Universidad de las Ciencias Informáticas

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

Título:

Componente de revisión de estándar de arquitectura de datos para el gestor SQLite

Autora: Sandra Lima Torres.

Tutores:

Ms. C Eduardo Luis Hernández Fernández

Ing. Mauricio Guanche Cañizares

Ciudad de la Habana, 2020

DECLARACIÓN DE AUTORÍA:

Declaro ser autor de la presente tesis y reconocemos 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 junio del año 2020.

Sandra Lima Torres

_____

Firma del Autor

Mauricio Guanche Cañizares

_____

Firma de Tutor

Agradecimientos.

A mi mamá por la dedicación y el esfuerzo, por mantenerse fuerte para sus hijos cuando su mundo se cae.

A mi papá, por todo lo que me enseñó y me quiso mientras estuvo, porque sé que estaría orgulloso.

A mi abuela Miriam, más que abuela madre, por tanto, tanto, tanto.

A mis hermanos, mi príncipe Marcos, mi siempre niña linda Amanda, y mi casi gemela Rachel.

A mi tío Migue y mi abuelo Jesús, por coronarme reina, por convencerme de que yo puedo con todo.

A mi familión d Playa, por adoptarme y enseñarme que rendirme no era una opción.

A mi familia de Cruces, orgullosos de la Lima que sobrevivió a la UCI.

A mi novio Michel, por tu amor, y por ser siempre tú ahí, en las buenas y en las malas.

A Las Mamis por sacarme de la burbuja, por ser la familia que yo escogí, por ser todas... buenas muchachas.

A Los Papis por...por ser los papis.

A todo aquel que me dijo de una forma u otra que no aguantaría un mes lejos de casa, su falta de confianza fue una razón más para seguir.

Resumen

La preocupación por la calidad de los sistemas basados en software ha aumentado a medida que éste se integra en cada aspecto de nuestras vidas cotidianas. Con el incremento del número de usuario y de los datos almacenados en este se hace necesarios bases de datos más robustas y seguras. Las pruebas de control de calidad han progresado de manera sistémica desde lo manual (la forma más básica) hasta las aplicaciones automatizadas.La Empresa de Tecnologías de la Información para la Defensa (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 el proceso de revisión de las bases de datos en el gestor SQLite son de forma manual, dificultando el proceso de revisión, donde puede existir un margen de error debido a la cantidad de comprobaciones que se deben revisar en la base de datos.El presente trabajo de diploma tiene como objetivo el desarrollo de un componente de revisión de arquitectura de datos para el servidor de gestión *SQLite*.Para ello se realiza un estudio de los sistemas gestores de bases de datos SQLite, las herramientas de pruebas a las bases de datos existentes y como está conformado un estándar de arquitectura de base de datos. Se utiliza el lenguaje de programación PHP por presentar características afines para el desarrollo de la aplicación informática. Se aplica la metodología ProDeSofty se realizan pruebas funcionales al componente para verificar su correcto funcionamiento.

Palabras claves: base de datos, calidad de *software*, estándares, revisión, *SQLite.*

Summary

Concern for the quality of affected systems in software has increased as it is integrated into every aspect of our daily lives. Our country is increasingly engaged in the realization of software that contributes to the information of society. With the increase in the number of users and the data stored therein, more robust and secure databases are needed. If you expect to build high quality software it is necessary to give greater importance and follow-up to the testing phase with the account each project. Quality control tests have progressed systematically from the manual (the most basic form) to automated applications. This diploma work aims to develop a data architecture revision component for the SQLite management server. For this, a study of the SQLite database management systems, the testing tools to the affected databases and how a database architecture standard formedis carried out. The PHP programming language is used for presenting related characteristics for the development of the computer application. The ProDeSoft methodology is applied and functional tests are performed on the component to verify its correct operation.
Keywords: database, software quality, criteria, review, SQLite.

Índice

Índice de Imágenes

Introducción

Durante las últimas décadas el desarrollo de los sistemas de información ha sido vertiginoso, introduciéndose cada vez más en nuestras vidas y haciendo que la gestión empresarial evolucione hacia nuevas formas de gestión, mejora de servicios y de la calidad de los mismos. Con la sucesión del ciclo de vida se ha alcanzado un alto nivel tecnológico, dando paso a que se haga más necesario altos índices de calidad. (De La Paz 2011)

(Reyes 2017) opina que dicha necesidad se observa más recientemente en la incursión de las Tecnologías de la Información y la Comunicación (TIC). Con el avance de las mismas la creación de proyectos y la obtención de información han de ser eficiente y entendibles para facilitar el uso a todo tipo de usuarios. Esta exigencia fomenta la calidad de *software.*En la calidad, *"(…)como regla de oro para todo lo que hagamos en el proceso de informatización*", puso especial énfasis el presidente de los Consejos de Estado y de Ministros. (Puig 2019)

La aplicación de los medios informáticos ha revolucionado la gestión de las empresas cubanas. El crecimiento y difícil manejo de grandes volúmenes de información explica las razones que progresivamente obligan a las organizaciones a desarrollar las tecnologías de bases de datos. El valor de una información actualizada ha crecido tanto que para incrementar o mantener la productividad se deben gestionar eficientemente todos los datos. Son los Sistemas Gestores de Bases de Datos (SGBD) los que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

En Cuba existen empresas que se encargan de realizar pruebas de control de calidad. Estas han progresado de manera sistémica desde lo manual (la forma más básica) hasta las aplicaciones automatizadas para los diferentes sistemas informáticos donde se encuentran las Bases de Datos (BD). A lo largo del ciclo de vida del desarrollo de software se toman medidas para asegurar la calidad de los sistemas de bases de datos, uno de los instrumentos que facilitan esta tarea es la adopción de estándares de diseños de Bases de Datos (BD). El uso de los mismos tiene innumerables ventajas. Entre ellas está asegurar la legibilidad del modelo de datos, incluso en etapas de análisis y diseño, facilitar la tarea de los programadores en el desarrollo de los sistemas y facilitar la portabilidad entre motores de BD, plataformas y aplicaciones.(Gomez, Rodriguez, 2017)

La Empresa de Tecnologías de la Información para la Defensa (XETID),posee un Centro de Calidad, Estándares y Seguridad (CCES) encargado de certificar y evaluar técnicamente productos informáticos de producción nacional o importada. Evalúa y certifica procesos de desarrollo y organizaciones para la

industria de software en las Fuerzas Armadas Revolucionaria (FAR), según normas nacionales e internacionales. Por la importancia que tiene la calidad para esta empresa, el CCES utiliza estándares y herramientas automatizadas para evaluar el producto. Todo esto con el fin de asegurar que se cumpla con el objetivo de brindar al público productos y servicios de calidad.

El auge de la plataforma Android ha causado gran impacto en el desarrollo de aplicaciones móviles a nivel nacional e internacional. Sobre todo, por la facilidad y comodidad de usar las aplicaciones desde cualquier dispositivo móvil en el momento y lugar que se necesite. Logrando una amplia aceptación en el mercado, lugar donde cada vez las partes involucradas son másexigentes. Debido a su facilidad de uso, su pequeño tamaño y su versatilidad SQLite es utilizado en una gran variedad de aplicaciones. Su uso es muy popular en las aplicaciones para móviles con sistema operativo Android (Tatés, 2018)

Una de las pruebas de calidad que se realizan en el Centro de Calidad, Estándares y Seguridad (CCES)es a las bases de datos de las aplicaciones, donde se evalúa que el estándar de arquitectura de datos que utiliza la Empresa de Tecnologías de la Información para la Defensa (XETID) sea empleado correctamente.Actualmente existe una herramienta automatizada que realiza este tipo de pruebas a las bases de datos (BD) en el gestor*PostgreSQL*, llamada Componente de revisión de estándar y arquitectura de datos (CEAD). La mismaestá integrada al marco de trabajo de la empresa llamado Zeolides.

La Empresa de Tecnologías de la Información para la Defensa (XETID) se encarga también de la realización de aplicaciones Android, tales como EnZona o Participación popular. El CCES realiza la revisión y certificación de estos productos, los cuales cuentan con bases de datos realizadas en *SQLite*, ya que se ejecuta en muchas plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración.

Aunque abundan sistemas, no se ha encontrado uno que compruebe que la arquitectura de datos en el gestor *SQLite*esté acorde a un estándar de nomenclatura dado y solo pocos sistemas logran mediante la instrucción de códigos comprobar solamente un elemento de la nomenclatura, por lo que resulta muy engorroso dicho proceso.

Actualmente, en el Centro Calidad, Estándares y Seguridad de la empresa XETID, algunos de los factores que afecta es el tiempo que se dedica a las pruebas de calidad que se realizan a la arquitectura de las bases de datos de *SQLite*, pues son de forma manual, dificultando el proceso de revisión, donde puede existir un margen de error debido a la cantidad de comprobaciones que se deben revisar en la base de datos.

Otro factor que afecta al centro es el no contar con un estándar de revisión de arquitectura para las bases de datos hechas en *SQLite*, dificultando la organización de la revisión, la lista de chequeo que actualmente se utiliza es muy extensa y general, no especifica características de *SQLite*.

A partir de la situación problemática anteriormente expuesta se formula el siguiente **problema de la investigación:** ¿Cómo contribuir a la revisión automática de la arquitectura de datos del gestor *SQLite* en las aplicaciones informáticas?

Definiendo como **objeto de estudio:**Proceso de estandarización de arquitecturas de base de datos.

Tomando como **campo de acción**: Proceso de revisión del estándar de arquitectura de datos del gestor *SQLite*.

Para dar solución al problema planteado se formula como **objetivo general**:Desarrollar una herramienta que contribuya a la comprobación automática de la arquitectura de las bases de datos creadas en el gestor *SQLite*.

Planteando como **idea a defender:**Si se desarrolla una solución informática que revise el estándar de arquitectura de datos en el gestor SQLite, se contribuirá a la identificación de errores en la arquitectura de las bases de datos creadas con ese gestor.

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

- ✓ Sistematizar el marco teórico de la investigación a partir del estudio del arte existente actualmente sobre el tema.
- ✓ Análisis y diseño delas funcionalidades del componente de revisión de arquitectura de datos.
- ✓ Implementarel componente de revisión de arquitectura de datos a partir de los requisitos identificados.
- ✓ Validar el componente de revisión de arquitectura de datos.

Para el logro de los objetivos específicos se trazaron como **tareas científicas para la investigación**:

1- Estudio y análisis de aspectos teóricos sobre las diferentes herramientas de revisión existentes.

2-Determinación de los requisitos funcionales y no funcionales para el desarrollo del componente de revisión de arquitectura de datos.

3- Análisis y diseño de las funcionalidades del componente de revisión de arquitectura de datos.

4-Diseño de la base de datos.

5- Implementación del componente de revisión de arquitectura de datos.

6- Definición y descripción de las pruebas a realizar.

7- Realización de las pruebas del componente de revisión de arquitectura de datos.

Para la realización de esta investigación se utilizaron los siguientes **métodos científicos:**

Dentro del **nivelteórico**:

- Analítico-sintético: Para analizar y extraer los conceptos más importantes en la bibliografía que se relacionan con el objeto de estudio.
- Histórico-lógico: Para analizar las soluciones existentes a este problema y para el estudio de las métricas definidas por las que se rige la empresa, permitiendo recrear un mejor desglose del problema.
- Modelación: Para la creación de modelos, propuestas y alternativas, donde serán representadas las características y relaciones fundamentales del problema que se estará estudiando.

Del nivel empírico que serán referenciados:

- Observación: Permite adquirir información necesaria y se utiliza en todas las fases de la investigación, además de que permite ver la posible solución del problema desde diferentes puntos de vista.
- Análisis documental: este método es usado en el análisis de la bibliografía consultada. Además, es empleado para analizar los conceptos asociados a la investigación.

A continuación, se describe brevemente cada uno de los capítulos en que se encuentra estructurada la tesis. Se compone por introducción, tres capítulos con sus correspondientes epígrafes, conclusiones, recomendaciones, referencias bibliográficas y anexos. Los capítulos están distribuidos de la siguiente forma:

Capítulo 1: Fundamentación teórica y metodológica que sustenta la revisión de la arquitectura de datos del gestor *SQLite*.

En este capítulo se realiza un estudio del gestor de bases de datos *SQLite*, las herramientas de revisión de arquitectura de datos existentes, y las métricas por las que se rige el Centro de Calidad de la XETID para la revisión de bases de datos. También se mencionan las herramientas, la metodología y tecnologías a utilizar para el desarrollo del componente de revisión de la presente investigación.

Capítulo 2: Fundamentación y presentación de la propuesta de solución.

En el capítulo se describen los procesos involucrados, que permitirá conocer particularidades del entorno donde se desarrollará la herramienta. Además, se describen las principales características que debe tener, se definen los requisitos funcionales y no funcionales, e historias de usuario y se realiza un

diagrama de clases del diseño basado en la metodología definida, para un mejor desglose de la herramienta.

Capítulo 3: Implementación y evaluación de los resultados después de ser aplicada la propuesta de solución.

En el capítulo se realiza la validación del componente propuesto. Se utiliza como método de validación las Pruebas Unitarias definidas por la metodología ProDeSoft, que permite verificar el correcto funcionamiento del componente.

CAPÍTULO 1: Fundamentación teórica y metodológica que sustentan la revisión de la arquitectura de datos del gestor SQLite

En este capítulo se realiza un estudio del gestor de bases de datos *SQLite*, las herramientas de revisión de arquitectura de datos existentes, y las métricas por las que se rige el Centro de Calidad de la XETID para la revisión de bases de datos. Se presenta además la metodología a utilizar para el desarrollo del componente.

1.1 Conceptos relacionados

Sistema gestor de base de datos (SGBD)

La definición de sistema gestor de base de datos (SGBD) es un tema muy debatido a nivel global, entre los autores que tratan el tema está(Almonacid, Jemima 2016)la que señala que, un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas, lenguajes y procedimientos, que permite almacenar, acceder y recuperar los datos en la base de datos, proporcionándole al usuario herramientas para describir y manipular dichos datos de manera eficiente y práctica.También (Almonacid, Jemima 2016)asegura que un SGBD actúa como un intermediario entre la base de datos y el usuario, ya que opera como interfaz entre estas dos entidades.

Según la apreciación de (Ramos, Montero 2016)“el objetivo primordial de un gestor es proporcionar eficiencia y seguridad a la hora de extraer o almacenar información en las bases de datos (BD). Los SGBD están diseñados para gestionar grandes bloques de información, que implica tanto la definición de estructuras para el almacenamiento como de mecanismos para la gestión de la información”.

Los SGDB son aplicaciones de *software* diseñadas para facilitar tareas. A medida que crecen el volumen de los datos y el número de usuarios (actualmente son habituales los centenares de gigabytes y los millares de usuarios) el apoyo de los SGBD se vuelve indispensable. En la actualidad existen muchos SGBD, los mas utilizados son *PostgreeSQL, MySQL, MongoDB, Oracle, y SQLite*.(Marin,2019)

El sistema gestor de bases de datos SQLite.

(González 2019)define que “*SQLite* es un sistema de gestión de bases de datos relacional, famoso por su pequeño tamaño.A diferencia de otros sistemas de gestión cliente-servidor el motor de *SQLite* no es un proceso independiente lo que hace que la latencia sea menor y el acceso más eficiente.Debido a su facilidad de uso, su pequeño tamaño y su versatilidad *SQLite* es utilizado en una gran variedad de

aplicaciones, entre ellas Mozilla Firefox o Skype.Su uso ha sido muy popular en las aplicaciones para *smartphones* con sistema operativo Android y iOS".

Algunas de las características más representativas de *SQLite* es que no necesita de un servidor para manipular las peticiones a la base de datos, es decir, permite operar directamente desde el dispositivo donde se encuentra almacenada la base de datos, y no requiere de administración, configuración o mantenimiento.(SQLite Consortium 2019)

Teniendo en cuenta la bibliografía consultada se puede afirmar que este gestor es mucho más rápido y ligero que *MySQL* y *Postgre*. Se ejecuta en muchas plataformas, es de dominio público y por tanto sin costo. Por lo cual, según (López 2016)las empresas de desarrollo de *software* en nuestro país tienen un aumento considerable de la producción de aplicaciones para móviles con sistema operativo Android que utilizan este sistema gestor de bases de datos, como por ejemplo EnZona, Participación Popular, ONAT, Transfer Móvil.

Estándar de arquitectura de base de datos

El SGBD debe ser lo bastante sencillo como para que los interesados puedan comprenderlo, además de coherente y estable, por lo que hay que cuidar su diseño y puesta en marcha. Por tanto, se hace necesaria la creación o adopción por parte de la entidad de un estándar dearquitectura de datos**.**(«Arquitectura de datos» 2019)

(Jorge Sanchez Asenjo 2008) hace referencia a que desde la aparición de los primeros gestores de base de datos se intentó llegar a un acuerdo para que hubiera una estructura común para todos ellos, a fin de que el aprendizaje, manejo y revisión de este *software* fuera más provechoso y eficiente. Los intentos por conseguir una estandarización han estado promovidos por organismos de todo tipo. Algunos son estatales, otros privados y otros promovidos por los propios usuarios. Entre estos organismos estaISO (*International Organization for Standardization*), el cual se encarga de la definición de estándares de gran prestigio internacional.

La arquitectura de datos se diseña y se desarrolla durante la etapa de planificación de un nuevo sistema para establecer la manera en que se procesarán, almacenarán y utilizarán los datos, y cómo se podrá acceder a ellos. Así, para diseñar un sistema eficiente, controlar el flujo de datos y garantizar su protección, es importante conocer la relación y el tipo de gestión necesaria para cada tipo de dato desde el principio. 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. («Arquitectura de datos» 2019)

Para conocer que es un estándar de arquitectura de BD es necesario primeramente referirse al concepto al que hace alusión (Mera y Villamarin 2012). “Estándar puede ser conceptualizado como la definición clara de un modelo, criterio, regla de medida o de los requisitos mínimos aceptables para la operación de procesos específicos, con el fin asegurar la calidad en la prestación de los servicios informáticos. Los estándares señalan claramente el comportamiento esperado y deseado en las aplicaciones y son utilizados como guías para evaluar su funcionamiento y lograr el mejoramiento continuo de los servicios.”

El estándar de arquitectura de datos constituye una guía para el desarrollo de la BD, desde el punto de vista arquitectónico. Consiste en establecer las bases para la puesta en práctica del diseño e implementación de bases de datos. En él deben quedar plasmados las pautas que, según normas internacionales y los convenios de la empresa, rigen el diseño y la implementación de las BD.*(«SQLite Database System Design and Implementation*» 2019).

Las afirmaciones anteriores permiten resumir que un estándar de arquitectura de datos es un documento establecido por consenso, aprobado por un cuerpo reconocido, y que 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, para que se use repetidamente.

Estándares de arquitectura de base de datos *a aplicar en las Pruebas de Calidad de los Datos*

ISO/IEC 25012

Uno de los estándares más utilizados en el mundo en relación al proceso de desarrollo de base de datos es la norma ISO/IEC 25012 de ella es preciso conocer que parte de La ISO/IEC 25000 conocida como SQuaRE (*System and Software QualityRequirements and Evaluation*), es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software, se encuentra compuesta por cinco divisiones: («NORMAS ISO 25000»)

- ISO/IEC 2500n – División de Gestión de Calidad
- ISO/IEC 2501n – División de Modelo de Calidad
- ISO/IEC 2502n – División de Medición de Calidad

- ISO/IEC 2503n – División de Requisitos de Calidad
- ISO/IEC 2504n – División de Evaluación de Calidad

Dentro de estas cinco divisiones se encuentra la ISO/IEC 2501n – División de Modelo de Calidad: Las normas de este apartado presentan modelos de calidad detallados incluyendo características para calidad interna, externa y en uso del producto software. Actualmente esta división se encuentra formada por:

ISO/IEC 25010 - Sistemas y modelos de calidad de software.

ISO/IEC 25012 - Modelo de calidad de datos: el cual define un modelo general para la calidad de los datos, aplicable a aquellos datos que se encuentran almacenados de manera estructurada y forman parte de un Sistema de Información.

La Calidad del Producto de Datos se puede entender como el grado en que los datos satisfacen los requisitos definidos por la organización a la que pertenece el producto. Son precisamente estos requisitos los que se encuentran reflejados en el modelo de Calidad de Datos mediante sus características.

Estándar de arquitectura de datos para la XETID

El estándar a utilizar en la revisión automática de la arquitectura a bases de datos en el gestor SQLite *es*Estándar de arquitectura de datos para la XETIDen su version 1.0. Este documento realizado por el Centro Calidad, Estándares y Seguridad de la empresa XETIDen el año 2017, constituye una guía para el desarrollo de la base de datos (BD). Desde el punto de vista arquitectónico tiene el propósito de brindar una visión la estrategia de desarrollo del Centro. El propósito de la totalidad de este documento consiste en establecer las bases para el conocimiento teórico y los fundamentos para la puesta en práctica del diseño e implementación de bases de datos.

Entre las pautas que contiene dicho estándar de arquitectura de BD están:

- Tipo de Base de Datos.
- Base tecnológica para la arquitectura de datos.
- Sistema Gestor de Base de Datos (SGBD).
- Herramienta de diseño.

- Herramientas de administración .
- Seguridad de datos.
- Estándar de nomenclatura.
- Tipos de datos.
- Políticas de indexado.
- Integridad.
- Normalización.
- Rendimiento.
- Preparación de las BD para la réplica.
- Políticas de trabajo.

Estos estándares son revisados en su mayoria de forma automática,*abundan sistemas* elaborados a nivel mundial que *en la búsqueda de alcanzar la perfección en las realizaciones de Base de datos (BD) realizan todo tipo de pruebas basadas en estos estándares de calidad.*

1.2 Herramientahomóloga

Componente de revisión de arquitectura de datos (CEAD)

La Empresa de Tecnologías de la Información para la Defensa (XETID), con el objetivo de desarrollar aplicaciones de alta calidad y automatizar los sistemas existentes, ha elaborado un sistema automatizado que facilita el proceso de pruebas de liberación del Centro de Calidad, Estándares y Seguridad perteneciente a la dicha empresa. Sus autores*(Guanche, Pérez y Reyes 2017)*aseguran que la creación del componente de revisión de arquitectura de datos(CEAD) viabiliza comprobar una correcta arquitectura de datos automáticamente, además de que parte de un estándar conocido.

Este componente, para los probadores de la calidad del centro, representa una poderosa herramienta que permite optimizar el tiempo de trabajo empleado a la revisión de la arquitectura de las bases de datos. Además, realiza una búsqueda exhaustiva de no conformidades proporcionando un pequeño margen de error.

La herramienta no es adaptable a cualquier estándar, actualmente está desarrollada para la revisión de las BD por el estándar de arquitectura de bases de datos en Postgree.(Guanche, Pérez y Reyes 2017). Por esta razónno es posible su utilización en la verificación de los productos con gestor de base de datos en SQLite. Si bien es cierto que el código de CEAD se puede modificar incluyendo extensas líneas de código, no es lo que el cliente (Centro de Calidad Estándares y Seguridad, CCES**,** de la Xetid) desea por varias razones. Por ejemplo, esta modificación del código trae consigo una sobrecarga de información a manejar por el probador en una misma interfaz. Por tanto, trae consigo una afectación al tiempo de familiarizacion de los trabajadores con la nueva interfaz. Para evitar estos problemas el cliente desea dos componentes separados(CEAD Postgree, y CEAD SQLite), ambos integrados al marco de trabajo Zeolides que utiliza la empresa, ambos componentes iguales en apariencia, pero cada uno con su funcionalidad dada. Desea que el probador utilice el componente que necesite según el tipo de base de datos que esté revisando en ese momento.

Luego de realizarse la investigación y estudio referente al tema de las herramientas existentes, se resume que la adherencia a un estándar de arquitectura de datos en particular es un área en la que se han presentado relativamente pocos trabajos. De cualquier forma, automatizar las pruebas de adherencia a un estándar es una ventaja para el desarrollo y la certificación de calidad de un producto de software. Al mismo tiempo se evidencia que este sistema no fue creado para realizar comparaciones entre un estándar y la arquitectura de una BD para el gestor *SQLite*,dificultando el proceso de revisión para el CCES.Por tanto, se procede a la creación de una herramienta para lograr automatizar la revisión de la arquitectura de datos en el gestor SQLite.

1.3. Metodología de desarrollo de software

Las metodologías de desarrollo de *software*, “son un conjunto de procedimientos, técnicas y ayudas a la documentación para el progreso de productos de *software*”. Estas definen quién está haciendo qué, cuándo y cómo alcanzar un objetivo específico, al proporcionar normas para el desarrollo eficiente del software con calidad. Actualmente existen dos grandes clasificaciones, las metodologías tradicionales y las ágiles. Estas últimas se basan en la satisfacción del cliente, equipos de *software*pequeños con alta motivación y un mínimo de productos de trabajo de la Ingeniería de *Software*. (Pressman 2010)

Proceso de Desarrollo de Software (ProDeSoft) v1.7

Proceso de Desarrollo de *Software* (ProDeSoft) es la metodología que utiliza la XETID en sus proyectos. Según su documento oficial (XETID 2017) esta metodología divide el ciclo de vida del producto de la siguiente forma:

Figura 1Ciclo de vida del producto en Prodesoft, tomado de PDS v1.7 Xetid

- 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.(XETID 2017)

- Modelación: se capturan las partes esenciales del sistema, se identifican los procesos de negocio más significativos y se aceptan los requisitos funcionales, se obtiene la línea base de la arquitectura y una estrategia para la construcción de la aplicación. El hito fundamental de esta etapa es la liberación de la arquitectura de sistema, datos y despliegue.(XETID 2017)

- Construcción: se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. En esta fase todas las características, componentes, y requerimientos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto.(XETID 2017)

- 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.(XETID 2017)

- 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.(XETID 2017)

(XETID 2017)plantea que es un modelo iterativo e incremental, es un enfoque en el que las fases del ciclo de vida se descomponen en iteraciones. En las primeras fases las iteraciones son realizadas con mayor énfasis en la determinación del alcance del proyecto, planificación, identificación y descripción de requisitos y la creación de la línea base de la arquitectura, luego a medida que va avanzando el proyecto el énfasis de las iteraciones cambia centrándose más en la implementación de componentes.

El documento (XETID 2017) se refiere a que la metodología cuenta con un desarrollo basado en componentes lo cual le permite alcanzar un mayor nivel de reutilización de *software*, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.

Conclusiones parciales

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.

CAPÍTULO 2: Fundamentación y presentación de la propuesta de solución

En el capítulo se describen los procesos involucrados, que permitirá conocer particularidades del entorno donde se desarrollará la herramienta. Además, se describen las principales características que debe tener, las herramientas y tecnologías a utilizar. Se definen los requisitos funcionales y no funcionales, las historias de usuario y se realiza el diagrama de clases del diseño basado en la metodología definida.

2.1 Tecnologías a utilizar para la realización de la aplicación

Lenguaje de Modelado

Lenguaje de Modelado Unificado 8.0 (*UML* por sus siglas en inglés). Está respaldado por el *Object Management Group* (por sus siglas en inglés *OMG*). *UML* ayuda a especificar, visualizar y documentar los modelos de sistemas de software, incluyendo su estructura y diseño, de manera que cumpla con todos estos requisitos. Se puede usar para el modelado de negocios. Si se utiliza cualquiera de la gran cantidad de diagramas *UML*, se pueden analizar los requisitos de su aplicación futura y diseñar una solución que les satisface. («OMG | Object Management Group» 2019)

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software. *UML* cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas, diagramas que son necesarios para la realización del módulo por lo que se selecciona como lenguaje de modelado a utilizar.

Herramienta de Modelado:

Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering o Ingeniería de *Software* Asistida por Computadora) son identificadas como cualquier herramienta que se emplea para automatizar alguna actividad. Estas ayudan en las tareas relacionadas con la fase de desarrollo del *software* como la

especificación, estructurado, análisis, diseño, codificación, pruebas y otras actividades como gestión de proyectos y gestión de la configuración. Además están destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste en términos de tiempo y de dinero.(«OMG | Object Management Group» 2019)

Visual Paradigm for UML v5.0.

Según su página oficial, visual-paradigm.com, *Visual Paradigm*es una herramienta profesional y multiplataforma, implementada en lenguaje *Java* que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite construir diagramas, generar código desde los diagramas y documentación, también brinda al equipo de desarrollo de software la posibilidad de realizar el análisis y diseño de los sistemas con mayor eficacia.(«Visual Paradigm Frequently Asked Questions» 2019)

Lenguaje de Programación:

PHP v5.6

(«PHP: ¿Qué es PHP? - Manual» 2019) explica que es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales.

El preprocesador de hipertexto (PHP) es un lenguaje de programación del lado del servidor, fácil de usar. Presenta las características de la sintaxis de los lenguajes C, *Java* y *Perl*, es ampliamente utilizado en el mundo para el desarrollo web.

JavaScript

Según («What is JavaScript?» 2019)*JavaScrip*t es un lenguaje de programación que te permite realizar actividades complejas en una página web, como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones gráficas 2D/3D. Es la tercera capa del pastel de los estándares en las tecnologías para la web, dos de las cuales son (HTML y CSS).

JavaScript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. Es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con *JavaScript* se puede crear diferentes

efectos e interactuar con los usuarios.(«Free JavaScript training, resources and examples for the community» 2019)

XML

Su página oficial, («Introducción a XML» 2019),plantea que*XML* es un lenguaje de marcado similar a HTML. Significa *Extensible Markup Language* (Lenguaje de Marcado Extensible) es un lenguaje de marcado de propósito general. Esto significa que, a diferencia de otros lenguajes de marcado, *XML* no está predefinido, por lo que se debe definir sus propias etiquetas. El propósito principal del lenguaje espermitir la transferencia de información entre diferentes sistemas.

Dentro de sus aplicaciones se utiliza para representar información estructurada en la web, de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por muy diversos tipos de aplicaciones y dispositivos.

Sistemas gestores de base de datos:

PostgreSQL v9.3:

Para la gestión de los datos de la aplicación se utiliza *PostgreSQL*en su versión 9.3, la cual plantea su página oficial, («PostgreSQL» 2019), es la última versión del sistema de base de datos relacional, orientado a objetos, de código abierto líder en el mundo. Esta versión amplía la confiabilidad, disponibilidad y capacidad de integración de *PostgreSQL* con otras bases de datos.Permite el intercambio de datos bidireccional entre sistemas. Fue seleccionado como gestor de BD por su conocida estabilidad, robustez, fuerte coherencia y seguridad.

IDE de desarrollo:

NetBeans v8.0

(«NetBeans IDE 8.2 Release Information» 2019) explica que *NetBeans IDE* es el IDE oficial para *Java* 8. Con sus editores, analizadores de código y convertidores, puede actualizar sus aplicaciones de manera rápida y sin problemas para usar nuevas construcciones de lenguaje *Java* 8. *NetBeans IDE 8.0* proporciona analizadores de código y editores listos para trabajar con las últimas tecnologías, posee nuevas herramientas para HTML5 y *JavaScript*, en particular para *AngularJS*; y mejoras en el soporte de PHP y C / C ++.

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java, pero nos permite programar en diversos lenguajes. Existe además un número importante de módulos para extenderlo. *NetBeans IDE* es un producto libre y gratuito sin restricciones de uso.

Marco de trabajo

Según el análisis de (José Miguel Reyes Pérez 2017) un *framework*(marco de trabajo)es un conjunto de herramientas, librerías, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables. Los *frameworks* se construyen para dar flexibilidad y generalidad, tratando de cubrir un dominio entero en vez de problemas determinados.

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. (Reyes 2017)

(«Tecnologias y Ciberseguridad.Marco de Trabajo para el Desarrollo de Aplicaciones Web (Zeolides).pdf.» 2019) hace hincapié en que Zeolides brinda prestaciones que aseguran un alto grado de calidad e integración de las aplicaciones realizadas. Aspectos como la seguridad, el monitoreo de trazas, la gestión del personal y los reportes, son contenidos entre las funcionalidades de la solución. Por lo que permite la obtención de soluciones con un alto grado de robustez, dadas por el número de funcionalidades brindadas durante el desarrollo de soluciones web.

Ext JS v2.2

(José Miguel Reyes Pérez 2017) plantea que este *framework*está bajo la licencia GPLv3 y multiplataforma, la versión utilizada es de las más avanzadas de *Ext JS*. Posibilita el uso del patrón Modelo Vista Controlador (MVC) para la vista. En las versiones anteriores de *ExtJS*el usuario del *framework*estructuraba su aplicación a su manera, problema que a partir de la versión 4.0 se está intentando solucionar, al definir una arquitectura para que todo el que use la plataforma y desarrolle a partir de ella lo haga de la misma manera, pudiéndose compartir los segmentos de código y ser más entendible por el que use el código.

Servidor para aplicaciones web:

Apache v2.2

Según .(«About the Apache HTTP Server Project - The Apache HTTP Server Project» 2019) el Proyecto Apache HTTP Server es un esfuerzo de desarrollo de software colaborativo destinado a crear una implementación de código fuente robusta, comercial, funcional y de libre acceso de un servidor HTTP (Web). El proyecto es administrado conjuntamente por un grupo de voluntarios ubicados en todo el mundo, que utilizan Internet y la Web para comunicarse, planificar y desarrollar el servidor y su documentación relacionada. Este proyecto es parte de la *Apache Software Foundation*. Además, cientos de usuarios han aportado ideas, código y documentación al proyecto

La investigación de (Pulido 2019) revela que desde 1996 es el servidor más utilizado en Internet y es el utilizado tanto en sistemas *GNU/Linux* como *Windows*. Es un servidor robusto y con un ciclo de desarrollo muy rápido, gracias a la gran cantidad de colaboradores con los que dispone.

Son elegidas estas tecnologías por las características presentadas, además la herramienta a crear debe estar integrada a una solución existente, CEAD, que tiene estas mismas particularidades.

2.2 Propuesta de solución

Teniendo en cuenta que los sistemas existentes no fueron creados para realizar pruebas a la arquitectura de datos y no permiten realizar comparaciones entre un estándar y la Base de Datos, lo que trae consigo que se dificulte el proceso de revisión siendo actualmente de forma manual y engorrosa. Se decide crear una herramienta que logre automatizar la extracción de elementos de la arquitectura de datos y compare un estándar definido con lo extraído.

La solución se basa en una herramienta de revisión de estándar y arquitectura de datos, la cual visualmente estará estructurada en dos áreas de trabajo fundamentales.La primera permitirá adicionar, modificar o eliminar una prueba determinada y listar todas las adicionadas anteriormente.Además, brindará la opción de darle seguimiento a una prueba determinada, así como ver los detalles de esta en específico y exportar el listado de pruebas como informe.El sistema también establecerá un lenguaje sencillo y común, que permitirá al usuario la selección de los elementos que desea revisar de la arquitectura (entre indexado, tipo de datos y nomenclatura). Para ello es necesario previamente que se adicione un estándar a la prueba, validándose que no permita que se realice la revisión hasta que no exista una conexión a una Base de Datos. También facilitará la obtención de incidencias y detalles de una prueba determinada. Una vez realizada la revisión, se podrá utilizar la siguiente área. La segunda área de trabajo brindará la posibilidad de visualizar los errores encontrados una vez que se haya

realizado el reconocimiento de errores a la base de datos, mostrándose detalles de la misma. Esta permitirá generar las no conformidades encontradas automáticamente.

2.3 Modelo conceptual de los datos y sus conceptos

Plantea (Sommerville 2011) que un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las cosas, entidades, ideas, conceptos u objetos del “mundo real” o dominio de interés.

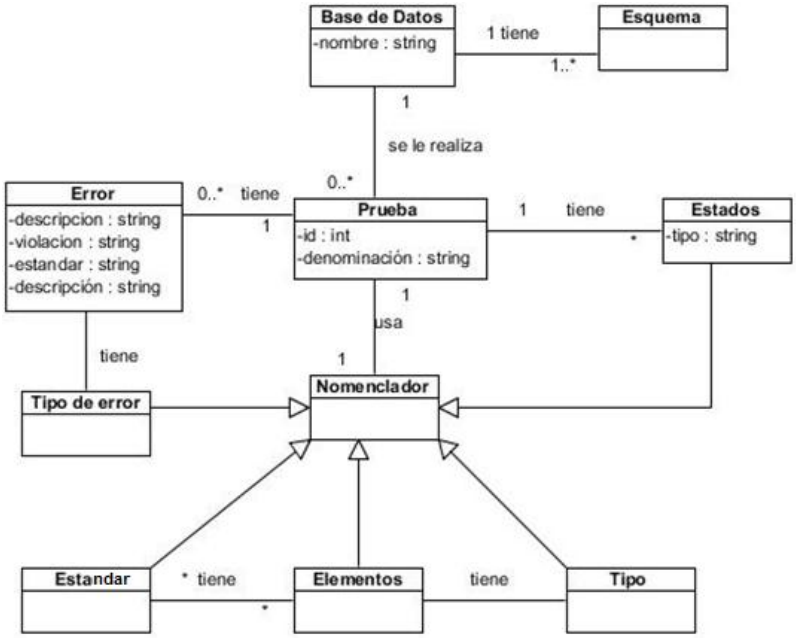

Figura 2Modelo Conceptual de los datos.

Principales conceptos

Prueba: Contiene las pruebas gestionadas a la Base de Datos.

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

Estado: Es el estado que tiene la prueba, comenzará primeramente en registrada, ejecutada o aceptada.

Error: Describe los errores físicos de arquitectura de datos de una Base de Datos.

Estándar: Contendrá un patrón, modelo o punto de referencia que permitirá medir o valorar elementos

similares.

Elemento: Define los nombres propios de bases de datos, esquemas, tablas, campos, llaves, funciones dominios y secuencias.

Esquemas: Esquemas a los cuales se le realizará la revisión.

2.4 Requisitos funcionales

Los requisitos funcionales son las facilidades que el sistema brindará o deberá brindar al usuario. Deben ser muy claros, fáciles de probar y específicos. (Sommerville 2011). Luego de la entrevista con el cliente, y a través de una encuesta aplicada al mismo, se alcanzó un total de cinco (5) requisitos funcionales.En el caso del Componente de revisión de arquitectura de datos *SQLite* de la XETID, se decidió dividir visualmente en dos áreas de trabajo, una para la selección de elementos de una arquitectura de datos que se desea probar y otra para la revisión de arquitectura de datos.

RF1 Gestionar prueba a la arquitectura de la Base de Datos: Permite Adicionar, Modificar y Eliminar pruebas ala arquitectura de la Base de Datos.

RF1.1 Adicionar prueba: Permite adicionar pruebas de arquitectura de datos.

RF1.2 Modificar prueba: Permite modificar pruebas de arquitectura de datos.

RF1.3 Eliminar prueba**:** Permite eliminar pruebas de arquitectura de datos.

RF1.4 Listar prueba: Permite listar las pruebas de arquitectura de datos adicionadas.

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

Descripción: Muestra todas las acciones realizadas a la prueba de arquitectura de Base de Datos desde su creación hasta su liberación.

RF3: Mostrar detalles de la prueba de arquitectura de la Base de Datos

Descripción: Muestra los detalles de las revisiones a la prueba de arquitectura de la Base de Datos.

RF4 Realizar revisión de la Base de Datos

Descripción: Realiza una prueba a la arquitectura de la Base deDatos seleccionada en la prueba.

RF5 Listar errores de revisión

Descripción: Se listan todos los errores encontrados durante la revisión a la arquitectura de datos.

2.5 Prototipos de interfaz de usuario y Especificación de requisitos

Especificación del requisito RF1.1 Adicionar prueba de la Base de Datos.

Conceptos tratados	**Conceptos**	**Atributos**
	Prueba Estándar Elemento Tipo	Id Denominación
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 de BD.** 2. Se muestra la interfaz Prueba de BD, y se habilita el espacio de trabajo Pruebas a Bases de Datos. 3. Se presiona el botón Adicionar de la barra de herramientas del espacio de trabajo Pruebas a Bases de Datos. 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 Adicionar datos de	

	conexión, en la cual se introducirán los datos: • Dirección: IP del servidor de BD. • Puerto: Puerto del servidor de BD. • Usuario: Usuario de BD • Clave: Contraseña del usuario de BD • Nombre de 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	Media.
Validaciones	Para las acciones Aceptar

	Los botones se habilitarán cuando se haya introducido los valores correctos, en el campo. Si ocurre un error al intentar acceder a la Base de Datos, el sistema muestra el mensaje: "*Los datos de conexión son incorrectos.* " Sobre tool-tips: La opción Adicionar del espacio de trabajo Pruebas a Bases de Datos muestra como mensaje *tool-tips* "*Adicionar prueba* ***(Alt+I)***". La opción Guardar del espacio de trabajo Adicionar datos de conexión muestra como mensaje tool-tips "*Guardar datos".* La opción Cancelar del espacio de trabajo Adicionar datos de conexión muestra como mensaje tool-tips "*Cancelar acción".* El botón Cancelar de la interfaz Pruebas a Bases de Datos muestra como mensaje tool-tips "*Cerrar ventana sin guardar los datos*". El botón Guardar de la interfaz Pruebas a Bases de Datos muestra como mensaje tool-tips "*Guardar prueba*". El botón Cancelar de la interfaz Pruebas a Bases de Datos muestra como mensaje tool-tips "*Cancelar acción*". Si existen campos con valores incorrectos se marca en rojo y se muestra un mensaje tool-tips 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 Interfaz de Usuario (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 Interfaz de Usuario Adicionar 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.

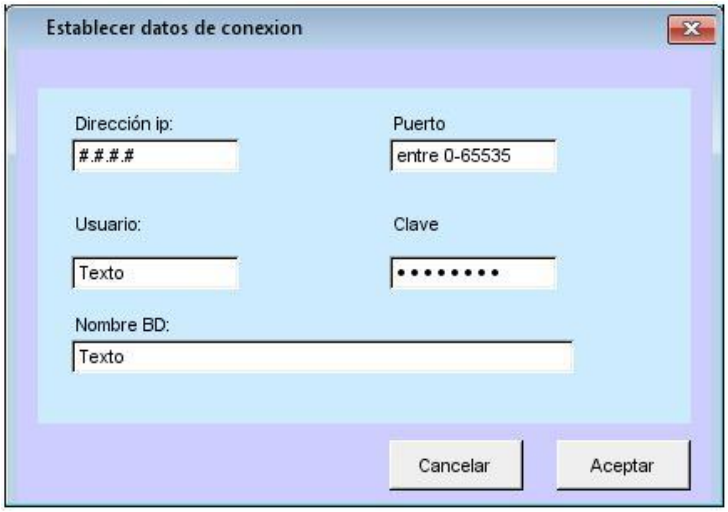

Figura 3PIU Adicionar datos de conexión

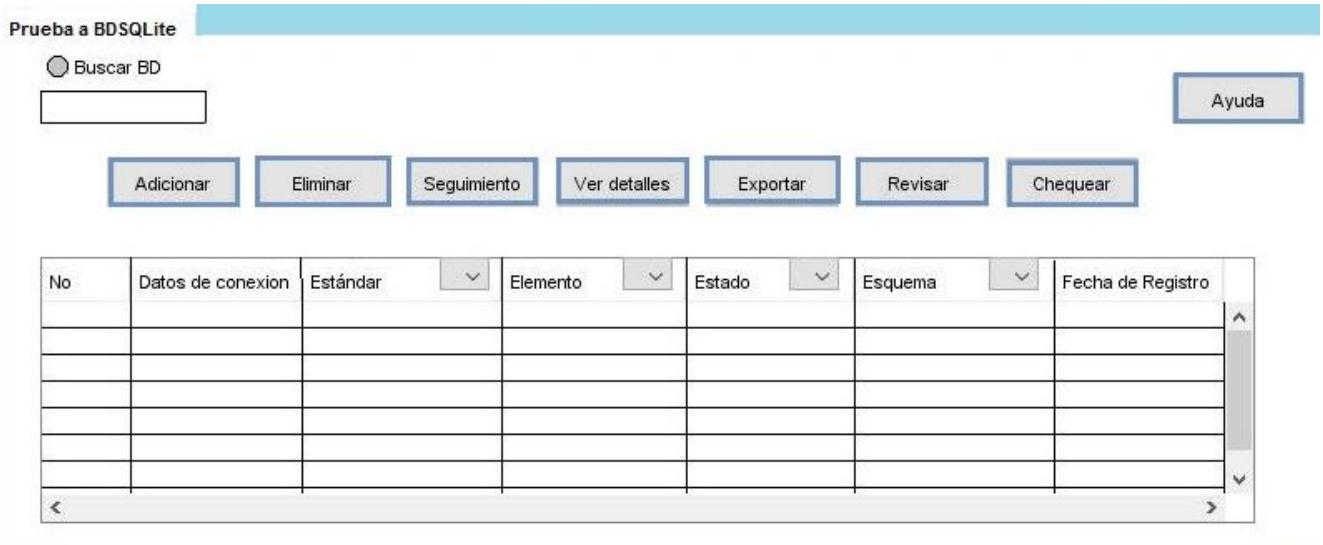

Figura 4PIU Interfaz Principal

Especificación del requisito RF4 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.	

	3. Se selecciona de la lista de pruebas, la que se le desea realizar la acción. 4. Se presiona el botón Revisar. 5. Se gestiona 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 se habilitará cuando se seleccione una prueba de la lista. Si ocurre un error al intentar acceder a la Base de Datos, el sistema muestra el mensaje: "La Revisión la Prueba no se pudo realizar.", mantiene abierta la interfaz Prueba a Bases de Datos. Sobre tool-tips: La opción Revisar del espacio de trabajo Prueba a Bases de Datos muestra como mensaje tool-tips “Ejecutar revisión”.
Post-condiciones	Se ha revisado la Prueba
Post-requisito	No procede.

Figura 5PIU Realizar revisión de la Base de Datos

Especificación del requisito RF5 Listar errores de revisión

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 principal, se habilita el espacio de trabajo Prueba a Bases de Datos. 2. S 3. Se listan en el grid_panel los errores generados. Ver figura 9.	
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.*" Sobre *tool-tips*:	
Post-condiciones	Se ha generado una lista con los errores encontrado en la prueba de BD.	

Figura 6PIU Listar errores encontrados

2.6 Requisitos no funcionales del sistema

Los requerimientos no funcionales describen las restricciones del sistema; no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de este como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. (Sommerville 2011)

Usabilidad

RNF1 El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.

RNF2 El software tendrá la opción de Ayuda, Especificaciones de requisitos y Manual de Usuario lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

RNF3 El sistema hará uso del empleo de teclas calientes.

RNF4 El sistema contendrá íconos que representan visualmente la acción para lograr un menor tiempo de aprendizaje.

RNF5 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.

RNF6 Tendrá una correcta combinación de colores para personas con diferentes problemas visuales.

RNF7 La presentación, composición y formularios del sistema se visualizará sin problemas en Opera y Firefox (la versión debe ser mayor de 28 y menor que 35).

RNF8 El sistema informa claramente al usuario cuando se produce un error sobre lo ocurrido y de cómo solucionar el problema.

Confiabilidad

RNF9 El sistema podrá realizar salvas automáticas de la información en tiempo real, para en caso de existir alguna falla ajena a este, no se pierda la información almacenada, ni la introducida en sus interfaces.

RNF10 El sistema se mantendrá disponible un 99% del tiempo comprendido en una semana de funcionamiento de 24 horas diarias. Esto equivale a reserva de una hora diaria para reparaciones en horario de bajo acceso y hasta 2 horas los fines semana.

RNF11 Los mensajes de error contendrán instrucciones claras en cuál es el paso siguiente.

RNF12 El tiempo de recuperación del sistema luego de haber ocurrido algún error será mínimo.

Rendimiento

RNF13 Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 8 segundos para las actualizaciones y 20 para las recuperaciones. (En dependencia siempre del tamaño de la BD)

RNF14 El sistema será realizado para un máximo de 20 usuarios.

Portabilidad

RNF15 El sistema se ejecutará en el Sistema Operativo Linux en cualquiera de susdistribuciones, así como en Windows.

Seguridad

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.

RNF17 Las contraseñas introducidas en el sistema serán encriptadas.

Software

RNF18 Para el servidor:

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

Hardware

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 Diseño de la propuesta de solución.

Diagrama de componentes

De acuerdo al modelo conceptual, la especificación de requisitos y los prototipos de interfaz de usuario definidos anteriormente para el Componente de revisión de la arquitectura de datos en el gestor de bases de datos SQLite, y dado que el sistema se incluye al marco de trabajo Zeolides de la Xetid, se especifica la integración del sistema con otros componentes. El componente CEAD SQLite toma de Zeolides el método de generación automática de no conformidades del componente del mismo nombre y además las almacena ahí. Para el botón de Ayuda de la interfaz principal del componente se utiliza el componente Ayuda. Para acceder al marco de trabajo Zeolides el usuario debe haberse autenticado previamente por lo que el componente toma funcionalidades de Usuario y Estructura y Composición (para conocer el rol y el área de trabajo de cada usuario). De igual manera toma elementos del componente Seguridad, asociados al proxy del sistema. Obteniendo el siguiente gráfico de diagrama de componentes externos.

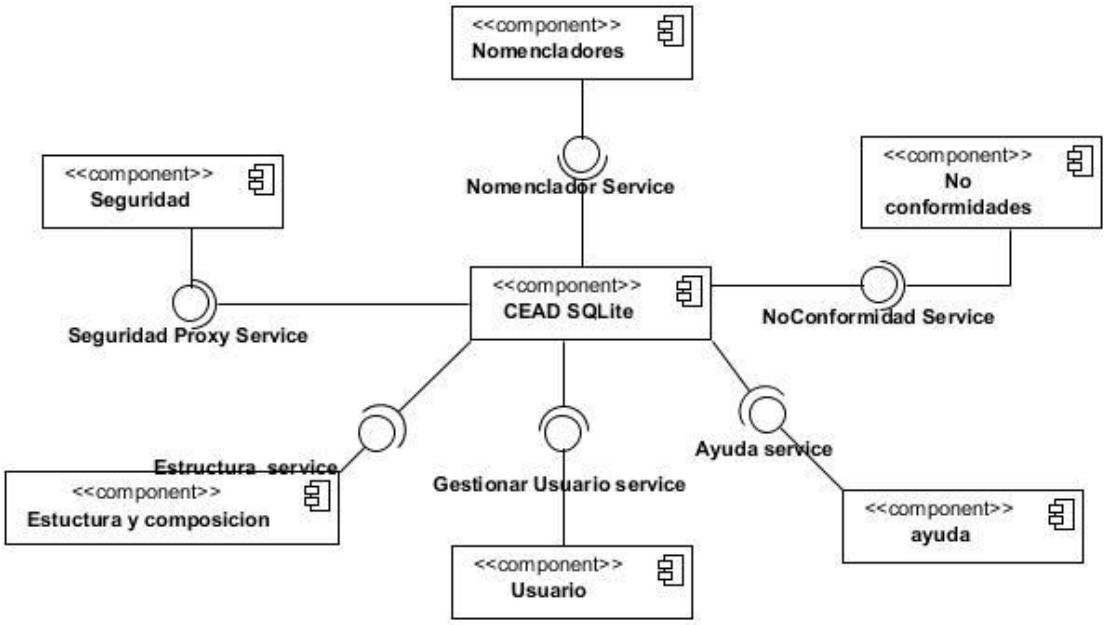

Figura 7Diagrama de componentes

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.

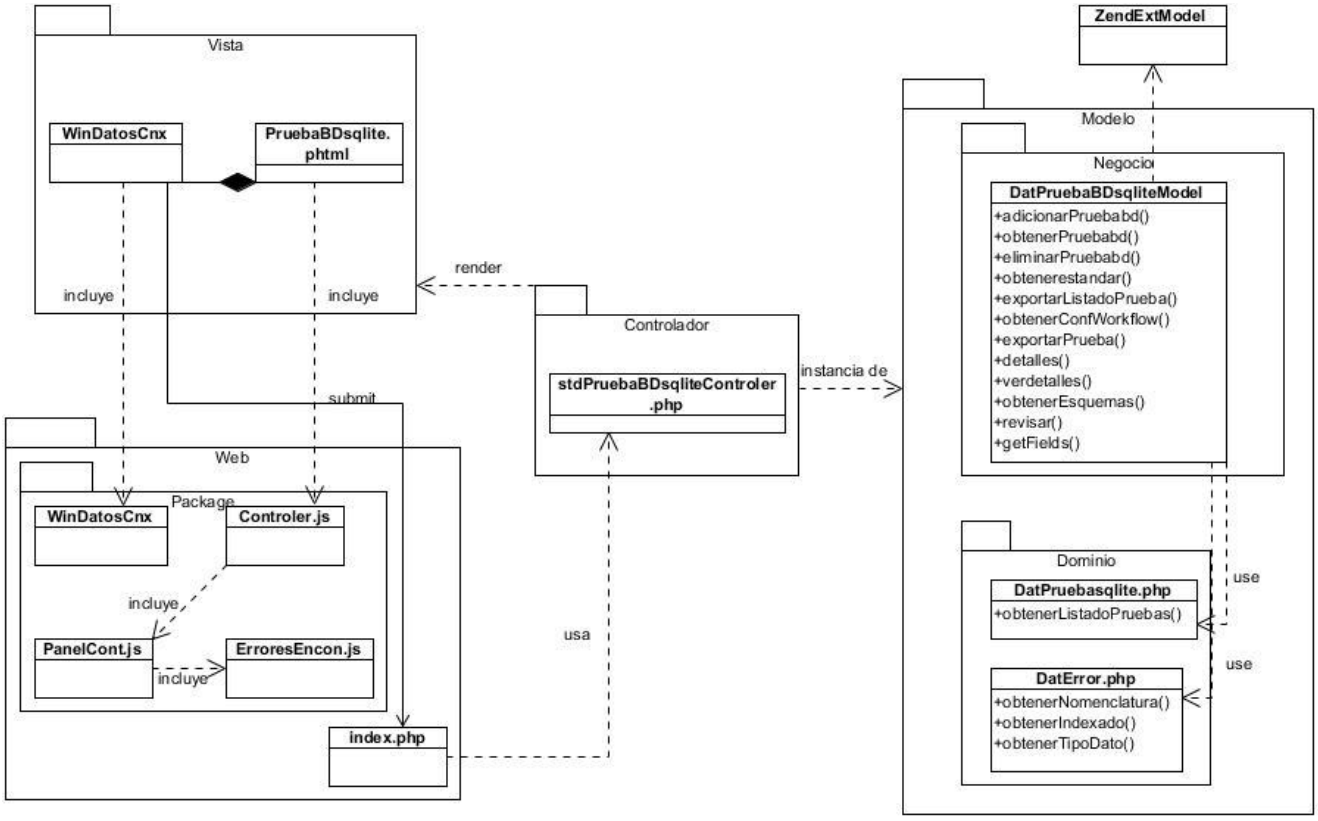

Figura 8Diagrama de clases de diseño

2.8 Diseño de la Base de Datos

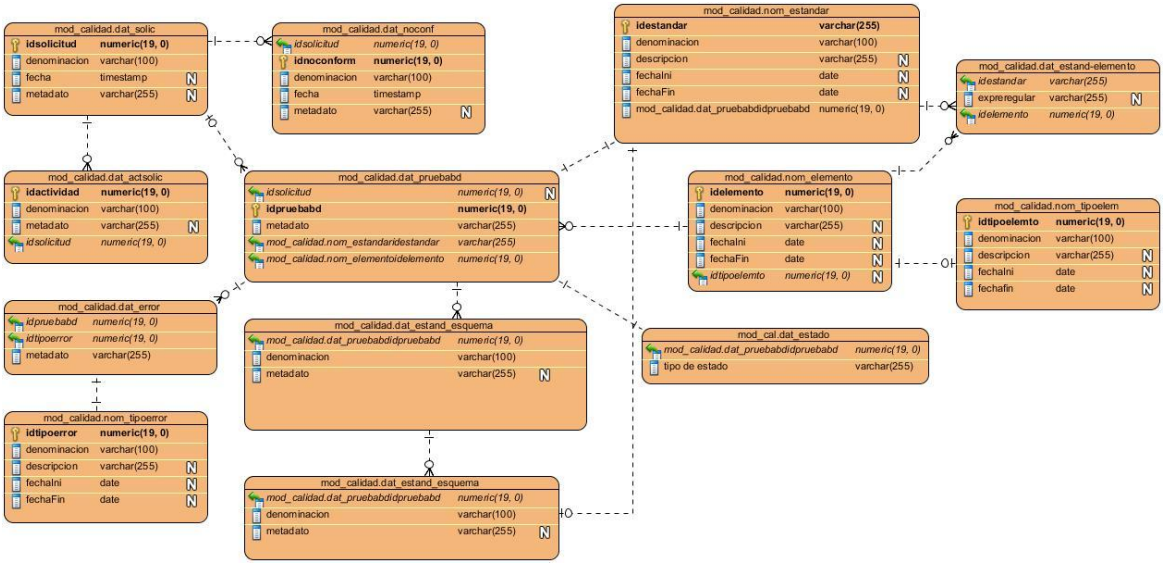

Figura 9Diagrama de arquitectura de base de datos.

2.9Patrones de diseño

Las profesoras (Gros, Escofet y Marimón 2016) basándose en lo descrito por (Gamma, Johnson y Vlissides 1995) definen que un patrón de diseño es la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software, que brindan una solución ya probada y documentada a dichos problemas que están sujetos a contextos similares.En el componente de revisión de estándar y arquitectura de datos en el gestor SQLite, se utilizaron los siguientes Patrones Generales de Software para la Asignación de Responsabilidades (GRASP por sus siglas en inglés) para asignar responsabilidades y los patrones Gang of four (GOF) conocidos como los patrones de la pandilla de los cuatro.

2.9.1 Patrones GRASP

Según el estudio que se realizó a (Visconti y Astudillo 2014) se definen cinco patrones GRASP utilizados en el desarrollo de la solución informática.

Patrón Experto

Asigna una responsabilidad a la clase que tiene la información necesaria para cumplirla. Los objetos se

valen de su propia información para hacer lo que se les pide. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento.(Visconti y Astudillo 2014)

En el componente, este patrón se utiliza, por ejemplo, para realizar las pruebas de arquitectura de datos . La clase entidad Prueba, es la que tiene la información necesaria para realizar esta operación, por tanto, es la clase experta.

Patrón Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.(Visconti y Astudillo 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

Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Este patrón se pone de manifiesto en todo el componente debido a que cada uno de los eventos generados por el usuario es redirigido a una clase controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión.

Patrón Creador

El patrón Creador guía la asignación de responsabilidades con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento, al escogerlo como creador, se da soporte al bajo acoplamiento.(Visconti y Astudillo 2014)

Este patrón se aplica principalmente a la clase controladora, que se encarga de crear objetos de la clase modelo. Por ejemplo, la clase *Stdpruebabdsqlitel*Controler* instancia a la clase *DatPruebabdsqliteModel* 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

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.(Visconti y Astudillo 2014). En el componente las clases tienen

responsabilidades moderadas en un área funcional y colaboran con las otras para llevar a cabo las tareas, por ejemplo, se puede observar que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

2.9.2 Patrones GOF

El catálogo de patrones más famoso es el contenido en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”, también conocido como: El libro GOF (Gang-Of-Four Book). Según este documento (Erich Gamma, Ralph Johnson y John Vlissides 1995), estos patrones se clasifican por su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos. Los patrones GOF utilizadosson los siguientes:

Patrón Facade (Fachada)

Se utiliza porque el sistema a desarrollar tiene que acceder a parte de la funcionalidad de un sistema complejo.Provee de una única interfaz para acceder a un sistema completo, que actúa como único punto de acceso al mismo, y hace que éste sea más fácil de utilizar.

2.10Patrón arquitectónico.

El diseño del sistema está basado en la arquitectura Modelo Vista Controlador (MVC). Es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, brindando un desarrollo del software integrado. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.(Servicio de Informática Universidad de Alicante 2020)

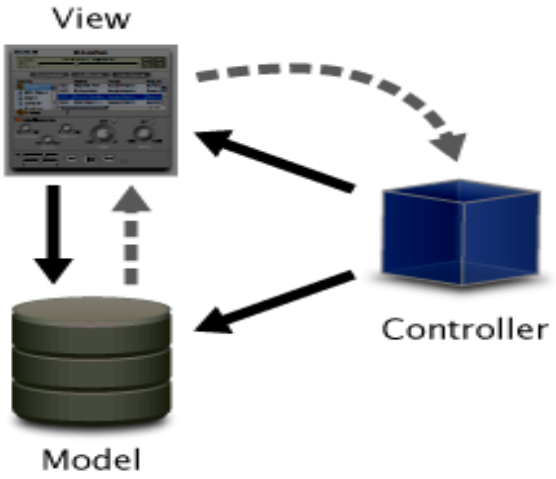

Figura 10Modelo-Vista-Controlador, tomada de internet

Modelo: Esta capa contiene una representación de los datos que maneja el sistema, su lógica de negocio, yse encarga de gestionar todo lo relacionado con el acceso y procesamiento de los datos. A través de clases modelos se gestiona todo lo relacionado con los elementos del proceso de monitoreo.(Servicio de Informática Universidad de Alicante 2020)

Vista: Esta capa se encarga de gestionar todo lo relacionado con la lógica de interfaz. Está integrada por un fichero .phtml encargado de cargar los ficheros ucid_all.js, ext-all-debug.js que será el fichero encargado de gestionar lo relacionado con la librería ExtJs, y el fichero .js que se encargará de llamar internamente a todos los demás ficheros que integran la vista y muestran la información visual al usuario.(Servicio de Informática Universidad de Alicante 2020)

Controlador: Esta capa,se encarga de gestionar y controlar todo el flujo de relación entre la vista y el modelo, gestionando el flujo de información entre ellos,realizándole el pedido al modelo de lo que el usuario quiere ver en la vista.(Servicio de Informática Universidad de Alicante 2020)

Conclusiones parciales

En este capítulo el estudio de las tecnologías de desarrollo de software a utilizar en la aplicación permitió la elección de las herramientas y la metodología adecuadas para desarrollar dicho sistema. Se definió la propuesta de solución, creando las bases para la correcta implementación del componente de revisión de la estandarización de la arquitectura de datos. La realizacióndel modelo conceptual de los datos, así como de los diagramas de componentes y clases del diseño, permitió conocer con mayor

profundidad los conceptospresentes en el problema y la relación existente entre ellos. Se especificaron los requisitos funcionales y no funcionales de la aplicación y su descripción detallada, sirviendo 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 evaluación de los resultados después de ser aplicada la propuesta de solución

En este capítulo se establecen criterios asociados a la implementación y validación del sistema propuesto. En la realización de diversos sistemas informáticos juega un papel fundamental el uso de las pruebas. Para lograr esto se debe llevar a cabo estrategias a la hora de evaluar dinámicamente el software partiendo del estudio y puesta en práctica del método de caja negra. Con este propósito se definieron los casos de pruebas afines con los procesos de negocio más significativos para la arquitectura, definidos por la metodología utilizada.

3.1 Diagrama de despliegue

El modelo de despliegue representa un modelo de objetos que muestra cómo se distribuye físicamente el sistema, teniendo en cuenta la funcionalidad entre cada nodo de cómputo. Puede describir diferentes configuraciones de red, incluidas las configuraciones para pruebas y para simulación, además de representar una correspondencia entre la arquitectura de software y la arquitectura del sistema (hardware).(PRESSMAN, 2010)

La figura muestra el diagrama de despliegue del sistema que se propone.

Figura 11Diagrama de despliegue

El diagrama establece la relación existente entre los nodos:

PC_Cliente: espacio de trabajo por el cual el cliente interactúa con la aplicación. Esta computadora debe tener instalado el Sistema Operativo Windows o GNU/ Linux, como navegador Web Internet Explorer o Mozilla Firefox 2.0 como mínimo.

Servidor Web: se utilizará el servidor Web Apache más el lenguaje de programación PHP. Contiene toda la información referente a la aplicación.

Servidor Base de datos: se utilizará PostgreSQL como base de datos que permitirá almacenar toda la información de la aplicación.

3.2 Implementación

La implementación del componente parte del diseño anteriormente descrito. Sigue un enfoque incremental como plantea la metodología ProDeSoft. En esta fase se desarrollan las clases y subsistemas encontrados durante el diseño además de integrar los componentes involucrados. Durante la Implementación se mantiene como principio la estandarización y reutilización de códigos y componentes.

3.3 Estándares de codificación

Los estándares de codificación son un conjunto de reglas de notación específico de cada lenguaje quecomprende todos los aspectos de la generación de código. Es muy importante en cualquier producto, principalmente si en su desarrollo intervienen muchos desarrolladores. La legibilidad del código fuente, repercute directamente en el entendimiento que pueda tener otro programador del mismo, aspecto crucial ya que todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades.(Hurtado, Crespo y Terrer 2015).

En la implementación del módulo se utilizaron los siguientes estándares de codificación propuestos en los documentos “Estándar de Codificación para PHP” v 2.0, y “Estándar de Diseño de Interfaces para las aplicaciones de Gestión” v 1.0, elaborados por el Comité de rol de Lógica de Negocio de la XETID. Estos contienen, por ejemplo:

Organización del código	• El código en una página se organizará por bloques • La indentación se realizará solamente con tabulaciones, no debe utilizarse nunca los cuatro espacios
Máximalongitud de las líneas	• Todas las líneas deben estar limitadas a un máximo de setenta y nueve (175) caracteres • En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas
Líneas en blanco	• Separar las funciones de alto nivel y definiciones de

	clases con dos líneas en blanco • Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco • Se puede utilizar líneas en blanco escasamente para separar secciones lógicas
Importaciones	• Las importaciones deben estar en líneas separadas • Siempre deben colocarse al comienzo del archivo • Deben quedar agrupadas de la siguiente forma: 1. Importaciones de la librería estándar 2. Importaciones terceras relacionadas 3. Importaciones locales de la aplicación / librerías • Cada grupo de importaciones debe estar separado por una línea en blanco
Codificaciones	• Utilizar la codificación UTF-8 • Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”
Comentarios	• Los comentarios deben ser oraciones completas • Si un comentario es corto el punto final puede omitirse • Los comentarios de una línea para aclaraciones del código aparecerán seguidos de los caracteres “//” en caso de código JavaScript
Espacios en blanco en expresiones y	• Evitar utilizar espacios en blanco en las siguientes

sentencias	situaciones: - Inmediatamente dentro de paréntesis, corchetes y llaves - Inmediatamente antes de una coma, un punto y coma o dos puntos - Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función • Deben rodearse con exactamente un espacio los siguientes operadores binarios: - Asignación (=) - Asignación de aumentación (+=, -=, etc.) - Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, isnot) - Expresiones lógicas (and, or, not).

3.3 Pruebas

El proceso adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este se ejecutan pruebas dirigidas a componentes del software y al sistema en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos. Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones esperadas y anormales, evaluando los resultados, es por eso que la realización de las mismas al software es un factor de vital importancia.(PRESSMAN, 2010)

Las técnicas de evaluación dinámica proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas,tal y como dice el (Programa de estudio de nivel básico para probador certificado, 2017)se agrupan en:

1. Pruebas de caja negra (Black-Box Testing): estas son pruebas funcionales. Parten de los requisitos funcionales para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro. Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación.

2. Pruebas de caja blanca (White-Box Testing): Son pruebas estructurales. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas. Las pruebas no funcionales incluyen, pero sin limitarse a ello, pruebas de rendimiento, carga, estrés, pruebas de usabilidad, prueba de mantenibilidad, pruebas de fiabilidad y pruebas de portabilidad.

3.3.2 Estrategia de pruebas

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 (XETID,2017). 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.(XETID,2017)

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. 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.(XETID,2017)

3.4 Diseño de los casos de prueba (DCP)

3.4.1 Casos de prueba para la Prueba de funcionalidad.

Se realizaron pruebas manuales a la funcionalidad de cada requisito, siguiendo una serie de pasos

establecidos en el plan de pruebas. El probador realizó 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 (No conformidad) con todo detalle: descripción, datos utilizados, capturas de pantalla, para facilitar la solución.

Diseño de Caso de Pruebapara la funcionalidad Adicionar prueba

Id del escenario	Escenario	Dirección	Puerto	Usuario	Clave	Nombre BD	Estándar	Elementos	Estado	Esquema	Respuesta del sistema
EP 1.1	Adicionar prueba correctamente.	V(10.12.170.216)	V(5432)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	El sistema muestra el siguiente mensaje de confirmación *"La prueba ha sido adicionada satisfactoriamente."*.
EP 1.2	Adicionar prueba incorrectamente existiendo datos de conexión erróneos.	I(10.12.12.12)	V(5432)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	El sistema muestra el siguiente mensaje de error *"Los datos de conexión son incorrectos."*.
		V(10.12.170.216)	I(222)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	
		V(10.12.170.216)	V(5432)	I(que el usuario no esté registrado en la BD)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	
		V(10.12.170.216)	V(5432)	V(asd)	I(que la contraseña no sea la del usuario)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	
		V(10.12.170.216)	V(5432)	V(asd)	V(asd)	I(cuando el nombre de la BD no exista)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	
EP 1.3	Adicionar prueba incorrectamente existiendo campos con valores incorrectos.	I(asdasdasd)	V(5432)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	El sistema muestra el siguiente mensaje de error "El formato establecido es:255.255.255.255.".
		V(10.12.170.216)	I(1234567)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	El sistema muestra el siguiente mensaje de error "El tamaño máximo requerido es 5 dígitos.".
		V(10.12.170.216)	V(5432)	I(asdasdasdasd asdfdsfsdfsdfsd fsdfsdfsfsfsf)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos, Indexado)	V(Registrado)	V(mod_persona)	El sistema muestra el siguiente mensaje de error "El tamaño máximo requerido es 100 caracteres.".
EP 1.3	Cancelar la operación.	V(10.12.170.216)	V(5432)	V(asd)	V(asd)	V (prueba)	V(XETID v1)	V(Nomenclatura, Tipo de datos,	V(Registrado)	V(mod_persona)	El sistema anula las acciones realizadas y abandona la interfaz **Adicionar prueba.**
EP 1.5	Mostrar ayuda detallada.	NA	NA	NA							Se muestra la ayuda detallada de la funcionalidad.

Figura 12Diseño de caso de prueba para la funcionalidad Adicionar Prueba

3.4.2 Casos de prueba para la Prueba Unitaria

Se realizó la prueba unitaria haciendo uso 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. En dicha prueba como se puede observar en la siguiente figura no se encontraron vulnerabilidades:

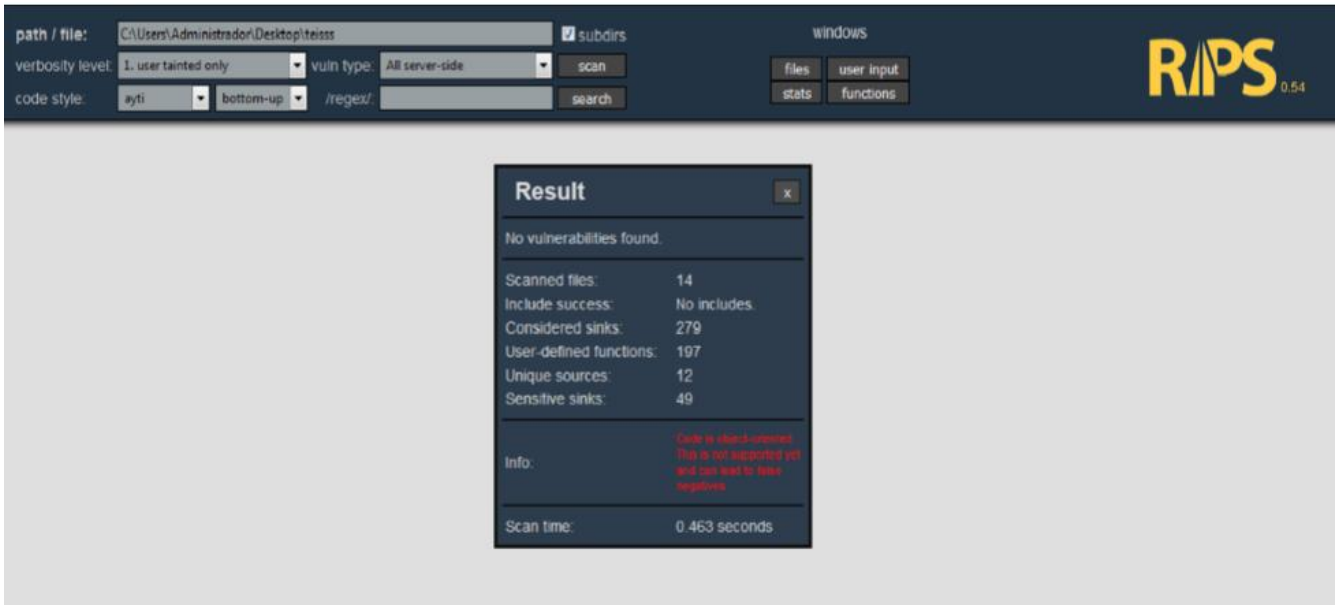

Figura 13Resultado de la prueba Unitaria

3.4.3 Casos de prueba para la Prueba de Seguridad

Al igual que las pruebas funcionales observadas anteriormente, los usuarios de negocios quieren la seguridad como un aspecto necesario de la calidad general. Una parte crítica de la creación de este compromiso es comunicarse con ellos acerca de los riesgos y beneficios, y establecer objetivos mutuamente acordados. Las actividades que se pueden realizar para hacer las pruebas de seguridad son diversas y se orientan a varios ámbitos, especialmente en lo relativo a asegurar el funcionamiento y disponibilidad de los servicios web y contenidos publicados. Al realizar la prueba de seguridad a través de la herramienta RIPS no se detectaron violaciones de seguridad. Ver **¡Error! No se encuentra el origen de la referencia.**.

Nota: La herramienta RIPS es la encargada 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 LDAP.
- Inyección SQL.
- Inyección XPath.

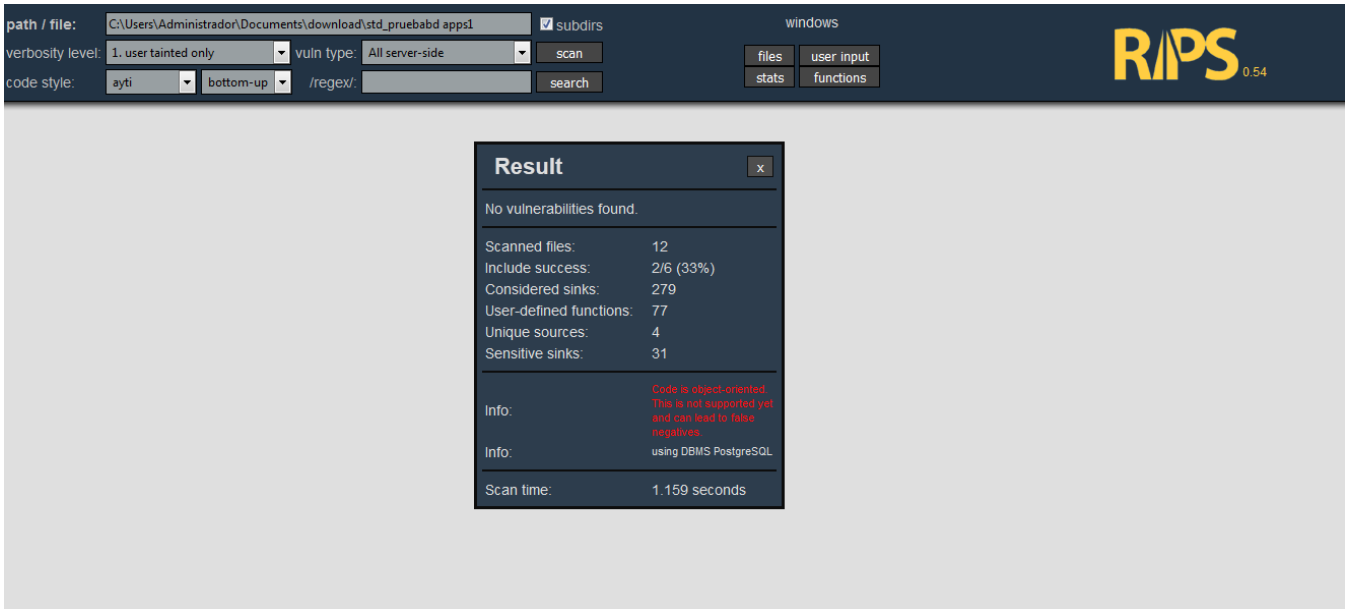

Figura 14Resultado de la Prueba de Seguridad

Se realizaron un total de 3 iteraciones que arrojaron como resultados un total de siete (7) no conformidades, divididas en dos (2) de ortografía, dos (2) de redacción, dos (2) de funcionalidad y una (1) de validaciónpara la primera iteración, las cuales quedaron resueltas. En una segunda iteración se identifican tres (3) nuevas no conformidades siendo dos (2) de funcionalidad y una (1) de validación, las cuales fueron resueltas. En una tercera iteración no se identifican nuevas inconformidades, obteniendo,

de esta manera, resultados satisfactorios. La siguiente gráfica, muestra los resultados antes descritos:

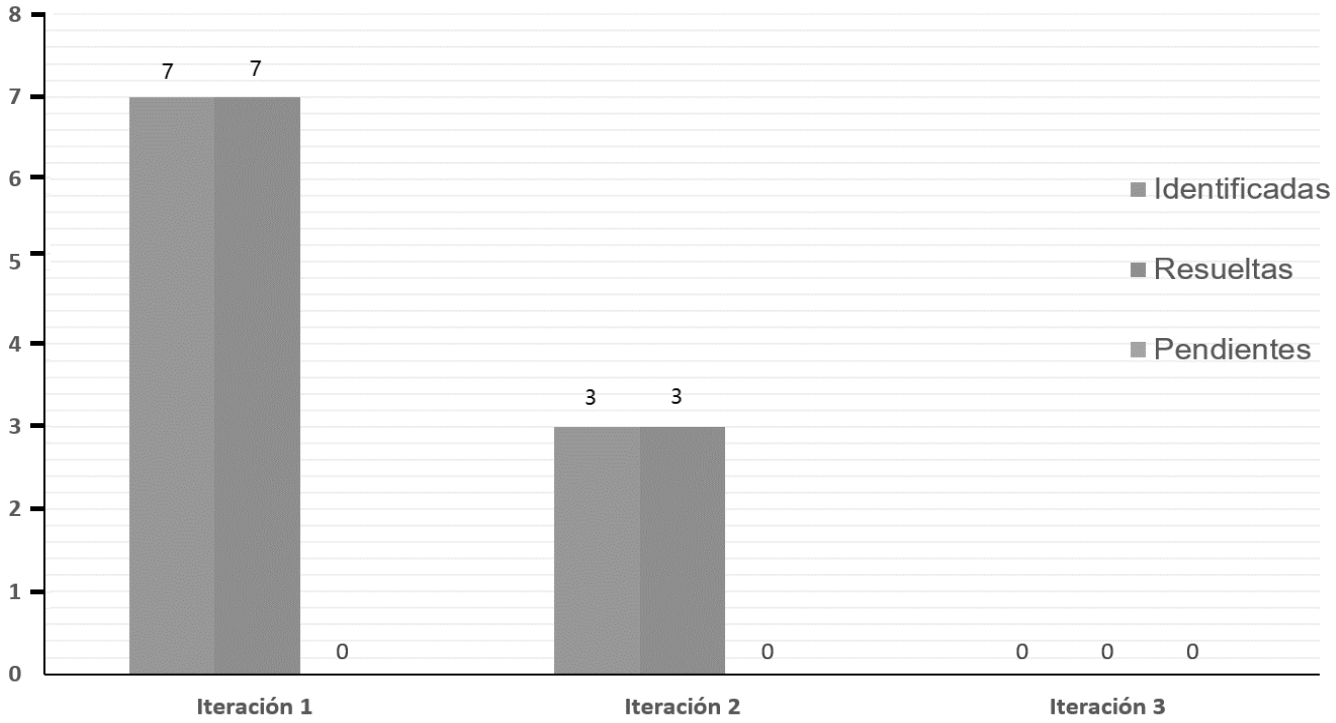

Figura 15Gráfica de No Conformidades por iteraciones

Las no conformidades de funcionalidad, aparecían al crear una nueva prueba, y al seleccionar uno de los elementos a revisar (indexado, nomenclatura o tipo de datos). En el primer caso, el sistema permitía adicionar una prueba, aunque esta ya había sido adicionada con anterioridad por el probador, no permitiendo una correcta evolución del RF2: Mostrar seguimiento de la prueba de arquitectura de la Base de Datos SQLite. Este se resolvió agregando en *Adicionar ()*una condicional que permitiera saber si la prueba ya había sido adicionada y si esto sucedía no se le permitiría al probador adicionarla mostrando un mensaje de error. En el segundo caso, al marcar uno de los elementos a revisar de la arquitectura (indexado, nomenclatura o tipo de datos),no permitía seleccionar más de un elemento a la vez. Esta no conformidad fue resuelta creando una clase para cada tipo de elemento donde almacenaría en un id cada uno de los elementos dado su tipo.

Las no conformidades de validación fueron encontradas en el formulario de adicionar una prueba donde se permitía que los nuevos parámetros incorporados no fuesen obligatorios. Esto se solucionó variando el id de cada uno de ellos a obligatorio que inicialmente estaban en nulo.

3.5 Interfaces principales del módulo de recomendación de ejercicios

Una vez finalizado el desarrollado del componente, es posible visualizar las interfaces principales de su funcionamiento, donde se observa el resultado obtenido durante la implementación. En la siguiente imagen se puede evidenciar el lugar que ocupa el componente CEAD SQLite dentro del marco de trabajo Zeolides, se listan una serie de BD ya agregadas en las que se puede apreciar su estado, el estándar con el que se está comparando, los elementos a probar(tipo de datos, indexado y/o nomenclatura), los esquemas a probar y la fecha de registro. Entre las opciones de la barra de herramientas se pueden apreciar cuatro de los cinco requisitos funcionales solicitados por el cliente, gestionar una base de datos, mostrar seguimiento de las mismas, ver los detalles de la prueba y realizar la prueba.

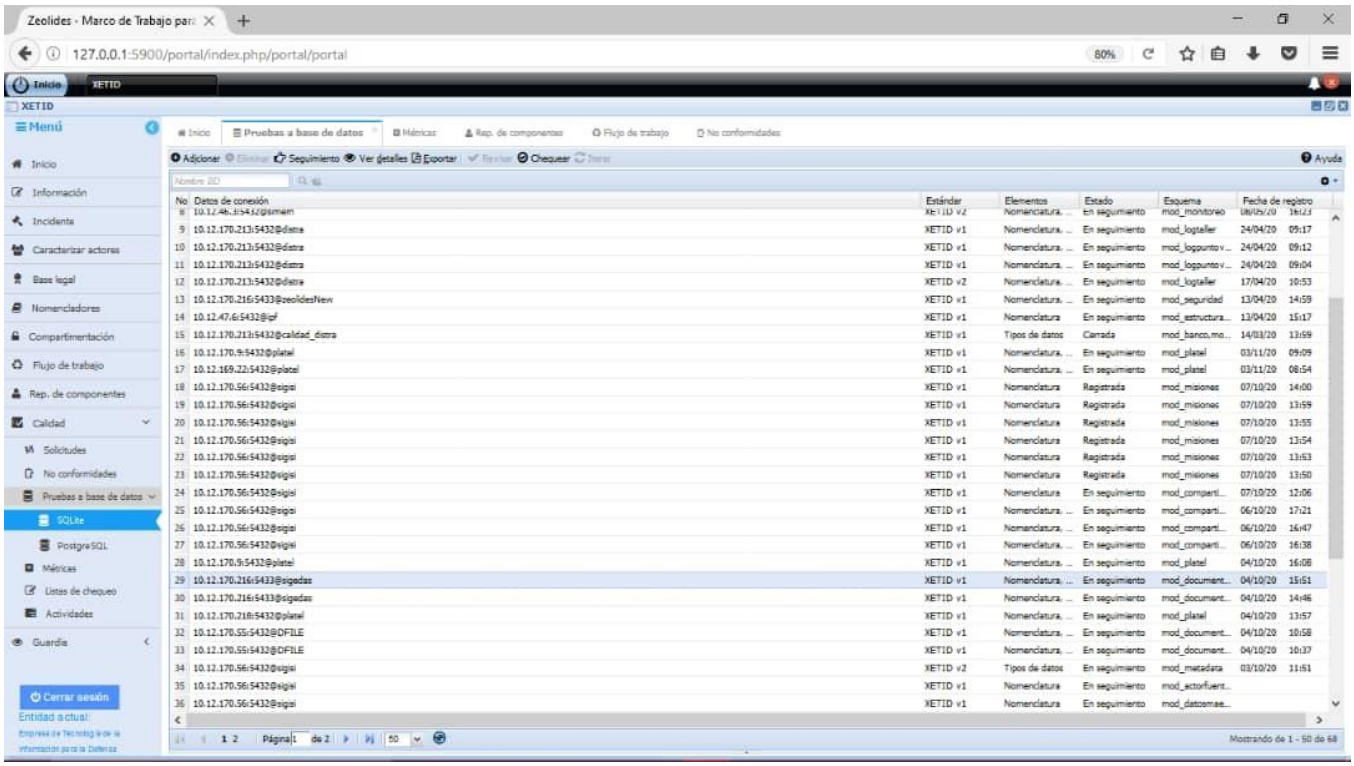

Figura 16Interfaz de usuario seleccionando la base de datos a probar

Una vez seleccionada una BD, como en la imagen anterior, se presiona el botón Revisar, se muestra una ventana emergente solicitando al usuario esperar unos minutos y luego se muestra una nueva ventana en la parte inferior que lista los errores encontrados, dando cumplimiento al quinto requisito funcional.

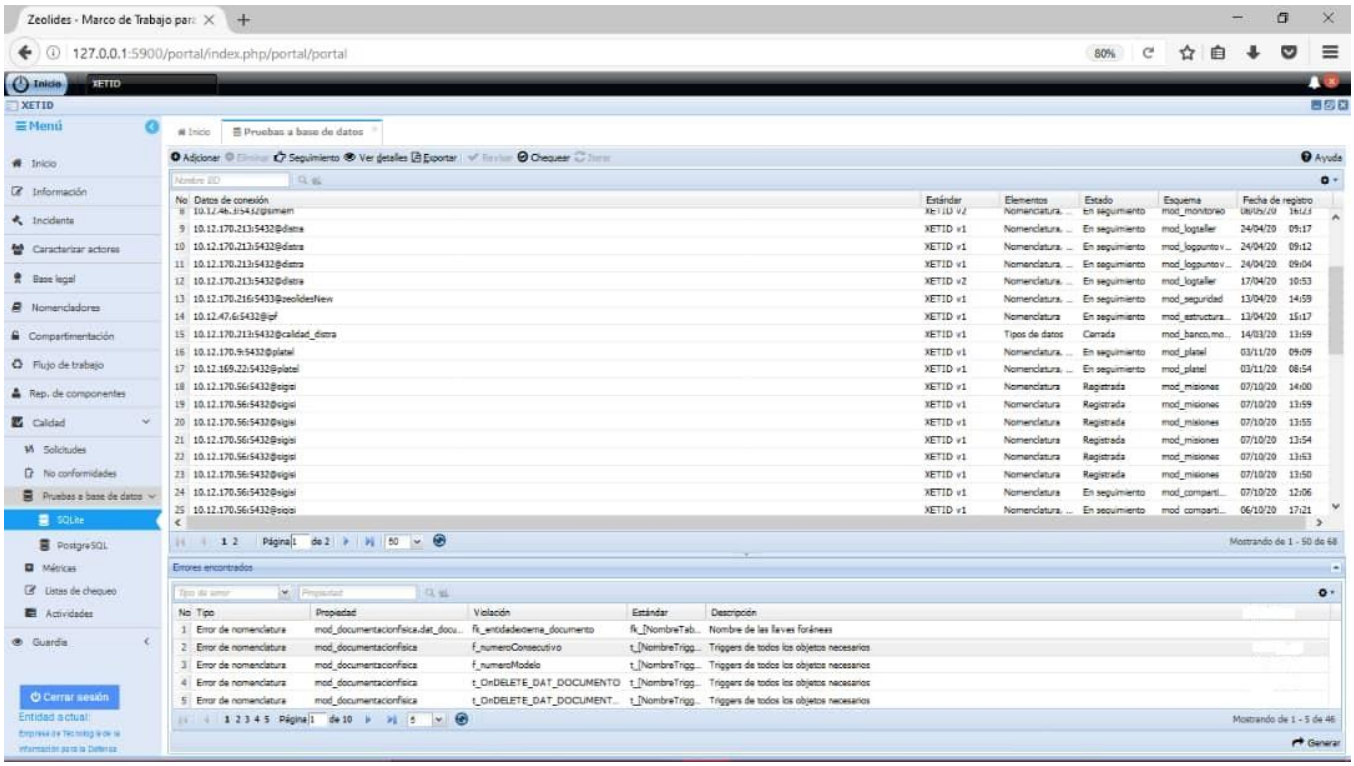

Figura 17Interfaz de usuario listando errores encontrados

Conclusiones

En este capítulo se mostró el diagrama de despliegue dando a conocer así la arquitectura del componente desde el punto de vista de los artefactos del software.Teniendo en cuenta todo lo abordado se pudo concluir que la utilización de los estándares de código facilita una mejor comprensión del sistema desarrollado si se desean realizar cambios.Mediante las pruebas realizadas se verificó el correcto desarrollo de las funcionalidades propuestas, dándole cumplimiento a las solicitudes requeridas por el cliente.

ConclusionesGenerales

Con el desarrollo del Componente de revisión de estándar de arquitectura de datos para el gestor SQLite, se cumplió con los objetivos trazados en la investigación, obteniendo como resultado un componente integrado al marco de trabajo Zeolides, capaz de comprobar automáticamente la arquitectura de las bases de datos del gestor *SQLite* en las aplicaciones informáticas. Se obtienen los siguientes resultados relevantes:

- Con la sistematización del marco teórico de la investigación se identificaronlos principales conceptos asociados al componente de revisión de arquitectura de datos del sistema gestor de base de datos *SQLite* y las relaciones entre estos, lo que permitió alcanzar una mayor comprensión de la propuesta de solución.

- El análisis del sistema homólogo y de los estándares de arquitectura de base de datos permitió identificar las tendencias en cuanto al desarrollo de herramientas revisión de estándar de arquitectura de datos e identificar las deficiencias que impiden sea utilizada.

- La integración de diversas áreas del conocimiento como son la ingeniería y gestión de software, base de datos, programación, entre otras, permitió el análisis, diseño e implementación del Componente de revisión de estándar de arquitectura de datos para el gestor SQLite.

La solución fue validada a partir de la definición correcta de una estrategia de pruebas, que permitió comprobar el correcto funcionamiento del Componente de revisión de estándar de arquitectura de datos para el gestor SQLite a partir de los requerimientos definidos por el cliente.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 SQLite, una herramienta que cumple con las necesidades del cliente y satisface los requerimientos del problema planteado.

Recomendaciones

Para el desarrollo de futuras investigaciones relacionadas con la presente se propone:

- Modificar los componentes CEAD SQLite y CEAD Postgree de forma tal que sea adaptable para todos los estándares. Así puede ser utilizado por cualquier empresa que revise la calidad de las bases de datos de los productos informáticos bajo sus propios estándares.
- Implementar un Componente de revisión de estándar de arquitectura de datos para el gestor MySQL e integrarlo a los anteriores.

Referencias Bibliográficas

About DbUnit. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://www.dbunit.org/.

About the Apache HTTP Server Project - The Apache HTTP Server Project. [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://httpd.apache.org/ABOUT_APACHE.html.

ALMONACID INZUNZA, DAMARIS JEMIMA, 2016. *Comparación entre gestores de bases de datos relacionales* [en línea]. S.l.: s.n. [Consulta: 12 noviembre 2019]. Disponible en: http://repositoriodigital.ucsc.cl/handle/25022009/1092.

Apache JMeter - Apache JMeter™. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://jmeter.apache.org/.

Arquitectura de datos: la base de una estrategia diferenciadora. *cognodata* [en línea], 2019. [Consulta: 9 diciembre 2019]. Disponible en: https://www.cognodata.com/blog/arquitectura-datos-estrategia-diferenciadora/.

CARLOS MARIO ZAPATA J y CRHISTIAN DE JESÚS CARDONA VELÁSQUEZ. M.SC., 2011. *Comparación de las características de algunas herramientas de software para pruebas de carga.pdf* [en línea]. S.l.: s.n. [Consulta: 14 noviembre 2019]. Disponible en: http://www.bdigital.unal.edu.co/28847/1/26734-93661-1-PB.pdf.

DAVID GONZÁLEZ MÁRQUEZ, 2019. *Creación de una práctica de bases de datos relacionales con SQLite.pdf* [en línea]. abril 2019. S.l.: s.n. [Consulta: 12 noviembre 2019]. Disponible en: https://repositorioinstitucional.ceu.es/bitstream/10637/10242/1/Creacion_DavidGonzalez_2019.pdf.

DBUnit | CAFEBABE. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: http://cafebabe.es/?p=151.

DORIS MARÍA MERA MERO y CARLOS VILLAMARIN ZAMBRANO, 2012. *CATEGORÍAS DE BENEFICIOS DE ESTANDARES Y PROCEDIMIENTOS.* [en línea]. Maestria. Facultad de Ciencias Informáticas Ingeniería en sistemas SEGURIDAD INFORMATICA: UNIVERSIDAD TECNICA DE MANABI. [Consulta: 12 mayo 2019]. Disponible en: http://bitchanger.live/10770797-Universidad-tecnica-de-manabi-facultad-de-ciencias-informaticas-ingenieria-en-sistemas-seguridad-informatica-tema.html.

ERICH GAMMA, RALPH JOHNSON y JOHN VLISSIDES, 1995. *Design Patterns - Elements of*

Reusable Object-Oriented Software [en línea]. 2003. Madrid: Addison Wesley. [Consulta: 20 febrero 2020]. Disponible en: https://groups.google.com/forum/#!topic/uptospnfi/q_MM_b6nAsg.

Free JavaScript training, resources and examples for the community. [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://www.javascript.com/.

GROS, B., ESCOFET, A. y MARIMÓN, M., 2016. Los patrones de diseño como herramientas para guiar la práctica del profesorado / The design patterns as tools to guide the practice of teachers. *Revista Latinoamericana de Tecnología Educativa - RELATEC*, vol. 15, no. 3, pp. 11-25. ISSN 1695-288X. DOI 10.17398/1695-288X.15.3.11.

HURTADO, E., CRESPO, R. y TERRER, R., 2015. Estándar de codificación para PHP. 2015. S.l.: s.n.

Introducción a XML. *Documentación web de MDN* [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/XML/Introducci%C3%B3n_a_XML.

JORGE SANCHEZ ASENJO, 2008. Gestión de Base de Datos.Sistemas gestores de bases de datos.pdf. *Gestion de Base de Datos* [en línea]. módulo del ciclo de FP de Grado Superior, Administración de Sistemas Informáticos en Red. S.l.: s.n., pp. 15-20. Disponible en: http://www.cartagena99.com/recursos/programacion/apuntes/IntroBBDD.pdf.

JOSÉ MIGUEL REYES PÉREZ, 2017. *CEAD. Componente de revisión de arquitectura de datos de la Empresa de Tecnología de la Información para la Defensa (XETID).* S.l.: Universidad de las Ciencias Informáticas.

LÓPEZ HERRERA PATRICIA, 2016. *Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL.pdf* [en línea]. S.l.: s.n. [Consulta: 12 noviembre 2019]. Disponible en: https://core.ac.uk/download/pdf/80528621.pdf.

MARCELLO VISCONTI y HERNÁN ASTUDILLO, 2014. Fundamentos de Ingeniería de Software. [en línea]. pdf. Departamento de Informática Universidad Técnica Federico Santa María. [Consulta: 19 febrero 2020]. Disponible en: http://roa.ult.edu.cu/bitstream/123456789/401/1/08-Patrones.pdf.

MARIA JESUS RAMOS, ALICIA RAMOS y FERNANDO MONTERO, 2016. *Sistemas Gestores de Bases de Datos* [en línea]. S.l.: MC Graw Hill. Ciclos Formativos.Grado Superior. Disponible en: www.mhe.es/cf/informatica. 15/11/2019

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/.

MAURICIO GUANCHE CAÑIZARES, DANIEL ENRIQUE PÉREZ ROJAS y JOSÉ MIGUEL REYES PÉREZ, 2017. CEAD. Componente de revisión de arquitectura de datos de la Empresa de Tecnología

de la Información para la Defensa (XETID). *Caliprot 2017*, pp. 5.

NetBeans IDE 8.2 Release Information. [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://netbeans.org/community/releases/82/.

NORMAS ISO 25000. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://iso25000.com/index.php/normas-iso-25000.

OMG | Object Management Group. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://www.omg.org/.

Oracle SQL Developer Downloads. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://www.oracle.com/tools/downloads/sqldev-v192-downloads.html.

PHP: ¿Qué es PHP? - Manual. [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://www.php.net/manual/es/intro-whatis.php.

PostgreSQL. [en línea], 2019. [Consulta: 4 diciembre 2019]. Disponible en: https://www.postgresql.org/about/news/1481/.

PRESSMAN, 2010. *Ingeniería del Software. Un Enfoque Práctico - .7ed.Pressman.PDF* [en línea]. S.l.: s.n. [Consulta: 13 noviembre 2019]. Disponible en: http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF.

Programa de estudio de nivel básico para probador certificado, 2017. [Consulta: 4 marzo 2020]Disponible en: https://www.jenkins.com/testers/programa/0258

PULIDO, J.A.M., 2019. Web usage mining aplicado a servidores web apache. *Perspectiv@s*, vol. 14, no. 13, pp. 25-30. ISSN 1996-1952.

Ruiz Gomez, Miriam, iglesias rodriguez, Francisco Javier, 2017.,Creación de una base de datos estandarizada ara la flota eólica de Brasil, Repositorio Institucional Universidad de Oviedo 2017[Consulta: 14 agosto 2020]Disponible en: https://www.digibuo.uniovi.es/dspace/105143808.

Server 2019 | Microsoft. *Microsoft SQL Server - ES (Español)* [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://www.microsoft.com/es-es/sql-server/sql-server-2019.

SERVICIO DE INFORMÁTICA UNIVERSIDAD DE ALICANTE, 2020. Introducción al Modelo vista controlador (MVC). [en línea]. [Consulta: 19 febrero 2020]. Disponible en: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

SOMMERVILLE, 2011. Ingeniería de Software. [en línea]. [Consulta: 18 febrero 2020]. Disponible en: http://artemisa.unicauca.edu.co/~cardila/Libro_Sommerville_9.pdf.

SQLITE CONSORTIUM, 2019. SQLite Home Page. *SQLite* [en línea]. [Consulta: 14 noviembre 2019].

Disponible en: https://www.sqlite.org/index.html.

SQLite Database System Design and Implementation (Second Edition, Version 2) [en línea], 2019. S.l.: s.n. [Consulta: 14 noviembre 2019]. Disponible en: https://books.google.com/books/about/SQLite_Database_System_Design_and_Implem.html?hl=es&id=yWzwCwAAQBAJ.

Tatés Perez, Omar. Implementacion de una aplicación movil Android para el seguimiento de asistencia de los estudiantes de la CISICI, utilizandio Android Studio. Tesis de grado de la Universidad del Norte, Ibarra,Ecuador, Facultad de ingeniería de ciencias aplicadas.Agosto 2018Disponible en:http://repositorio.unt.edu.ec/bitstream/123456789/8552/04%201sc%20475%20trabajo.

Tecnologias y Ciberseguridad. Marco de Trabajo para el Desarrollo de Aplicaciones Web (Zeolides).pdf [en línea], 2019. 2019. S.l.: s.n. [Consulta: 4 diciembre 2019]. Disponible en: https://www.xetid.cu/Descargas/20%20TyC.pdf.

Ventajas y desventajas de Oracle SQL. *Techlandia* [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://techlandia.com/ventajas-desventajas-oracle-sql-lista_464674/.

Ventajas y desventajas de usar SQL-Server – Site Title. [en línea], 2019. [Consulta: 14 noviembre 2019]. Disponible en: https://tareaofi10.wordpress.com/2017/11/10/ventajas-y-desventajas-de-usar-sql-server/.

Visual Paradigm Frequently Asked Questions. [en línea], 2019. [Consulta: 4 diciembre 2019]. Disponible en: https://www.visual-paradigm.com/support/faq.jsp.

What is JavaScript? *Documentación web de MDN* [en línea], 2019. [Consulta: 20 noviembre 2019]. Disponible en: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript.

XETID, 2017. *Proceso de Desarrollo de Software v1.7*. 2017. S.l.: s.n.

YAIMA PUIG, 2019. El proceso de informatización de la sociedad cubana es un hecho. *Granma.cu* [en línea]. [Consulta: 3 diciembre 2019]. Disponible en: http://www.granma.cu/cuba/2019-02-18/el-proceso-de-informatizacion-de-la-sociedad-cubana-es-un-hecho-18-02-2019-22-02-12.

YIXANDER DE LA PAZ MILÁN, 2011. *Desarrollo de un componente de monitoreo para el Servidor de Gestión PostgreSQL,* [en línea]. Trabajo de Diploma par a optar por el título de Ingeniero en Ciencias Informáticas. S.l.: Universidad de las Ciencias Informáticas. [Consulta: 12 marzo 2019]. Disponible en: https://repositorio.uci.cu/jspui/handle/ident/TD_04429_11.