INVESTIGACIÓN

Plataforma para la gestión, administración y monitoreo de servidores PostgreSQL

Platform for management, administration and monitoring of PostgreSQL servers

Ing. Osmar Capote Vázquez[1*], Ing. Barbara Dayanne Acosta Caldevilla[2], Ing. Jesús Martínez Méndez[3]

Recibido: 11/2017 | Aceptado: 02/2018

PALABRAS CLAVE
PostgreSQL
Servidores
Base de datos
Sistemas Gestores de base Datos (SGBD)

RESUMEN
Dentro de los servidores de base de datos más usados actualmente por su código abierto, simplicidad y seguridad se encuentra PostgreSQL. Actualmente no se cuenta con una aplicación web que integre las funcionalidades del ecosistema del gestor de bases de datos PostgreSQL. La actual investigación pretende dar una solución práctica, pues se está desarrollando un sistema informático que permita integrar la gestión, administración y monitoreo del gestor de bases de datos PostgreSQL. Es por ello que el objetivo principal sea la creación de una herramienta libre y que funcione sobre la web para gestionar, administrar y monitorear los servidores PostgreSQL. También se realiza una descripción de las tecnologías libres a usar en su desarrollo y de la metodología de desarrollo de software.

KEYWORDS
PostgreSQL
Servers
Database
Data Base Management Systems (DBMS)

ABSTRACT
Among the most commonly used database servers for its open source, simplicity and security is PostgreSQL. Currently, there is no web application that integrates the ecosystem functionalities of the PostgreSQL database manager. The current research aims to provide a practical solution, since a computer system is being developed to integrate the management, administration and monitoring of the PostgreSQL database manager. That is why the main objective is the creation of a free tool that works on the web to manage, manage and monitor PostgreSQL servers. There is also a description of the free technologies to be used in their development and the methodology of software development.

Introducción

En las últimas décadas, la humanidad ha sido testigo de un vertiginoso avance en las tecnologías, la informática y las telecomunicaciones. El aumento del desarrollo tecnológico ha contribuido a que las Tecnologías de la Información y las Comunicaciones (TIC) impulsen la vida cotidiana y profesional de las personas a nivel mundial. El mayor impacto de las TIC

1* Universidad de las Ciencias Informáticas (UCI). La Habana, Cuba. ocapote@uci.cu
2 Universidad de las Ciencias Informáticas (UCI). La Habana, Cuba. bdacosta@estudiantes.uci.cu
3 Universidad de las Ciencias Informáticas (UCI). La Habana, Cuba. jmendez@estudiantes.uci.cu

se encuentra en el mundo empresarial, asumiendo el liderazgo las grandes empresas transnacionales principalmente en países desarrollados (Alderete, 2012).

Con el acelerado avance de la sociedad moderna y el desarrollo empresarial alcanzado en esta etapa, surge la necesidad de almacenar los datos. Desde que los datos dieron el salto de lo analógico a lo digital, su crecimiento ha aumentado considerablemente. Tras el punto de partida, con el nacimiento de Internet, el incremento de los datos se ha visto impulsado por la proliferación de los dispositivos móviles y está destinado a dar un salto aún mayor con el Internet de las Cosas (IoT). Así pues, en muy poco tiempo se ha pasado de hablar de terabytes (1.000 gigabytes) a zettabytes (1 billón de gigabytes) (Parnell, 2015).

Interactuar con los datos, clasificarlos en pequeños grupos que describan sus características principales, basándose en la similitud o diferencia entre ellos es una de las principales funciones de las bases de datos, las cuales necesitan de servidores que las gestionen. Los servidores de base de datos surgen con motivo de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos y que además requieren compartir la información con un conjunto de clientes de una manera segura (Barzanallana, 2000).

Dentro de los servidores de base de datos más usados actualmente por su código abierto, simplicidad y seguridad se encuentra PostgreSQL. En la actualidad muchos sistemas en los que la consistencia y persistencia de base de datos es fundamental, utilizan como mejor opción los servidores de base de datos PostgreSQL, delegando en los mismos la responsabilidad de la gestión y el almacenamiento de la información. PostgreSQL es un sistema gestor de bases de datos (SGBD) objeto-relacional, distribuido bajo licencia PostgreSQL y con su código fuente disponible libremente. Es el SGBD de código abierto más potente del mercado, utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (PostgreSQL-es 2010).

Cuba se encuentra inmersa en un desarrollo tecnológico, donde el principal objetivo es informatizar todas las esferas de la sociedad y lograr la soberanía tecnológica. Por ello, los SGBD que se utilizan generalmente son los que están bajo la licencia de código abierto, como son los gestores PostgreSQL y MySQL. Lo que ha contribuido a que sus empresas tengan una mayor productividad y calidad en los productos y servicios que brindan. Son varias las que solicitan los servicios de la Universidad de las Ciencias Informáticas de Cuba (UCI) con vistas a informatizar y posteriormente analizar todo un cúmulo de datos que poseen las mismas. Estos datos crecen día a día y resulta de vital importancia mantenerlos organizados y poder acceder a ellos tan rápido como sea posible, acciones que se pueden realizar utilizando los modernos SGBD, que cuentan con un grupo de funcionalidades que permiten manejar gran cantidad de datos eficientemente.

En la actualidad, no se cuenta con una aplicación web que integre las funcionalidades del ecosistema del SGBD PostgreSQL. Existe la necesidad de desarrollar una plataforma integral para gestionar, administrar y monitorear los servidores; que muestre el flujo de trabajo del gestor PostgreSQL. La instalación, la configuración, el monitoreo, la optimización y los mantenimientos no se realizan de forma gráfica sino a través de una terminal a base de comandos. Algunos softwares de forma independiente realizan una u otra función.

Materiales y métodos

Se empleó el método analítico-sintético el cual posibilitó extraer e identificar conceptos, características y otros elementos de la bibliografía consultada que posteriormente ayudan a establecer una propuesta adecuada a las necesidades del sistema. El método inductivo-deductivo, para analizar el comportamiento y las características que poseen las herramientas para la administración de servidores PostgreSQL.

Se utilizó observación como método básico para recopilar información, debido a que es la técnica de investigación sobre la que se sustentan todas las demás (Hernández & Coello, 2012).

La entrevista con el cliente y los trabajadores, fue una de las fundamentales fuentes de recopilación de información, para entender qué problemas existen y saber cómo solucionarlos. Siendo empleada también en trabajadores de otros centros productivos que están inmersos en el tema de administración de servidores de PostgreSQL.

Se empleó el modelado mediante diagramas. Esto permitió reflejar la estructura, las relaciones y características de la solución propuesta. En este caso, con el uso de Business Process Model and Notation (BPMN), se facilita también el diseño de clases necesario para la implementación de la plataforma.

Resultados y discusión

Conceptos asociados

PostgreSQL: es un sistema de gestión de bases de datos objeto – relacional, distribuido bajo licencia Berkeley Software Distribution (BSD) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (Martínez, 2013).

Bases de datos: se define como un conjunto exhaustivo de datos estructurados, fiables y homogéneos, organizados independientemente de su utilización e implementación en una computadora, accesibles en tiempo real, que pueden compartir varios usuarios con necesidades de información diferentes y no predecibles en el tiempo (Camellea, 2004). También se puede definir como un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada (Date, 2001).

Sistema gestor de bases de datos: software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD). El objetivo fundamental de un SGBD consiste en suministrar al usuario las herramientas que le permitan manipular, en términos abstractos, los datos, o sea, de forma que no le sea necesario conocer su modo de almacenamiento en la computadora, ni el método de acceso empleado (Martínez, 2016).

Herramientas y lenguajes informáticos

Se utilizó como metodología de desarrollo de software ágil: XP, BPMN para el modelado del negocio, Herramienta Case: Visual Paradigm 8.0, Lenguaje de Programación: Python 3.6, IDE: Pycharm 2017.1.2, Sistema Gestor de Base de Datos: PostgreSQL 9.6 y los Frameworks: Django 1.10, jQuery 1.9.1 y Bootstrap 3.3.1.

Proceso de administración de servidores PostgreSQL

La modelación del proceso de negocio permite realizar una exploración del dominio del problema, con el fin de lograr comprensión por parte del equipo de desarrollo, de los procesos que se realizan actualmente en la entidad y la relación que existe entre estos. Durante este proceso se van determinando necesidades operacionales, así como restricciones que presenta la entidad, obteniéndose finalmente un entendimiento del negocio para dar paso a la fase inicial del sistema. Además, permite comprender las características del negocio a través de la descripción de los procesos del mismo (Espinosa, 2009).

El proceso para la administración de un servidor PostgreSQL comienza con la instalación del servidor, luego el administrador configura las opciones de este, tales como control de acceso, mediante los archivos de configuración luego reinicia el servicio mediante la consola, y posteriormente se accede al servidor, por cualesquiera de los clientes que existen. En la figura 1 se visualiza el proceso que se lleva a cabo para administrar un servidor PostgreSQL. En la tabla 1 se muestra el cronograma (ficha) de proyecto en el cual quedan plasmadas las acciones a realizar de forma ordenada y lógica de cada actividad.

Metodología

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte, se tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas informáticas y notaciones que se usarán. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías de desarrollo de software ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas (Letelier, 2006).

En esta investigación se emplea la metodología de desarrollo de software ágil XP. La cual contempla sus niveles organizativos, de desarrollo y de avances por fases: exploración, planificación y diseño del software.

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 59 - 67, 2018 tono

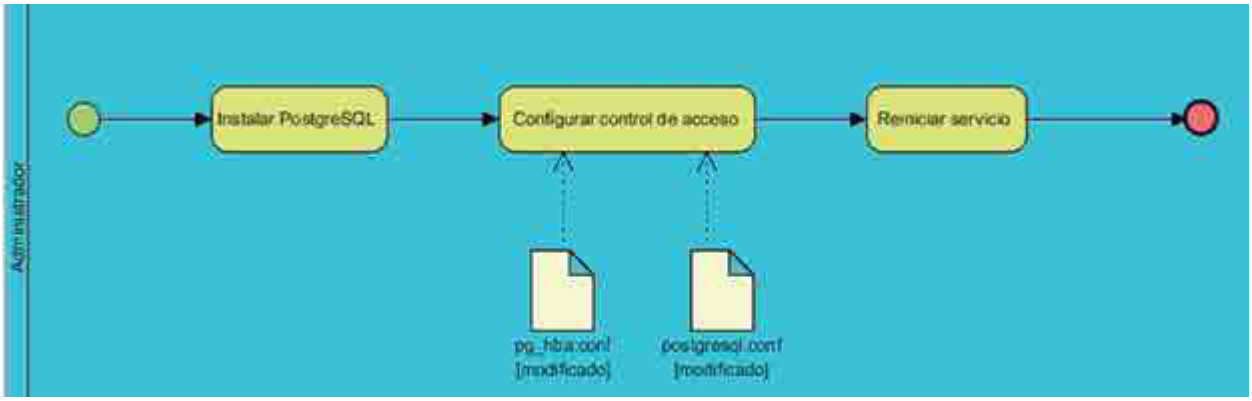

Figura 1. Proceso del negocio actual.

Ficha de proceso	
Proceso	Proceso del negocio actual para la administración de un servidor PostgreSQL
Descripción	Proceso que se realiza actualmente de forma general en las áreas donde se utilizan servidores PostgreSQL
Entradas	Usuario y contraseña del administrador para acceder al sistema
Salidas	Configuraciones de los servicios PostgreSQL en el cliente usado
Reglas del negocio	Se necesita tener previamente instalado el servidor para luego comenzar a administrarlo.
Actividades principales	
Actividad 1	Instalar PostgreSQL: el administrador debe instalar el PostgreSQL
Actividad 2	Configurar control de acceso: el administrador modifica los archivos pg_hba.conf y postgresql.conf
Actividad 3	Reiniciar el servicio: el administrador reinicia el servicio por la consola

Tabla 1. Ficha de proceso.

Exploración

Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios (HU). Los programadores estiman los tiempos de desarrollo en base a esta información. Debe quedar claro que las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración (Joskowicz, 2008).

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Simultáneamente el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Borja, 2015). Como resultado de aplicar la fase de exploración y hacer uso del artefacto que genera, se tienen 21 Historias de Usuario (HU), pues, son estas el artefacto que genera dicha fase, coincidiendo con las 21 funcionalidades a implementar en el sistema.

Planificación

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. El proyecto comienza recopilando HU las que sustituyen a los tradicionales "casos de uso". Una vez obtenidas las HU, los programadores evalúan rápidamente el tiempo de desarrollo de cada una. Si alguna de ellas

tiene "riesgos" que no permiten establecer con certeza la complejidad del desarrollo, se realizan pequeños programas de prueba, para reducir estos riesgos. Una vez realizadas estas estimaciones, se organiza una reunión de planificación con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas en los que todos estén de acuerdo. Una vez acordado este cronograma, comienza una fase de iteraciones, en donde, en cada una de ellas se desarrollan, pruebas y se instalan unas pocas "historias de usuarios". (Joskowicz, 2008)

Estimación de esfuerzo

Las estimaciones de esfuerzo asociado a la implementación de las HU la establecen los programadores utilizando como medida el punto de estimación. Un punto de estimación equivale a una semana de programación. Las HU generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración (Letelier & Penadés, 2006).

Plan de iteraciones

Las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada historia de usuario se traduce en tareas específicas de programación. Asimismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (Joskowicz, 2008).

Plan de entrega

El plan de entrega es el resultado de tomar acuerdos con el cliente sobre el contenido de la primera entrega y determinar un cronograma para las demás entregas del producto (Letelier & Penadés, 2006). El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). XP denomina a esta reunión "Juego de planeamiento" (*Planning game*), pero puede denominarse de la manera que sea más apropiada al tipo de empresa y cliente (por ejemplo, Reunión de planeamiento, "*Planning meeting*" o "*Planning workshop*"). Típicamente, el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario (Joskowicz, 2008).

En esta fase de la metodología XP se generan los siguientes artefactos: el Plan de Estimación de Esfuerzos por historia de usuario, el plan de duración de las iteraciones y por último y no menos importante el Plan de Entrega. Al aplicar la fase de planificación y hacer uso de los artefactos que genera la puesta en práctica de dicha fase, se obtiene como resultado que las 21 HU agrupadas en 3 iteraciones poseen un tiempo de demora para su realización de 26.5 semanas.

Diseño

La metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:

Simplicidad: Por ello XP propone implementar el diseño más simple posible que funcione. Se sugiere nunca adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando.

Soluciones "spike": Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "spike"1), para explorar diferentes soluciones. Estos programas son únicamente para probar o evaluar una solución, y suelen ser desechados luego de su evaluación.

Recodificación: La recodificación *refactoring* consiste en escribir nuevamente parte del código de

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 59 - 67, 2018 tono

un programa, sin cambiar su funcionalidad, a los efectos de hacerlo más simple, conciso y/o entendible. Muchas veces, al terminar de escribir un código de programa, pensamos que, si lo comenzáramos de nuevo, lo hubiéramos hecho en forma diferente, más clara y eficientemente. Sin embargo, como ya está listo y "funciona", rara vez es reescrito. Las metodologías de XP sugieren recodificar cada vez que sea necesario.

Patrones de arquitectura

Los patrones arquitectónicos abstraen los principales componentes de una arquitectura de sistema para ofrecer una solución a un problema. Por lo tanto, aplicar patrones arquitecturales permite al diseñador concentrarse en los detalles propios de su aplicación, utilizando soluciones ya existentes a los problemas recurrentes dentro de un dominio dado. También permite que su diseño sea más fácil de comprender por otros participantes del proceso al utilizar una estructura bien identificada, facilitando las tareas de documentación, comunicación y potenciales extensiones o adaptaciones del sistema a nuevos requerimientos (Ballari, 2001). Los patrones de arquitectura definen la estructura básica de una aplicación y pueden contener o estar contenidos en otros patrones. Proveen un subconjunto de subsistemas predefinidos, incluyendo reglas y pautas para su organización. Y son una plantilla de construcción (Cardoso & Camacho, 2004).

Arquitectura Cliente Servidor

Se decide utilizar la modalidad o arquitectura Cliente-Servidor porque en ella confluyen una serie de aplicaciones basadas en dos categorías que cumplen funciones diferentes: una requiere servicios y la otra los brinda. Pero que, a la vez, pueden realizar tanto actividades en forma conjunta como independientemente. En el caso del cliente es aquel que requiere un servicio del servidor. En esta categoría se realizan funciones de software basándose en el hardware, pero en caso de no tener la capacidad de procesar los datos necesarios, recurre al servidor y espera a que este le brinde los servicios solicitados. El cliente es una estación de trabajo o computadora que está conectada a una red a través de la cual puede acceder al servidor.

Entre las características fundamentales de esta arquitectura se evidencia que tanto el cliente como el servidor pueden realizar tareas en forma conjunta como separada ya que el cliente también tiene sus propias aplicaciones, archivos y bases de datos y que, además, pueden estar en la misma plataforma o en plataformas diferentes. Por otra parte, el servidor puede brindar varios servicios a la vez, tanto al mismo cliente como a clientes múltiples (Ocampo & Montoya, 2013).

Model Template View (MTV)

Se propone utilizar para el desarrollo de la aplicación el patrón de arquitectura MTV ya que permite separar los datos, la interfaz y la lógica del sistema.

El modelo: Define los datos almacenados, se encuentra en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos (Montero, 2012). En el modelo se representan las clases Acceso y Configuration.

La plantilla: Recibe los datos de la vista y luego los organiza para la presentación al navegador web. Las etiquetas que Django usa para las plantillas permiten que sea flexible para los diseñadores del *frontend*, incluso tiene estructuras de datos como *if*, por si es necesaria una presentación lógica de los datos, estas estructuras son limitadas para evitar un desorden poniendo cualquier tipo de código Python. Esto permite que la lógica del sistema siga permaneciendo en la vista (Montero, 2012). En la plantilla se representan las vistas que mostraran la interfaz por la cual el administrador gestionará los servicios PostgreSQL, ejemplo de estas son: Acceso.html, Configuration.html, Entry.html y CantidadUser.html.

La vista: Se presenta en forma de funciones en Python, su propósito es determinar qué datos serán visualizados. El ORM de Django permite escribir código Python en lugar de SQL para hacer las consultas que necesita la vista. Se encarga de tareas conocidas como el envío de correo electrónico, la autenticación con servicios externos y la validación de datos a través de formularios (Montero, 2012). En la vista se encuentran implementados los métodos que se mostrarán con la Plantilla entre estos métodos se encuentran MyAcceso, CantidadUser y Entry.

El patrón de arquitectura MTV (Figura 2), cumple perfectamente con el fin particular de cualquier framework, una estructura bien definida que da soporte a un proyecto web, también ayuda a que el proyecto sea organizado y bien desarrollado.

Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (Buschmann & Meunier, 1996). Patrones GRASP, los Patrones de Principios Generales para Asignar Responsabilidades (GRASP) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Experto en Información: Las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad. El sistema para la gestión, administración y monitoreo de servidores PostgreSQL hace uso de este patrón y se evidencia cuando se desea mostrar la cantidad máxima de usuarios conectados concurrentemente, ya que la única clase con la responsabilidad de conocer esta información es Postgre. De la misma manera sucede con la clase PostgreSQL que es la encargada de realizar las operaciones referentes a los parámetros de configuración que contempla el archivo postgres.conf.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. Este patrón se evidencia dentro del archivo view.py en la funcionalidad *access* () ya que dentro de esta funcionalidad se crean instancias de la clase Acceso y de la clase PostgreSQL.

Alta Cohesión: Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponden y mantienen una estrecha relación con el resto de las clases. Este patrón se evidencia en la clase postgre dado que necesita información de una conexión y la clase que le provee esta información es la clase Acceso.

Bajo Acoplamiento: Determina el nivel de dependencia de una clase con respecto a otras. Una clase con bajo acoplamiento no depende de muchas otras. Este patrón es utilizado por el framework Django en su interior ya que utiliza las URL como forma de separar clases y funcionalidades, esto permite que si deja de funcionar alguna de nuestras vistas no afecta a las demás.

Controlador: Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. La respuesta puede ser mostrar una vista, ejecutar un método, devolver un mensaje, etc. (Buschmann & Meunier, 1996). Se evidencia el uso de este patrón en el archivo views.py que es el encargado de definir clases y funcionalidades que controlan las vistas que se muestran en la aplicación.

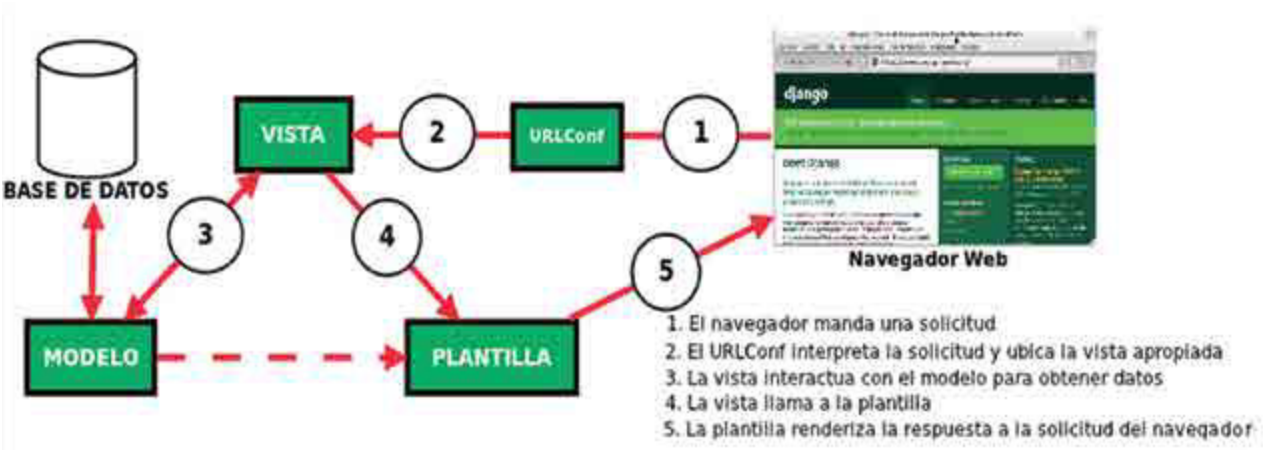

Figura 2. Patrón MTV (Fuente: Montero, 2012).

ISSN: 1813-5056 Vol. 14, No. 1 , enero-julio, pp. 59 - 67, 2018 tono

Tarjetas Clase Responsabilidad Colaboración (CRC)

Las tarjetas CRC en la metodología XP se usan para el diseño de software orientado a objetos. Estas tarjetas se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Una clase es cualquier persona, cosa, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las cosas que conoce y las que realizan, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades (Balarezo, 2013). Al aplicar la fase de diseño se genera el artefacto Tarjetas CRC, de las cuales se concluye: Como resultado de la puesta en práctica de dicha fase se llevaron a cabo 4 Tarjetas CRC referentes a las 4 clases principales por su contenido de implementación. De las mismas se detallaron en las tarjetas CRC nombre de la clase, responsabilidad (qué función realiza con la implementación de cada una de ellas) y la colaboración con otras clases.

Resultados

La herramienta Web a desarrollar facilita la gestión, administración y monitoreo de servidores PostgreSQL. Solo tendrán acceso a la aplicación los usuarios que posean el rol de administrador. La aplicación permitirá agregar servidores PostgreSQL dado una dirección IP o seleccionar uno o varios servidores que ya tenga registrados la aplicación y administrarlo. Una vez seleccionados los servidores se podrá instalar, desinstalar, iniciar, detener, reiniciar y mostrar el estado del servicio PostgreSQL. El administrador podrá configurar el puerto de escucha del servidor, el número máximo de conexiones permitidas, el tiempo máximo de autenticación de los clientes, la opción de contraseñas cifradas para la autenticación y la cantidad de conexiones reservadas para los superusuarios. La aplicación contará con la capacidad de gestionar roles de las bases de datos, gestionar clústeres de bases de datos y gestionar las BD dígase crear, eliminar, modificar y mostrar una base de datos.

Permitirá además la gestión de reglas de control de acceso al servidor y el consumo de recursos, los cuales serían, configurar la cantidad de memoria que el servidor utiliza para buffers, el número máximo de buffers temporales usados por cada sesión de la BD, la cantidad de memoria de trabajo y la de operaciones de mantenimiento. La aplicación además permitirá activar y desactivar SSL, hacer mantenimiento, hacer Backup y restauraciones de BD, monitorear el uso de los recursos a través de gráficos, optimizar y gestionar rendimiento de las BDs, instalar múltiples versiones de PostgreSQL en un mismo servidor y crear instancias del servidor. Otras de sus funcionalidades serán seleccionar qué tipo de BD va a crear ya sea relacional o geoespacial y además se administrarán objetos espaciales. También se recuperarán BD utilizando Point-in-time-recovery (PITR).

Conclusiones

Existen varias herramientas informáticas libres que permiten una adecuada gestión, administración y monitoreo del gestor PostgreSQL, pero estas carecen funcionalidades integradas. Con el desarrollo de esta herramienta se facilita el trabajo de los usuarios, proporcionándoles la fácil gestión, administración y monitoreo de los servidores PostgreSQL en una sola interfaz.

La herramienta muestra funcionalidades y las tecnologías a usar en su desarrollo, así como la metodología de desarrollo de software por la que se regirá el equipo para llevar a cabo esta tarea, además de realizar una explicación y modelado de cómo funciona actualmente el flujo del proceso de negocio en entidades, organizaciones, empresas donde hagan uso y tengan destinado en la infraestructura manejo con servidores PostgreSQL.

Referencia

Alderete, M. V. (2012). EL DEBATE: ¿Las TIC reducen la distancia entre las empresas? *Revista Iberoamericana CTS*. www.revistacts.net/elforo/484-el-debate-ilas-tic-reducen-la-distancia-entre-las-empresas

Balarezo, J. (2013). *Metodologías Ágiles, Programación Extrema XP*. Universidad Nacional de Trujillo. From hhttp://es.slideshare.net/EvelingGiselleCruzVs/metodologia-monografia

Ballari, T. (2001). *Uso de patrones arquitectónicos web para el diseño de una aplicación domótica.* Paper presented at the III Workshop de Investigadores en Ciencias de la Computación

Borja López, Y. Metodología ágil de desarrollo de software XP.

Brid-Aine, P. (2015). De terabytes a zettabytes: el crecimiento de los datos mundiales *Lenovo. Think Progress*. www.think-progress.com/es/tendencias/de-terabytes-a-zettabytes-el-crecimiento-de-los-datos-mundiales/

Buschmann, F., Meunier, R., Hans Rohnert, R., Sommerlad, P. y Stal, M. (1996). *Pattern-Oriented Software Architecture. A System of Patterns*.

Camellea (Ed.). (2004). *Gestión de Base de Datos con ADO.NET*. La Habana: Científico Técnica.

Cardoso, E. y Camacho, F. (Eds.). (2004). *Arquitecturas de software. Guía de estudio*.

Date, C. (Ed.). (2001). *Introducción a los Sistemas de Bases de Datos* (7 ed.): Pearson Educación.

Emc2Net. PostgreSQL-es. Portal en español sobre PostgreSQL from https://e-mc2.net/es/postgresql-es

Espinosa, S. (2009). Mapa de procesos. Business & Mgmt.

GuzmánFernández,I.,CastilloFigueroa,R.yFloresPérez,D.(2011).Propuestadearquitectura de una herramienta web para la administración del gestor PostgreSQL. http://rcci.uci.cu/?journal=rcci&page=article&op=download&path%5B%5D=95&path%5B%5D=89

Hernández León, R. A. y Coello González, S. (2012). *El proceso de investigación científica* (2 ed.). La Habana: Editorial Universitaria.

Joskowicz, J. (2008). *Reglas y prácticas en extreme programming* http://www.cyta.com.ar/ta0502/v5n2a1.htmi.

Letelier, P. y Penadés, C. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).

LETELIER, Patricio. y PENADÉS, Carmen. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2006. Url: hhttp://www.cyta.com.ar/ta0502/v5n2a1.htmi.

Menéndez-Barzanallana Asensio, R. (2000). Servidores de bases de datos. www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/sgbd.html

Montero, S. I. (2012). Django para perfeccionistas con deadlines.

Ocampo, P. y Montoya, A. (2013). Sistema de procesamiento de información de gestión digital de historias clínicas en entorno web (SPIGDATA_HC).

Van Rossum, G. (2009). *El tutorial de Python* Retrieved from http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf