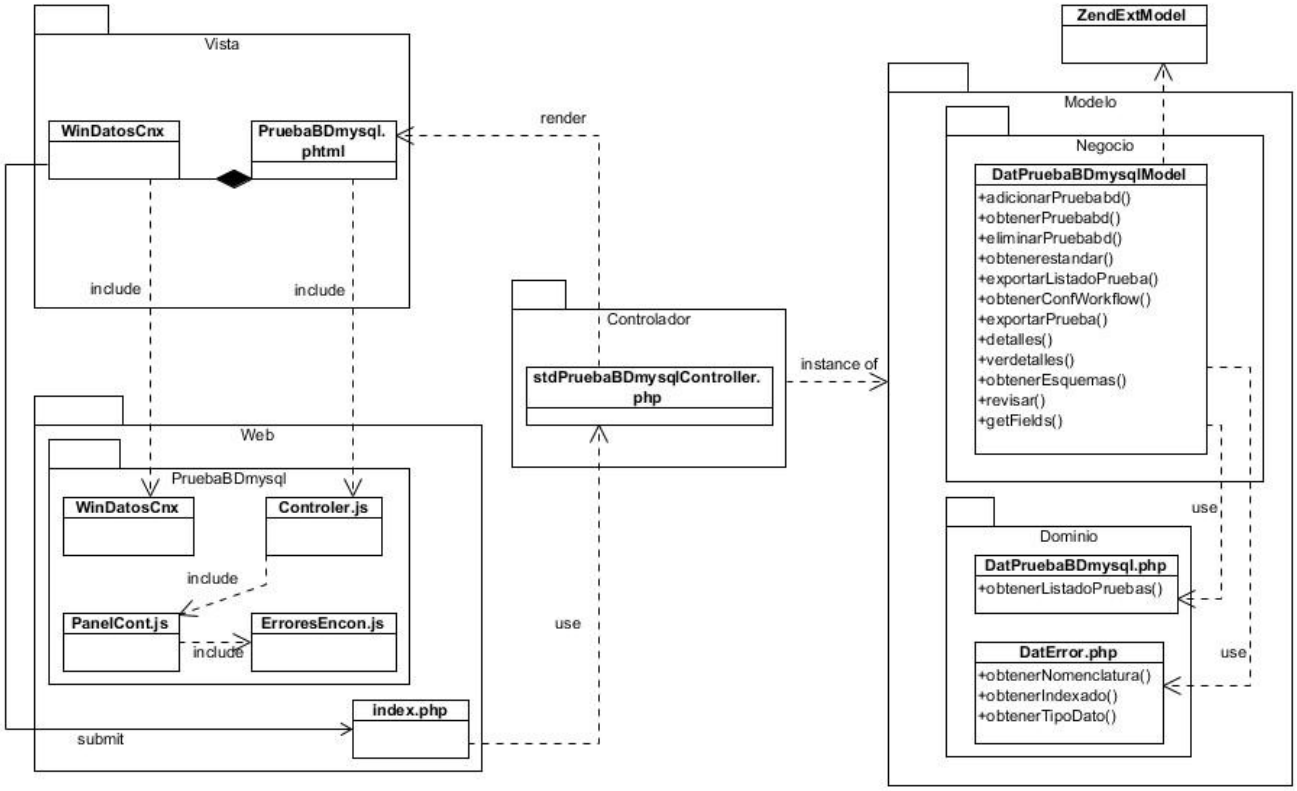


Figura 10: Diagrama de clases del diseño

Fuente: Elaboración propia

2.10 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que se pueden encontrar a la hora de desarrollar una aplicación. Aunque la aplicación sea única, tendrá partes comunes con otras aplicaciones: acceso a datos, creación de objetos u operaciones entre sistemas. Se pueden solucionar problemas utilizando algún patrón, ya que son soluciones probadas y documentadas por multitud de programadores (Huiza 2018).

2.10.1 Patrones GRASP

Los patrones de diseño GRASP (*General Responsibility Assignment Software Patterns*) son unas recomendaciones para asignar responsabilidades a las varias clases de un diseño orientado a objetos. En lugar de proporcionar diseños concretos que resuelven cierto problema, como es el caso de otros patrones de diseño, los patrones GRASP realmente sientan unas bases, más abstractas, que sirven de guía a la hora de repartir en clases concretas las funcionalidades y responsabilidades que se derivan de los requerimientos (Molgó Sendra 2014).

Patrón Experto

Es un criterio para determinar dónde delegar responsabilidades. La idea básica es que, dada una funcionalidad a implementar, se debe determinar cuál es la información necesaria para acometerla, y donde está almacenada. La clase que debe implementar esta funcionalidad es la que contiene la mayor parte de la información requerida (Molgó Sendra 2014).

Este patrón se pone en práctica en la definición de la clase modelo del componente CEAD MySQL, un ejemplo de esto es la clase DatPruebabdmysqlModel la cual tendrá la responsabilidad de gestionar las pruebas, dominando la mayor parte de la información.

Patrón Bajo Acoplamiento

Este criterio está directamente relacionado con el de alta cohesión. Un sistema con bajo acoplamiento es aquel que tiene pocas dependencias entre las clases, y en el cual un cambio en una clase determinada provoca un impacto bajo en el resto. Los sistemas que gozan de bajo acoplamiento son más fáciles de reutilizar y de mantener (Molgó Sendra 2014). Este patrón se tuvo en cuenta para eliminar dependencias innecesarias entre las clases asignándole a cada clase la información necesaria para realizar sus funciones.

Patrón controlador

Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Este patrón ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de la interfaz. Al delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras (Casas De la Torre 2017).

El uso de este patrón se hizo necesario para la clase controladora del componente ya que esta es la responsable de gestionar las funcionalidades del sistema. La arquitectura MVC brinda una capa

específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón.

Patrón Creador

Este patrón proporciona una guía para la decisión de cuál debe ser la clase responsable de crear los objetos de otra. En concreto, según el patrón Creador una clase B debería ser la responsable de crear una clase A si alguna de las siguientes condiciones es cierta:

- Las instancias de B contienen o agregan por composición las instancias de A
- Las instancias de B almacenan instancias de A
- Las instancias de B realizan un uso intensivo de las instancias de A
- Las instancias de B tienen la información necesaria para inicializar las instancias de A

Este patrón se aplica principalmente a la clase controladora, que se encarga de crear objetos de la clase modelo. Por ejemplo, la clase StdpruebabdmysqlControler instancia a la clase DatPruebabdmysqlModel que permite gestionar las pruebas. También se adapta a las clases del paquete Domain, quienes permiten el acceso a la información almacenada a nivel de datos.

Patrón Alta Cohesión

Según el criterio de alta cohesión, las clases deben estar lo más focalizadas posible entorno a un objetivo, y por lo tanto sus responsabilidades deben estar altamente relacionadas y agrupadas. Un sistema con alta cohesión es más legible y mantenible. En cambio, un sistema con baja cohesión, donde un elemento dado tiene demasiadas responsabilidades o donde una responsabilidad concreta está repartida por varios componentes, es difícil de comprender, reutilizar y mantener (Molgó Sendra 2014).

Este patrón se ve evidenciado cuando la clase controladora utiliza su modelo específico. Mediante el cual accede a los datos que necesita para la realización de una funcionalidad en concreto con la información que realmente le hace falta y posteriormente pasársela a la vista. Un ejemplo más detallado se visualiza en la clase StdpruebabdmysqlControler que accede a los datos mediante la clase DatPruebabdmysqlModel

2.10.2 Patrones GOF

Patrones GOF (*Gang of Four patterns*) son un conjunto de una veintena de patrones, agrupados según el tipo: patrones creacionales, estructurales o de comportamiento. A diferencia de los patrones GRASP, estos exponen soluciones óptimas a problemas más concretos (Molgó Sendra 2014).

Patrón Facade (Fachada)

Este patrón tiene la característica de ocultar la complejidad de interactuar con un conjunto de subsistemas proporcionando una interface de alto nivel, la cual se encarga de realizar la comunicación con todos los subsistemas necesarios. La fachada es una buena estrategia cuando se requiere interactuar con varios subsistemas para realizar una operación concreta ya que se necesita tener el conocimiento técnico y funcional para saber qué operaciones de cada subsistema se tiene que ejecutar y en qué orden, lo que puede resultar muy complicado cuando los sistemas empiezan a crecer demasiado (Blancarte 2016). Al crearse una interfaz sencilla, como punto de acceso en común con otros sistemas o subsistemas, se simplifica la complejidad a los clientes externos con una interfaz clara y un acceso unificado de las funcionalidades haciendo más fácil su uso. Todas las funcionalidades en una interfaz principal.

2.11 Diseño

En el desarrollo o ciclo de vida de un sistema informático, el diseño del sistema constituye un elemento fundamental del mismo. El papel del diseño es adquirir conocimiento del funcionamiento del software. Este constituye un punto de partida para comenzar con la implementación.

2.11.1 Diseño de la Base de Datos

Teniendo como entrada el Modelo conceptual, la Especificación de la arquitectura de sistema y la Especificación de los requisitos de software se diseñan las tablas, sus atributos y relaciones agrupados por componentes. Esto se puede realizar ya sea por paquetes o colores delimitando cada uno de ellos, obteniendo como resultado el Modelo de datos (Prodesoft 2014b). Si los requerimientos del software incluyen la necesidad de crear, ampliar o hacer interfaz con una base de datos, o si deben construirse y manipularse estructuras de datos complejas, el equipo del software tal vez elija crear un modelo de datos como parte del modelado general de los requerimientos (Pressman 2010b).

En el diseño de la BD se tiene en cuenta la integración con componentes externos a la solución diseñada (Prodesoft 2014). A continuación, se presenta el modelo de la base de datos diseñado para el componente CEAD MySQL.

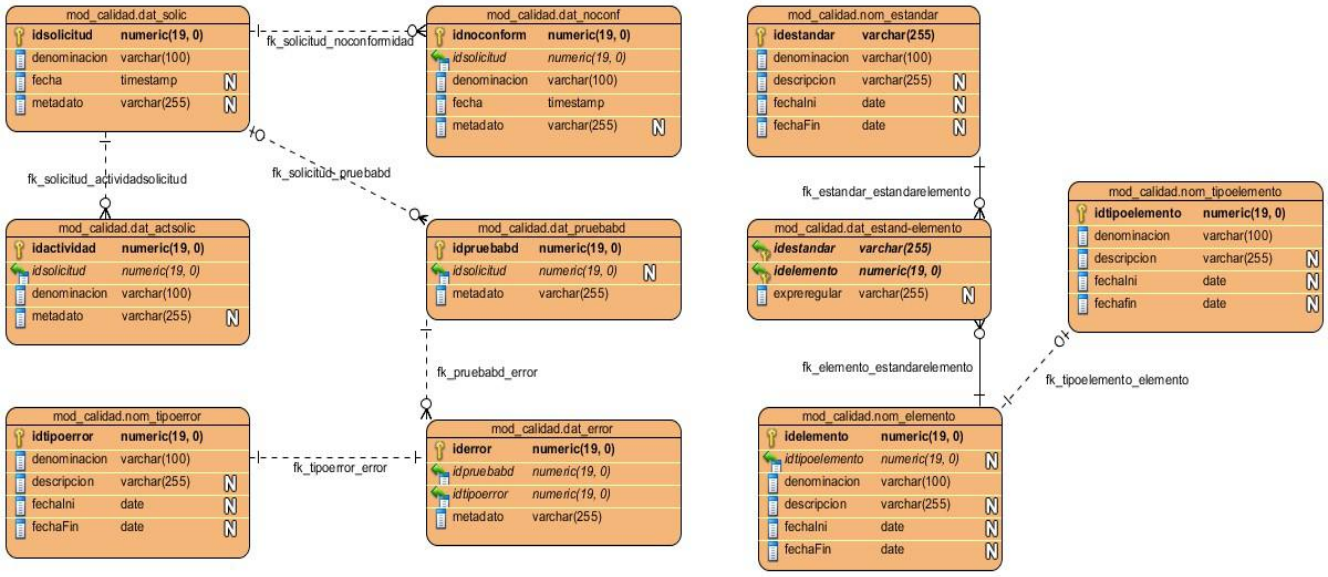


Figura 11: Modelo de datos

Fuente: Elaboración propia

Con este modelo de datos se describe la estructura de la base de datos diseñada para el componente, la cual está integrada por todas las entidades que se identificaron a partir del análisis del modelo conceptual y las especificaciones de requisitos. Además, se muestran los atributos que caracterizan cada una de estas entidades, haciendo posible su identificación única a partir del empleo de llaves primarias y foráneas. De esta misma manera se tuvo en cuenta la integración de la solución propuesta con el componente externo Nomenclador, con sus correspondientes entidades identificadas y atributos.

2.11.2 Diagrama de componentes

De acuerdo al modelo conceptual, la especificación de requisitos, los prototipos de interfaz de usuario definidos anteriormente para el CEAD MySQL, se especificó la integración del sistema con otros componentes, obteniendo el siguiente gráfico de diagrama de componentes externos.

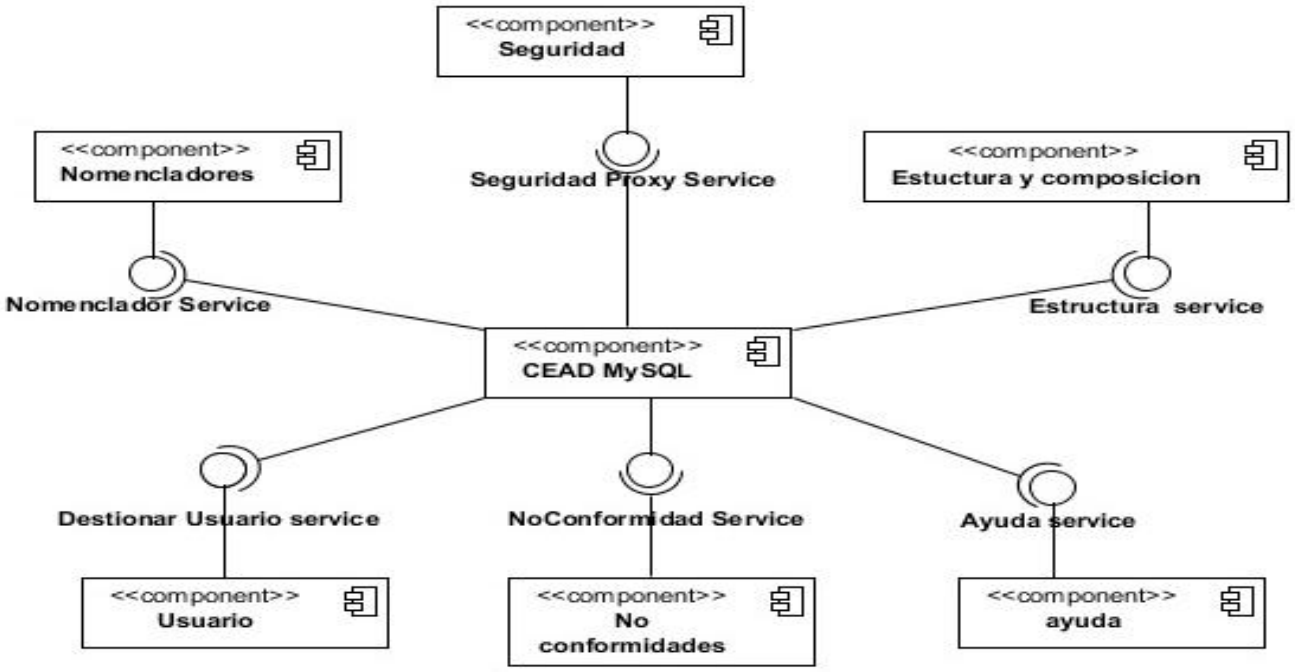


Figura 12: Diagrama de componentes externos

Fuente: Elaboración propia

2.11.3 Diseño del modelo de despliegue

Se modela la distribución física de los nodos de hardware donde pueden ejecutarse los paquetes y componentes identificados. En el modelo se representa como se distribuyen los sistemas en cada nodo físico (Prodesoft 2014b).

A continuación, se muestra el diagrama de despliegue diseñado para el componente:

Figura 13: Diagrama de despliegue

Fuente: Elaboración propia

PC_Cliente: representa el conjunto de computadoras a través de las cuales los usuarios pueden actualizar y consultar la información que se encuentra en el Servidor. Esta computadora debe tener instalado el Sistema Operativo Windows o GNU/ Linux, como navegador Web Internet Explorer o Mozilla Firefox 28.0 como mínimo. La comunicación entre las PC Clientes y el Servidor se establece utilizando el conjunto de protocolos de comunicación HTTP.

Servidor Web: representa el servidor donde se encuentra desarrollada la aplicación web, se utilizará el servidor Web Apache más el lenguaje de programación PHP. Contiene toda la información

referente a la aplicación. Este accede al servidor de Base Datos de PostgreSQL para el manejo de la información mediante el protocolo TCP/IP.

Servidor Base de datos: Representa un servidor de base de datos, en el cual se ubica toda la información persistente del sistema, almacenándose los datos que son creados, actualizados y consultados por el administrador. Se utilizará PostgreSQL como servidor de base de datos.

2.10 Conclusiones del capítulo

En este capítulo se crearon los pilares que permitirán la correcta implementación del componente para la validación de la arquitectura de datos para MySQL, definiendo la propuesta de solución y el modelo conceptual que permitió conocer los conceptos asociados al problema. Se especificaron los requisitos funcionales y no funcionales de la aplicación y su descripción detallada que dieron paso a una mejor comprensión de los resultados que se pretenden obtener y sirvieron de guía para la codificación del sistema. La definición de la arquitectura y los patrones de diseño a utilizar, permitieron establecer las bases para fomentar la reutilización y las buenas prácticas de programación.

Capítulo 3: Implementación y prueba

3.1 Introducción

En este capítulo se presenta todo lo relacionado con la implementación y la validación del componente a desarrollar como solución propuesta al problema planteado desde un inicio. El uso de técnicas y estrategias para la validación de cualquier producto informático desarrollado son necesarias para aumentar el nivel de calidad de estos. Todo esto debido a la posibilidad de encontrar errores al efectuarse una nueva revisión. Partiendo de lo anterior se llevan a cabo una serie de pruebas para garantizar el correcto funcionamiento del componente.

3.2 Implementación

La Implementación comienza con el resultado obtenido del diseño detallado. El objetivo principal de este es desarrollar lo descrito en la arquitectura y el sistema como un todo. De forma más específica, los propósitos de la Implementación son:

- Planificar las integraciones de sistema necesarias en cada iteración. Siguiendo para ello un enfoque incremental.
- Implementar clases, componentes y subsistemas encontrados durante el diseño.
- Integrar componentes.

Durante la Implementación se mantiene como principio la estandarización y reutilización de códigos y componentes (Prodesoft 2014b).

3.2.1 Estándares de codificación

Es una necesidad para una empresa de desarrollo de software tener definidas las pautas para la construcción del código fuente, que garanticen un código robusto, eficaz y que contribuyan en la mayor medida posible al máximo rendimiento de las aplicaciones que conforman. Estas pautas se definen en los estándares de codificación, que tienen como objetivo, establecer estilos de códigos sólidos con vistas de que un proyecto de software se convierta en un producto fácil de comprender y de mantener (Hurtado, Crespo y Terrer 2015).

En la implementación de la propuesta de solución se utilizaron los estándares definidos en el documento (Estándar de codificación para PHP Versión 2.0), elaborado por el Comité de rol de Lógica de Negocio de la XETID. Entre los estándares explicados se encuentran los relacionados con:

- ✓ Formato de los archivos PHP.
- ✓ Convenciones de nombre.
- ✓ Estilo de código.

3.3 Prueba

La prueba es el proceso que ejecuta un programa con objeto de encontrar un error. Un buen caso de prueba es el que tiene alta probabilidad de encontrar un error que no se ha detectado hasta el momento. La prueba demuestra que las funciones de software parecen funcionar de acuerdo con las especificaciones, y que los requerimientos de comportamiento y desempeño aparentemente se cumplen. Además, los datos obtenidos según se realiza la prueba dan una buena indicación de la confiabilidad del software y ciertas indicaciones de la calidad de éste como un todo (Pressman 2010a).

Las pruebas del software son la actividad más común de control de la calidad realizada en los proyectos para asegurar el correcto funcionamiento del software. Tienen como objetivos la verificación de la correcta implementación de los requisitos explícitamente establecidos, la adecuada integración de los componentes que conforman el sistema y la ejecución de casos de prueba que permitan detectar el mayor número de No conformidades y corregirlas antes de la entrega del software al cliente (Prodesoft 2014).

Es importante destacar que las pruebas reducen la probabilidad de que aparezcan defectos ocultos en el software, pero incluso si no se encuentra ningún defecto, nunca será una garantía de su corrección (Prodesoft 2014).

3.3.1 Técnicas de prueba

Las técnicas de evaluación dinámicas permiten obtener diferentes criterios para generar casos de pruebas que provoquen errores en el sistema que se desee probar.

Pruebas de caja negra: requiere la introducción de procesamiento previo y posterior en la interfaz del componente para eliminar o anular los conflictos (Pressman 2010a). En otras palabras, las pruebas de caja negra buscan verificar que la relación entre las entradas y las salidas sean correctas. Se centran en los requisitos funcionales y se desarrollan sobre la interfaz del software.

Pruebas de caja blanca: estudia los detalles del procesamiento interno del componente y hace modificaciones en el nivel de código para eliminar cualquier conflicto (Pressman 2010a). Después de estudiar varios conceptos se puede decir que las pruebas de caja blanca están dirigidas a las funciones internas de un componente, es decir que se centran en buscar errores en el código fuente.

3.3.2 Estrategias de pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar

la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto. Una estrategia para la prueba de software debe incluir pruebas de bajo nivel, que son necesarias para verificar que un pequeño segmento de código fuente se implementó correctamente, así como pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requerimientos del cliente (Pressman 2010b).

Una estrategia para probar el software también se puede ver en el contexto de espiral como se muestra en la siguiente figura, donde se pueden observar los niveles de pruebas que conducen al análisis de diferentes requerimientos del sistema.

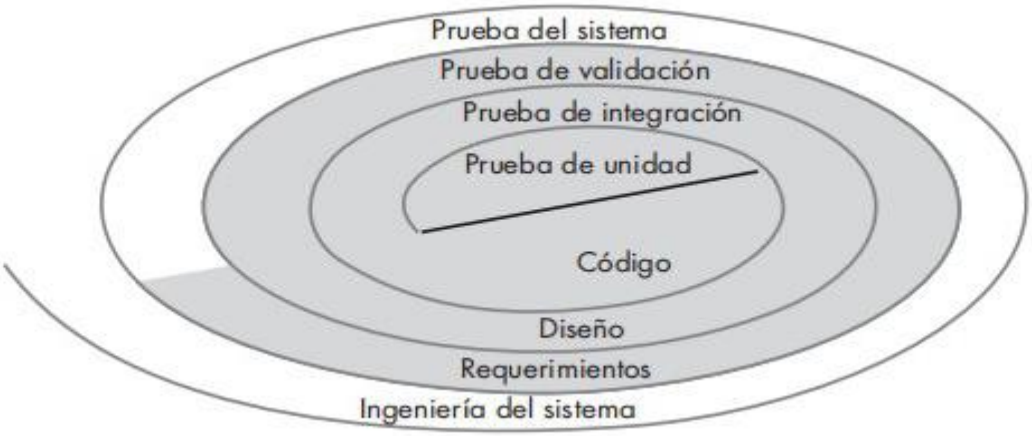


Figura 14: Estrategia de pruebas

Fuente: (Pressman 2010b)

Niveles de prueba

Pruebas unitarias: están basadas en la funcionalidad de los componentes y permiten asegurar que cada uno de estos funciona correctamente por separado. En ellas, los errores están más acotados y son más fáciles de localizar, lo que facilita a los desarrolladores la solución de los mismos permitiendo llegar a la integración con un mayor grado de seguridad del correcto funcionamiento de cada parte del sistema por separado. Para ejecutar las pruebas de unidad se seleccionan las clases de pruebas que fueron diseñadas para cada funcionalidad de forma independiente (Prodesoft 2014b). En este nivel se realizarán estas pruebas con la ayuda de la herramienta RIPS.

Pruebas de integración: son una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es

tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño (Pressman 2010b).

Pruebas del sistema: donde se va a comprobar si el producto cumple con los requisitos especificados. Pueden incluir pruebas basadas en riesgos y/o especificaciones de requisitos, procesos de negocio, casos de uso u otras descripciones de texto de alto nivel o modelos de comportamiento de sistema, interacciones con el sistema operativo y recursos del sistema. Las pruebas de sistema deben de estudiar los requisitos funcionales y no funcionales del sistema y las características de calidad. Para ello se aplicarán técnicas de prueba de caja negra (Sánchez 2015). En este nivel se realizarán pruebas de funcionalidad a partir del diseño de diferentes casos de pruebas, pruebas de rendimiento con la herramienta JMeter y seguridad a partir del empleo de la herramienta RIPS.

3.3.3 Diseño de casos de prueba (DCP)

Los casos de prueba ayudan a validar que el sistema desarrollado realice las funciones para las que ha sido creado en base a los requisitos del usuario, por lo que resulta importante diseñar al menos un caso de prueba para cada requisito del sistema con el objetivo de asegurar que todas las funcionalidades sean comprobadas (Prodesoft 2014).

A partir del análisis de conceptos de caso de prueba de diferentes autores, se entiende por este que es un conjunto de condiciones o variables bajo las cuales se determinara si una aplicación, un sistema de software o una característica de estos es parcial o completamente satisfactoria.

- **Casos de prueba para la Prueba unitaria**

Se realizó la prueba unitaria mediante la utilización de la herramienta RIPS, que permite realizar esta automatización del proceso de identificación de potenciales funciones vulnerables en el código fuente de aplicaciones PHP, mediante su análisis estático (Gutierrez 2017). Como se puede observar en la siguiente figura, al utilizar la herramienta RIPS no se encontraron vulnerabilidades que pudieran afectar el correcto funcionamiento del componente.

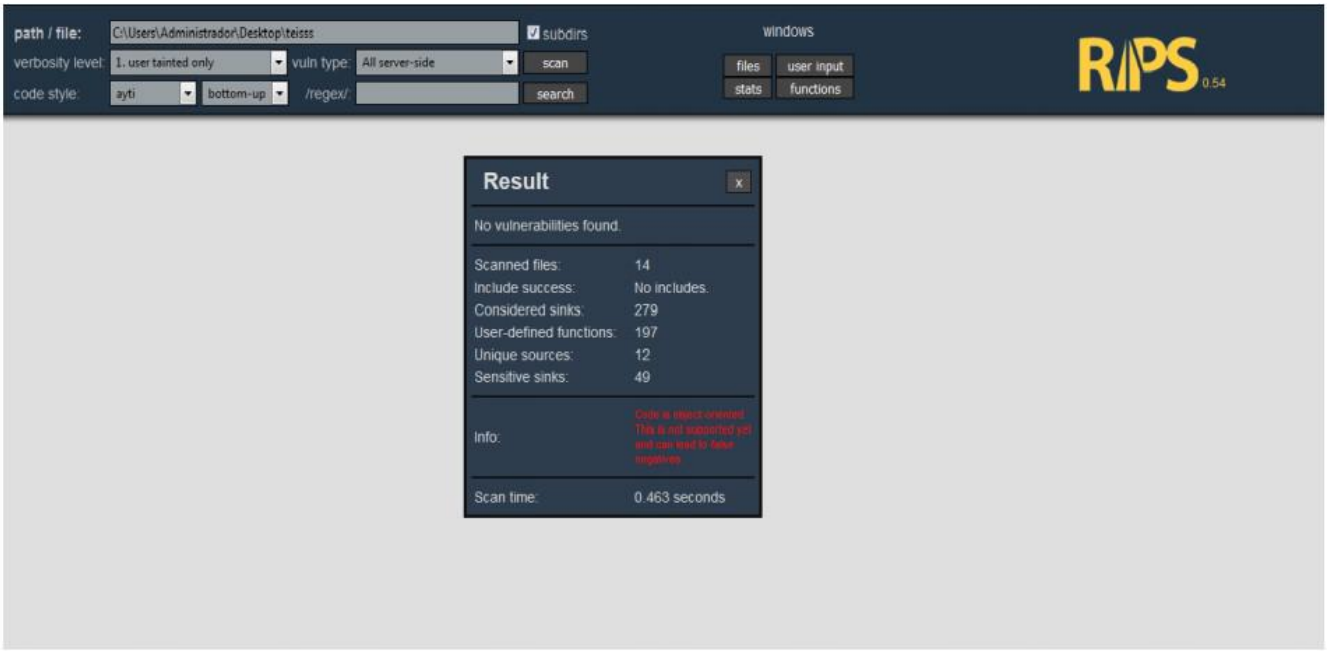


Figura 15: Resultado de la prueba unitaria con la herramienta RIPS

- **Casos de prueba para la Prueba de seguridad**

Pruebas de seguridad: validan los servicios de seguridad de una aplicación e identifican posibles fallos y debilidades. Muchos proyectos utilizan un enfoque de caja negra para las pruebas de seguridad, lo que permite a los expertos, sin conocimiento del software, probar la aplicación en busca de agujeros, fallos, exploit y debilidades (Quijano 2018). Se realizó la prueba de seguridad utilizando la herramienta RIPS. Esta se encarga de revisar varias violaciones de seguridad tales como:

- ✓ La ejecución de código.
- ✓ La ejecución de comandos.
- ✓ Inyección de cabecera.
- ✓ Divulgación de archivos.
- ✓ Inclusión de archivos.
- ✓ La manipulación de archivos.
- ✓ Inyección SQL. (Reyes Pérez 2016a).

Luego de efectuar la prueba exploratoria con la herramienta se obtuvieron los siguientes resultados, donde no se encontraron violaciones de seguridad que pudieran afectar la calidad del componente, como se puede observar en la siguiente figura:

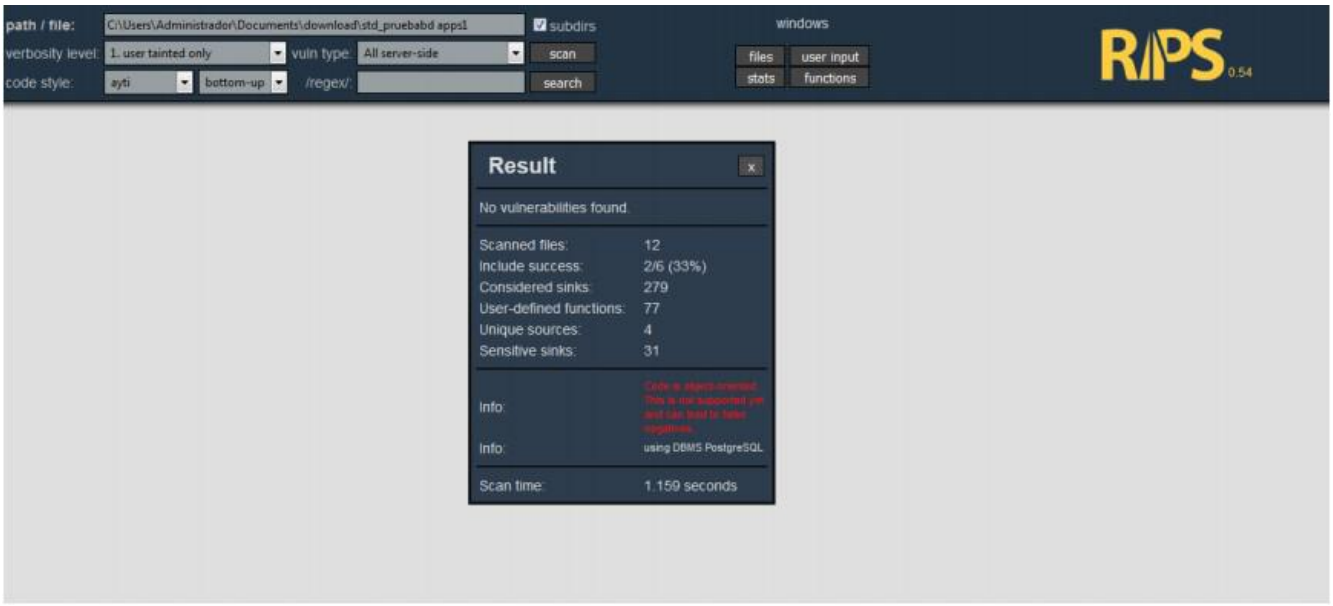


Figura 16: Resultado de la prueba de seguridad con la herramienta RIPS

- **Casos de prueba para la Prueba de rendimiento**

Pruebas de rendimiento: determinan la capacidad de respuesta, el rendimiento, la confiabilidad y/o la escalabilidad de un sistema bajo una carga de trabajo determinada. En aplicaciones web, las pruebas de rendimiento a menudo están estrechamente relacionadas con las pruebas de estrés, la medición del retraso y la capacidad de respuesta bajo una carga pesada (Quijano 2018). Se realizó la prueba de rendimiento haciendo uso de la herramienta JMeter. Esta herramienta es una de las más utilizadas para realizar pruebas de rendimiento pues permite probar la carga masiva de los datos y realizar pruebas de esfuerzo a los servidores (Claudia 2018b).

Para este tipo de prueba se observaron los resultados del JMeter en la funcionalidad del componente Revisar BD. Como se puede ver a continuación:

Funcionalidad Revisar BD

Etiqueta	#Muestras	Media	Mediana	Línea de 90%	Mín.	Máx.	%Error	Rendimiento	Kb/sec
/std/std_ba...	30	6317	7559	9821	2231	10387	0.00%	2.7/sec	1.2
/std/std_ba...	30	820	875	1176	1	1331	3.33%	3.3/sec	31.0
/std/std_ba...	30	828	980	1216	149	1268	0.00%	3.2/sec	6.0
/std/std_ba...	30	797	909	1210	167	1289	0.00%	3.0/sec	5.6
/std/std_ba...	30	774	823	1213	148	1324	0.00%	2.9/sec	6.1
/std/std_ba...	150	1907	1007	7559	1	10387	0.67%	10.4/sec	32.4

Después de interpretar los resultados obtenidos, para un total de 30 usuarios usando simultáneamente esta funcionalidad y con un tiempo medio de respuesta del sistema de 1.907 segundos se llega a la conclusión de que el tiempo de respuesta es satisfactorio según el Manual de Usuario del JMeter para las pruebas de rendimiento a sistemas web. Sin embargo, el sistema comienza a dar fallos a partir de 40 usuarios conectados simultáneamente, al realizarse otra iteración de esta prueba.

Luego de realizar la prueba de rendimiento en su totalidad se llega a la conclusión de que el sistema cumple con los requisitos de rendimiento al estresarse después de 40 usuarios conectados simultáneamente y con un tiempo de respuesta a las peticiones del usuario satisfactorio siendo este menos de 10 segundos.

- **Casos de prueba para la Prueba de funcionalidad**

Pruebas de funcionalidad: pruebas automatizadas o manuales que prueban las funcionalidades de la aplicación o módulo construidos desde el punto de vista del usuario final, con sus diferentes roles, para validar que el software hace lo que debe y, sobre todo, lo que se ha especificado. En el caso de las manuales, las ejecuta un tester como si fuese un usuario, pero siguiendo una serie de pasos establecidos en el plan de pruebas, diseñado en el análisis de los requisitos para garantizar que hace lo que debe (casos positivos), que no falla (casos negativos) y que es lo que se ha solicitado. El tester realizará las acciones indicadas en cada paso del caso de prueba comprobando que se cumple el resultado esperado. Si el resultado es distinto, se reportará un defecto con todo detalle: descripción, datos utilizados, capturas de pantalla, para facilitar la solución (Quijano 2018).

DCP Adicionar prueba

Id del escenario	Escenario	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7	Variable 8	Variable 9	Respuesta del sistema
		Dirección	Puerto	Usuario	Clave	Nombre BD	Estándar	Elementos	Estado	Esquema	
EP 1.1	Adicionar prueba correctamente	V (10.12.170. 216)	V(5432)	V(avl)	V(avl)	V(prueba)	V(XETIDv 1)	V(Nomenc latura, Tipo de Datos, Indexado)	V(Registra do)	V(mod_per sona)	El sistema muestra el siguiente mensaje de confirmaci ó n: " La prueba ha sido adicionada satisfactoriamente.".
EP 1.2	Adicionar prueba con datos de conexi ó n erróneos	V(10.12.13. 14)	V(5432)	V(avl)	V(avl)	V(prueba)	V(XETIDv 1)	V(Nomenc latura, Tipo de	V(Registra do)	V(mod_per sona)	El sistema muestra el siguiente mensaje de error: "Los datos de conexión son incorrectos.".
		V (10.12.170. 216)	V(363)	V(avl)	V(avl)	V(prueba)	V(XETIDv 1)	V(Nomenc latura, Tipo de	V(Registra do)	V(mod_per sona)	
		V(10.12.13. 14)	V(5432)	V(usuario no registrado en la BD)	V(avl)	V(prueba)	V(XETIDv 1)	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	
		V(10.12.13. 14)	V(5432)	V(avl)	V(rrr)	V(prueba)	V(XETIDv 1)	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	
		V(10.12.13. 14)	V(5432)	V(avl)	V(avl)	V(nombre de la BD que no existe)	V(XETIDv 1)	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	
EP 1.3	Adicionar prueba	I(jncjidauc)	V(5432)	V(avl)	V(avl)	V(prueba)	V(XETIDv1	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	El sistema muestra el siguiente mensaje de error: "El formato establecido es 255.255.255.".
		V(10.12.170	I(1256355)	V(avl)	V(avl)	V(prueba)	V(XETIDv1	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	El sistema muestra el siguiente mensaje de error: "El Tamaño requerido es de 5 dígitos.".
		V(10.12.170.	V(5432)	I(nasuca scu	V(avl)	V(prueba)	V(XETIDv1	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	El sistema muestra el siguiente mensaje de error: " El tama ñ o m á ximo requerido es de 100
EP 1.4	Cancelar operación	V(10.12.170.	V(5432)	V(avl)	V(avl)	V(prueba)	V(XETIDv1	V(Nomenc latura, Tipo de Datos,	V(Registra do)	V(mod_per sona)	El sistema anula las acciones realizadas y abandona la interfaz **Adicionar Prueba**.

Al realizarse las pruebas de funcionalidad se obtuvieron 6 No conformidades durante la primera iteración. Para la segunda iteración ya solo se obtuvo una No conformidad y en la tercera iteración ya no existía ninguna.

3.4 Conclusiones del capítulo

En este capítulo de describió el proceso de implementación y prueba durante el desarrollo del componente de validación del Estándar de arquitectura de datos en MySQL. Se mostró el diagrama de despliegue dando a conocer así la arquitectura del componente desde el punto de vista de los artefactos del software. La utilización del Estándar de codificación para PHP contribuyó en la mayor medida posible al máximo rendimiento del componente al establecer estilos de código sólidos. Así mismo, el proceso de prueba y los casos de usos elaborados permitieron verificar que el componente desarrollado satisface las necesidades del cliente.

Conclusiones generales

Una vez estudiados los conceptos asociados al dominio de la investigación, que permitieron esclarecer el problema al cual se estaba tratando de dar solución y las diferentes herramientas que pudieran darle solución al problema, las cuales no permitían la validación de un estándar definido en la nomenclatura de una base de datos en MySQL, se obtienen los siguientes resultados relevantes:

- ✓ Los sistemas existentes actualmente no enriquecen el proceso de revisión de base de datos que se lleva a cabo en el CCES de la XETID, debido a que no contribuyen con la revisión de bases de datos desarrolladas en todos los SGBD como MySQL.
- ✓ El CEAD MySQL permite validar de manera automática que la nomenclatura de la arquitectura de datos de las bases de datos en MySQL cumpla con el Estándar de arquitectura de datos utilizado en la XETID.
- ✓ Este componente elimina la manera antigua de realizar pruebas a bases de datos de forma manual para los trabajadores del CCES, disminuyendo el tiempo de trabajo empleado en la realización de las pruebas.
- ✓ Se hace menor el margen de errores cometidos luego de utilizar la herramienta por su capacidad de obtener no conformidades de manera mucho más rápida y confiable a la anteriormente utilizada.
- ✓ Garantiza a la empresa la calidad de los productos que se ofrecen en el mercado pues valida que se cumpla con el Estándar de arquitectura de datos de la XETID, que permite estandarizar el desarrollo, generalizar las herramientas, nomenclaturas y otros elementos de las bases de datos. Además, asegura la legibilidad del modelo de datos y facilita la portabilidad entre motores de bases de datos, plataformas, aplicaciones y la tarea de los programadores en el desarrollo de sistemas.

De esta manera se ha cumplido con el objetivo principal de la investigación. Siendo el componente de validación del Estándar de arquitectura de datos en MySQL, una herramienta que cumple con las necesidades del cliente y satisface los requerimientos del problema planteado.

Referencias bibliográficas:

- ABIZTAR, 2018. *UML- El Modelo Conceptual* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://www.youtube.com/watch?v=cxVRO6iGLHs.
- AMOEDO, D., 2018. JMeter, realiza pruebas de carga y mide el rendimiento desde Ubuntu. *Ubunlog* [en línea]. [Consulta: 2 marzo 2020]. Disponible en: https://ubunlog.com/jmeter-pruebas-carga-mide-rendimiento/.
- APONTE, M., CABRERA, R., CORTEZ, R. y HERNÁNDEZ, Y., 2015. Gestión de la Calidad. [en línea]. [Consulta: 9 diciembre 2019]. Disponible en: https://maponteCursos.files.wordpress.com/2016/05/maponteGestionCalidaduneg.pdf.
- AYALA, Y., 2014. *Estándar para la modelación de negocio y gestión de requisitos*. 2014. S.l.: s.n.
- BLANCARTE, O., 2016. Introducción a los patrones de diseño - Un enfoque práctico. *Introducción a los patrones de diseño* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://reactiveprogramming.io.
- BORGES, S., 2019a. ¿Qué es PostgreSQL? - Para qué sirve, Características e Instalación. *Infranetworking* [en línea]. [Consulta: 24 febrero 2020]. Disponible en: https://blog.infranetworking.com/servidor-postgresql/.
- BORGES, S., 2019b. Servidor Web: ¿Qué es? ¿Para qué sirve? *Infranetworking* [en línea]. [Consulta: 23 marzo 2020]. Disponible en: https://blog.infranetworking.com/servidor-web/.
- CALENDAMAIA, 2014. NetBeans. *Genbeta* [en línea]. [Consulta: 25 febrero 2020]. Disponible en: https://www.genbeta.com/desarrollo/netbeans-1.
- CASAS DE LA TORRE, F.A., 2017. Patrones GRASP. [en línea]. Software. S.l. [Consulta: 19 febrero 2020]. Disponible en: https://es.slideshare.net/Indiana_1969/patrones-grasp-76283581.
- CLAUDIA, 2018a. Las mejores herramientas para pruebas en 2018: ¿Cuál usaremos? *Pandora FMS - The Monitoring Blog* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://pandorafms.com/blog/es/herramientas-para-pruebas/.
- CLAUDIA, 2018b. Las mejores herramientas para pruebas en 2018: ¿Cuál usaremos? *Pandora FMS - The Monitoring Blog* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://pandorafms.com/blog/es/herramientas-para-pruebas/.
- COGNODATA, cognodata, 2019. Arquitectura de datos: la base de una estrategia diferenciadora. *cognodata* [en línea]. [Consulta: 26 noviembre 2019]. Disponible en: https://www.cognodata.com/blog/arquitectura-datos-estrategia-diferenciadora/.
- CONSTANZO, M.A. y CASAS, S.I., 2018. Usabilidad de framework web: identificación de problemas y propuesta de evaluación. *XXIV Congreso Argentino de Ciencias de la Computación (La Plata, 2018).* [en línea]. S.l.: s.n., [Consulta: 25 febrero 2020]. ISBN 978-950-658-472-6. Disponible en: http://sedici.unlp.edu.ar/handle/10915/73289.
- CUEVA LOVELLE, J.M., 1999. Calidad del Software - PDF Free Download. [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://docplayer.es/21137600-Calidad-del-software.html.

- DAMELE, B. y STAMPAR, M., 2020. sqlmap: automatic SQL injection and database takeover tool. [en línea]. [Consulta: 2 marzo 2020]. Disponible en: http://sqlmap.org/.
- DE LEÓN, Á., 2019. ¿Qué es el Servidor Apache? ¿Para qué sirve? Guía de Instalación fácil. *Infranetworking* [en línea]. [Consulta: 25 febrero 2020]. Disponible en: https://blog.infranetworking.com/que-es-apache-servidor/.
- ESPINOZA CALLO, C., 2016a. La Importancia de la Arquitectura de Datos. [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://www.linkedin.com/pulse/la-importancia-de-arquitectura-datos-c%C3%A9sar-espinoza-callo.
- ESPINOZA CALLO, C., 2016b. La Importancia de la Arquitectura de Datos. [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://www.linkedin.com/pulse/la-importancia-de-arquitectura-datos-c%C3%A9sar-espinoza-callo.
- GALEANO, L., 2016. VISUAL PARADIGM. *prezi.com* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://prezi.com/j84ywfydzvit/visual-paradigm/.
- GARCÍA, 2019. Los lenguajes de programación y su orden lógico de aprendizaje. [en línea]. [Consulta: 23 marzo 2020]. Disponible en: https://codelearn.es/blog/los-lenguajes-de-programacion-y-su-orden-logico-de-aprendizaje/.
- GARCIA, M., 2015. ExtJS - Presentacion del framework. *Recursos para formacion* [en línea]. [Consulta: 25 febrero 2020]. Disponible en: https://recursosformacion.com/wordpress/2015/06/extjs-presentacion-del-framework/.
- GUTIERREZ, Y., 2017. *Componente para la Integración de la Biblioteca Virtual y el Gestor Documental de la Empresa de Tecnologías de la Información para la Defensa (XETID)*. S.l.: Universidad de las Ciencias Informáticas.
- HERNÁNDEZ PÉREZ, L.E., 2017. Visual Paradigm para UML es una herramienta para desarrollo. *prezi.com* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://prezi.com/3v2w1mpsptxx/visual-paradigm-para-uml-es-una-herramienta-para-desarrollo/.
- HUIZA, E., 2018. La importancia de los patrones de diseño. *RootNite Blog* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://blog.rootnite.com/opinion/la-importancia-de-los-patrones-de-diseno/.
- HURTADO, E., CRESPO, R. y TERRER, R., 2015. *Estándar de codificación para PHP*. 2015. S.l.: s.n.
- IBERMÁTICA, 2017. XML, la solución ante la diversidad de formatos. *Iberdok* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: http://iberdok.com/2017/06/22/xml-la-solucion-ante-la-diversidad-de-formatos/.
- IEEE, 2020. 1471-2000 - IEEE Recommended Practice for Architectural Description for Software-Intensive Systems. [en línea]. [Consulta: 9 julio 2020]. Disponible en: https://standards.ieee.org/standard/1471-2000.html.
- KHEPRI, W., 2018. Las 25 mejores herramientas de Kali Linux. *Medium* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://medium.com/@williamkhepri/las-25-mejores-herramientas-de-kali-linux-7b0048226ec.

- LARGO, E., 2016. *Patrones de diseño en Java: MVC, DAO y DTO* [en línea]. 2016. S.l.: s.n. Disponible en: https://www.ecodeup.com/patrones-de-diseno-en-java-mvc-dao-y-dto/.
- LÓPEZ, A., 2014a. *Estándar de arquitectura de datos*. 2014. S.l.: s.n.
- LÓPEZ, A., 2014b. *Estándar de diseño de interfaces para las aplicaciones de gestión*. 2014. S.l.: s.n.
- LÓPEZ, A., 2014c. *Estándar de iconografía*. 2014. S.l.: s.n.
- LÓPEZ, A., 2014d. *Estándar de teclas calientes*. 2014. S.l.: s.n.
- MARÍN, R., 2019. Los gestores de bases de datos (SGBD) más usados. *Canal Informática y TICS* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/.
- MOLGÓ SENDRA, jordi, 2014. *Ingenieria en Informatica* [en línea]. 2014. S.l.: s.n. Disponible en: https://repositori.udl.cat/bitstream/handle/10459.1/47600/jmolgos.pdf?sequence=1&isAllowed=y.
- MORALES, N., 2016. Ventajas y Desventajas de PHP. *Taringa!* [en línea]. [Consulta: 24 febrero 2020]. Disponible en: http://localhost:3000/+info/ventajas-y-desventajas-de-php_hqubb.
- MORENO PÉREZ, J.C., 2018. *Entornos de desarrollo* [en línea]. S.l.: Sintesis. Disponible en: https://www.sintesis.com/data/indices/9788491711612.pdf.
- MOZILLA, 2019. Plugins | MDN. [en línea]. [Consulta: 9 julio 2020]. Disponible en: https://developer.mozilla.org/es/docs/Plugins.
- PÉREZ, J.A. y ALCÁNTARA, G., 2014. *Guía para realizar el modelado en el Visual Paradigm utilizando lenguaje Sysml*. 2014. S.l.: s.n.
- POWERDATA, 2014. Introducción a la Calidad de Datos: Definición, Control y Beneficios. [en línea]. [Consulta: 21 agosto 2020]. Disponible en: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/368784/introducci-n-a-la-calidad-de-datos-definici-n-control-y-beneficios.
- PRESSMAN, R.S., 2010a. *Ingeniería de Software. Un enfoque práctico* [en línea]. 7 ma. S.l.: s.n. Disponible en: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.
- PRESSMAN, R.S., 2010b. *Ingeniería de Software. Un enfoque práctico* [en línea]. 7 ma. S.l.: s.n. Disponible en: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.
- PRODESOFT, X., 2014a. *Proceso de desarrollo de software*. 2014. S.l.: s.n.
- PRODESOFT, X., 2014b. *Proceso de desarrollo de software*. 2014. S.l.: s.n.
- QUIJANO, J., 2018. ¿Qué pruebas debemos hacerle a nuestro software y para qué? *Genbeta* [en línea]. [Consulta: 11 julio 2020]. Disponible en: https://www.genbeta.com/desarrollo/que-pruebas-debemos-hacerle-a-nuestro-software-y-para-que.
- RAWAT, 2016. ashish9342/FreeCodeCamp. *GitHub* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://github.com/ashish9342/FreeCodeCamp.

- REYES PÉREZ, J.M., 2016a. *CEAD. Componente de revision de arquitectura de datos de la Empresa de Tecnologia de la Informacion*. S.l.: Universidad de las Ciencias Informáticas.
- REYES PÉREZ, J.M., 2016b. *Trabajo de Diploma*. 2016. S.l.: s.n.
- ROBLEDANO, Á., 2019. qué es Javascript. *OpenWebinars.net* [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://openwebinars.net/blog/que-es-javascript/.
- RODRÍGUEZ, A., 2014. *Estándar de mensajería para las acciones*. 2014. S.l.: s.n.
- SÁNCHEZ, J.M., 2015. *Pruebas de Software. Fundamentos y Técnicas* [en línea]. 2015. S.l.: s.n. Disponible en: http://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf.
- UNAD, 2018. Beneficios de Esta Tecnología | LENGUAJE DE MODELADO UNIFICADO UML. [en línea]. [Consulta: 19 febrero 2020]. Disponible en: http://stadium.unad.edu.co/ovas/10596_9839/beneficios_de_esta_tecnologa.html.
- UNIVERSIDAD DE ALICANTE, 2014. Pruebas con DBUnit. [en línea]. [Consulta: 18 febrero 2020]. Disponible en: http://www.jtech.ua.es/j2ee/publico/lja-2012-13/sesion08-apuntes.html.
- UNIWEBSIDAD, 2020. 8.3. Tooltip (Introducción a JavaScript). [en línea]. [Consulta: 23 marzo 2020]. Disponible en: https://uniwebsidad.com/libros/javascript/capitulo-8/tooltip.
- VILLAR, 2017. Requerimientos Funcionales y No Funcionales. *Porqueria* [en línea]. [Consulta: 18 febrero 2020]. Disponible en: https://ingenieriadesoftwareutmachala.wordpress.com/2017/01/20/requerimientos-funcionales-y-no-funcionales/.

Glosario de términos

Teclas calientes (*hot keys*): Teclas o combinaciones de estas, que al presionarse indican a la aplicación que ejecute un flujo de acciones. Garantizan rapidez y se convierten en un comodín para usuarios especializados en el uso de la aplicación (López 2014d).

Tooltips: son pequeños recuadros de información que aparecen al posicionar el ratón sobre un elemento. Normalmente se utilizan para ofrecer información adicional sobre el elemento seleccionado o para mostrar al usuario pequeños mensajes de ayuda (Uniwebsidad 2020).

Plugins: son pequeños fragmentos de software que interactúan con el navegador para proporcionar algunas funciones que en la mayoría de los casos son muy específicas. Ejemplos típicos de plugins son los usados para mostrar los distintos formatos gráficos, o para reproducir ficheros multimedia. Los plugins son ligeramente diferentes de las Extensiones, que modifican o se añaden a funcionalidades ya existentes (Mozilla 2019).