Facultad 1

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Sistema de administración de servidores PostgreSQL V3.0 para el Centro de Telemática

Autor: Marcos do Rosário da Costa Fernandes

Tutores: MSc. Yalice Gámez Batista

Ing. Antonio Hernández Dominguez

La Habana, junio del 2020

Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: “Sistema de administración de servidores PostgreSQL V3.0 para el Centro de Telemática” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de junio del año 2020.

Marcos do Rosário da Costa Fernandes

Firma del Autor

MSc. Yalice Gámez Batista

Firma del Tutor

Ing. Antonio Hernández Dominguez

Firma del Tutor

Datos de contacto

Tutor 1:

Antonio Hernández Dominguez

Ingeniero en Ciencias Informáticas.

Centro de Telemática. Facultad 2. UCI

Email: ahdominguez@uci.cu

Tutor 2:

Yalice Gámez Batista

Máster en Informática Industrial y Automatización

Profesora Auxiliar de la Facultad 1.

Departamento de Sistemas Digitales. Facultad 1. UCI

Email: yaliceg@uci.cu

"Gracias a que hay un sentimiento que se llama amor a la patria, somos fuertes; gracias a que hay un sentimiento de amor a los semejantes y de solidaridad de todos para con todos, somos fuertes; gracias a que hay un estado mental que se llama conciencia revolucionaria, somos fuertes".

Fidel Castro Ruz, 4 de julio de 1960

Agradecimientos

Agradezco primeramente a mi familia por brindarme lo mejor de sí y por sembrar en mi la fe del triunfo.

A la dirección de la Universidad de las Ciencias Informáticas por la atención, dedicación y trato al longo de la carrera.

A mis tutores que a pesar de la distancia siguieron orientándome.

A todos mis profesores que a lo largo de esta hermosa carrera contribuyeron a mi formación profesional.

A todos mis compañeros por haber dedicado un espacio de su tiempo y que de una forma u otra aportaron su granito de arena en este trabajo.

Al estado Angolano y Cubano por haberme dado la oportunidad de estudiar esta profesión y haber puesto en mi la confianza de convertirme en un hombre de ciencias.

Dedicatoria

Este trabajo va dedicado a todos los familiares y amigos que yo y mis compañeros perdimos en busca de este sueño, especialmente a mi tío Mario José Fernandes.

RESUMEN

El Centro de Telemática adscrito a la Facultad 2 de la Universidad de las Ciencias Informáticas cuenta con la segunda versión de un sistema para administrar y monitorear de forma remota y centralizada los servidores de base de datos, PostgreSQL, el cual se encuentra actualmente gestionando servidores con importantes volúmenes de datos. El sistema hoy no cuenta con las funcionalidades necesarias para la realización de copias de seguridad de los datos, para la instalación del servicio PostgreSQL en los servidores sin la conexión de red, así como la gestión de las bases de datos relacionales y la instalación de diferentes versiones del servicio PostgreSQL en el mismo servidor. En la presente investigación se describen los resultados de los procesos de análisis, diseño, implementación y pruebas de la tercera versión del Sistema de Administración de Servidores PostgreSQL, que contribuye en la instalación remota, gestión de bases de datos relacionales y copia de seguridad por medio de Punto de recuperación en el tiempo (PITR, por sus siglas en inglés), haciendo de forma automática dichos procesos y mejorando la forma de administración de los servidores que tienen como sistema gestor de base de datos PostgreSQL, contribuyendo en la administración de los servidores de base de datos del centro. Para guiar el desarrollo de la gestión de dichos resultados se utilizó como metodología de desarrollo de software la variación de AUP para la UCI, como lenguajes de programación Python, HTML, CCS y JavaScript, como marcos de trabajo Django, y como entorno de desarrollo integrado PyCharm.

Palabras clave: PostgreSQL, seguridad de la información, instalación remota, Base de Datos Relacionales, Punto de Recuperación en el Tiempo.

ABSTRACT

The Telematics Center, attached to School 2 of the University of Informatics Sciences, has the second version of a system to remotely and centrally administer and monitor the PostgreSQL database servers, which is currently managing servers with considerable volumes of data. Nowadays, the system does not have the necessary functionalities for backing up the data, installing the PostgreSQL service on the servers without the network connection, as well as managing the relational databases and installing different versions of the PostgreSQL service on the same server. The current research describes the results of the analysis, design, implementation and testing processes of the third version of the PostgreSQL Server Administration System. It contributes to remote installation, relational databases management and making copies security through Point in Time Recovery (PITR for its acronyms in English), automating such processes, improving the way of managing servers, which use PostgreSQL as Database Management System and contributing to manage the database servers at the Center. To guide the management development of such outcomes, the Agile Unified Process (AUP) variation for the UCI was used as a software development methodology, Python, Hyper Text Marked Language (HTML), Cascading Style Sheets (CCS) and JavaScript as programming languages, Django as framework, and PyCharm as an integrated development environment.

Keywords: PostgreSQL, information security, remote installation, Relational Database, Time Recovery Point.

ÍNDICE

Índice de tablas

Índice de figuras

INTRODUCCIÓN

Las Tecnologías de Información y las Comunicaciones (TIC) tienen un papel decisivo en todos los ámbitos de la sociedad, lo que posibilita alcanzar una visión integral y globalizante, pues están presentes en todas las actividades y procesos que se realizan en las organizaciones. Con el acelerado avance de la sociedad moderna del siglo XXI 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 (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 sistemas gestores que las gestionen.

Los sistemas gestores de base de datos (SGBD), surgen por la necesidad que tienen las empresas de manejar grandes y complejos volúmenes de datos. Dentro de esos sistemas se encuentra PostgreSQL el cual es muy utilizado por muchas empresas por su código abierto, simplicidad y seguridad (Paré, 2005). Muchos sistemas en los que la consistencia y persistencia de las bases de datos es fundamental, delegan en estos servidores la responsabilidad de la gestión y almacenamiento de la información.

En Cuba es inminente un salto en el desarrollo tecnológico en el área de las tecnologías de la información y las comunicaciones. Como parte del fortalecimiento de la soberanía tecnológica cubana, el país ha desarrollado un programa para la informatización de la sociedad. En este tiene una reconocida participación la Universidad de las Ciencias Informáticas (UCI); la cual a partir del vínculo docencia - investigación - producción como modelo de formación, sirve de soporte a la industria cubana del software.

La Universidad de las Ciencias Informáticas (UCI) se desempeña como principal impulsor en la aplicación de las tecnologías libres en el país. Tiene la misión de formar profesionales comprometidos con su Patria, altamente calificados en la rama de la informática, y producir aplicaciones y servicios informáticos, a partir del vínculo docencia - investigación - producción como modelo de formación, sirviendo de soporte a la industria cubana del software (UCI, 2002).

La UCI cuenta con una amplia red de centros de desarrollo de software que ejecutan proyectos informáticos, con gran impacto en la informatización de la sociedad cubana. Entre ellos está el Centro de Telemática (TLM) adscrito a la Facultad 2 de la universidad. Este tiene como objetivo desarrollar sistemas y servicios informáticos integrales para las Telecomunicaciones y la Seguridad Informática,

altamente comprometido con la revolución, capaz de integrar los procesos docentes, productivos e investigativos con alto nivel, contando con un personal especializado en dichas áreas.

El centro desarrolla varias aplicaciones y servicios que poseen servidores de bases de datos, que utilizan como gestor de bases de datos (SGBD) PostgreSQL. Este cuenta con una aplicación web desarrollada en el centro atendiendo a la necesidad de administrar y monitorear los servidores que contienen como sistema gestor de base de datos el PostgreSQL, denominado “Sistema de Administración y Monitoreo de PostgreSQL (AD&MONPSQL)” en su versión 2.0. Este tiene como objetivo administrar y monitorear de manera centralizada los servidores de PostgreSQL. Las principales fortalezas del sistema son:

- Configuración del sistema operativo a utilizar (Ubuntu y Debian).
- Instalar, desinstalar y controlar el estado del servicio PostgreSQL.
- Monitoreo del uso de la memoria y el CPU del servidor (Caldevilla, Méndez, 2018).
- Instalación del extensor postGIS (CentOS 7 y Windows Server 2012) (Fajardo, 2019).
- Configuración de certificados SSL (CentOS 7 y Windows Server 2012) (Fajardo, 2019).
- Gestión de base de datos geoespacial (CentOS 7 y Windows Server 2012) (Fajardo, 2019).

Actualmente el centro cuenta con máquinas servidoras que hacen uso del SGBD PostgreSQL para la gestión y almacenamiento de los datos. La instalación del servicio PostgreSQL se realiza mediante la red y dependiendo de la disponibilidad de los servidores de los repositorios configurados en los sistemas operativos existentes en las máquinas, lo que hace dificulta la instalación del servicio en caso de que las máquinas no tengan conexión con los servidores de los repositorios o haya una falla en la configuración de los mismos.

Otra de las limitaciones que tiene es que las copias de respaldo se realizan de forma manual lo que provoca demoras indeseadas en el funcionamiento del sistema. El objetivo principal de las copias de respaldo es preservar información por un periodo de tiempo determinado para ser recuperada y restaurada en caso de problemas o fallas que se presenten en la base de datos a nivel físico o lógico (Carreño, 2018).

Los sistemas informáticos en la actualidad pueden ser víctimas de diversos tipos de problemas: ataques remotos por parte de hackers o de virus informáticos, destrucción física de los soportes de almacenamiento, e incluso eliminación accidental por parte de un usuario autorizado. En estos casos

se recurre manual o automáticamente a las copias de seguridad para restaurar la información perdida, minimizando así el daño en materia de datos o información.

Un problema común con las copias de seguridad es que llevan tiempo para recuperar los datos, por lo que se pierde información en ese proceso. A partir de la versión 8.0 del PostgreSQL, hay una nueva función disponible, llamada Punto de Recuperación en el Tiempo o PITR (Point In Time Recovery, por su sigla en inglés), que se basa en las nuevas características de escritura anticipada o WAL (Write-Ahead Logging, por su sigla en inglés) de PostgreSQL. PITR ayuda a mitigar el problema de la información perdida, almacenando archivos de registro adicionales de las transacciones lejos de las bases de datos principales. De esta forma posibilita que el evento que causó la corrupción de la base de datos no dañe también los archivos de registro. Luego, después de restaurar los archivos de la base de datos principal, es posible ejecutar eventos en los archivos de registro adicionales para recuperar eventos entre la copia de seguridad y el bloqueo, de hecho, es posible ejecutar los comandos en un punto arbitrario en el tiempo, de ahí la aparición de esta función (Stones y Matthew, 2006).

Para implementar PITR en los servidores, hay que configurar dos de los ficheros de configuración principales (postgresql.conf y pg_hba.conf). Estos ficheros poseen una gran cantidad de instrucciones y el manejo erróneo de los mismos, puede afectar la disponibilidad de la información en un determinado servidor. Las configuraciones de dichos archivos se realizan manualmente a través de un terminal utilizando comandos específicos para cada sistema operativo. Este proceso se vuelve engorroso para los especialistas y puede conllevar a la ocurrencia de errores humanos.

Los profesionales del Centro de Telemática de la Facultad 2 necesitan gestionar la información presente en las bases de datos relacionales de los servidores, pero la aplicación AD&MONPSQL en su versión 2.0 carece de las funcionalidades necesarias para administrar dichas bases de datos. Como consecuencia existen dificultades en la administración de la información y pérdida de tiempo al ejecutar los comandos necesarios para hacerlo manualmente.

A partir de las limitaciones de la anterior aplicación se define el siguiente **problema a resolver:** AD&MONPSQL v2.0 presenta limitaciones en la instalación del servicio PostgreSQL, en la garantía de la seguridad de la información de los servidores, así como en la gestión de las bases de datos relacionales, lo que dificulta la administración de los servidores en los que se pretende instalar PostgreSQL y garantizar la seguridad de sus datos.

Se puede diferir como **objeto de estudio:** El proceso de administración de los servidores PostgreSQL para la instalación remota del servicio PostgreSQL y gestión de las bases de datos relacionales y puntos de recuperación de forma centralizada. La presente investigación tiene como **objetivo general:** Desarrollar el sistema AD&MONPSQL v3.0 que garantice la instalación remota del servicio PostgreSQL y gestión de bases de datos relacionales y puntos de recuperación en la administración de los servidores instalados en los sistemas operativos Ubuntu, Debian y CentOS 7. Enmarcado en el **campo de acción:** La instalación remota del servicio PostgreSQL y gestión de las bases de datos relaciones y puntos de recuperación en los servidores de datos con los sistemas operativos Ubuntu, CentOS 7 y Debian.

Para accionar del diagnóstico al pronóstico se declaran las siguientes preguntas científicas:

- ¿Cuáles son los principales conceptos y definiciones asociados a la instalación remota del servicio PostgreSQL, copias de seguridad y la gestión de bases de datos relacionales para sentar las bases teóricas de la investigación?
- ¿Cuáles son las herramientas, tecnologías y la metodología de desarrollo de software a utilizar para una mejor familiarización con el entorno de trabajo?
- ¿Cuáles son las funcionalidades necesarias para que AD&MONPSQL gestione los puntos de recuperación, instale de forma remota el servicio PostgreSQL y gestione bases de datos relacionales en los servidores?
- ¿Qué métodos de prueba se deben aplicar para la validación de la propuesta de solución?

Teniendo como tareas de investigación:

- Determinación de los principales conceptos y definiciones asociados a la instalación remota del servicio PostgreSQL, copias de seguridad y la gestión de bases de datos relacionales en servidores PostgreSQL para sentar las bases teóricas de la investigación.
- Identificación de las herramientas, tecnologías y la metodología de desarrollo de software a utilizar para una mejor familiarización con el entorno de trabajo.
- Identificación de los requisitos que posee el sistema AD&MONPSQL v3.0 que permiten gestionar puntos de recuperación, instalar de forma remota el servicio PostgreSQL y gestionar bases de datos relacionales en los servidores.
- Implementación de las funcionalidades necesarias para que AD&MONPSQL gestione los puntos de recuperación, instale de forma remota el servicio PostgreSQL y gestione bases de datos relacionales en los servidores.

- Validación del funcionamiento del sistema por medio de pruebas utilizando los métodos de caja blanca y caja negra.

Métodos científicos:

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Se clasifican en teóricos y empíricos, los cuales están dialécticamente relacionados (León y Coello, 2012). Para el progreso de la presente investigación se utilizaron los siguientes métodos científicos de la investigación como guía para el desarrollo que condujo al logro del objetivo propuesto.

Métodos teóricos:

Analítico-Sintético: Posibilitó extraer e identificar conceptos, características y otros elementos de la bibliografía consultada para sentar las bases en el estudio del comportamiento y las características del sistema AD&MONPSQL con el fin de determinar la necesidad de actualización del mismo como propuesta de solución.

Modelación: Se utilizó para ofrecer información acerca de la estructura del diseño y las diferentes aristas y relaciones que se dan en la administración de PostgreSQL con el sistema AD&MONPSQL v2.0, así como, facilitar la comprensión y estudio del fenómeno real. Esto permite reflejar la estructura, las relaciones y características de la solución propuesta.

Histórico-lógico: Se utilizó con el objetivo de realizar un estudio sobre los antecedentes del tema a investigar, permitiendo conocer acerca de la existencia y características de sistemas para la administración de los servidores PostgreSQL hasta el estado actual.

Métodos Empíricos:

Observación: Se utilizó para adquirir conocimientos y recoger información en el diagnóstico inicial y final de la Gestión Puntos de Recuperación, Instalación remota del servicio PostgreSQL y Gestión Bases de Datos Relacionales. También se toma como punto de partida en el diseño de la investigación, ya que se observan todos los procedimientos que se realizan para administrar los servidores PostgreSQL por medio de AD&MONPSQL v2.0.

Entrevista: Se aplicó antes, durante y después de la investigación al cliente como fundamental fuente de recopilación de información, para entender qué problemas existen en la aplicación web AD&MONPSQL v2.0 y saber cómo solucionarlos.

El documento está estructurado por 3 capítulos, las conclusiones generales, las referencias bibliográficas, la bibliografía general utilizada y los anexos. La estructura de los capítulos se define a continuación:

Capítulo 1: Fundamentación teórica: Se definen los conceptos asociados y se analizan las principales herramientas informáticas que permiten administrar al SGBD PostgreSQL. Una vez realizado este análisis se describen brevemente las principales características de AD&MONPSQL v2.0, como sistema base de la aplicación a desarrollar. Por último, se fundamentan las tecnologías, lenguajes de programación, metodología y herramientas informática utilizadas en el desarrollo del sistema.

Capítulo 2: Análisis y diseño del sistema: Se presenta la propuesta de solución al problema planteado, quedando definidos los requisitos funcionales y no funcionales del sistema. Así como todos los artefactos que propone la metodología de desarrollo de software. También se detalla la arquitectura del sistema y elementos del diseño del sistema (patrones GRASP y GoF).

Capítulo 3: Implementación y pruebas del sistema: Se describen los estándares de codificación seguidos para la implementación de la propuesta de solución. Así como representación de los diagramas de componentes y de despliegue correspondientes con dicha propuesta. Para culminar se representan las pruebas realizadas al sistema, a partir de los indicadores a evaluar para sus resultados.

CAPÍTULO I. FUNDAMENTOS TEÓRICOS SOBRE LA ADMINISTRACIÓN DE LOS SERVIDORES POSTGRESQL

En el presente capítulo se abordan los conceptos asociados al mantenimiento necesario en AD&MONPSQL v2.0 para el desarrollo de su versión 3.0, y a la investigación. Se hace el análisis de los diferentes sistemas que se utilizan para la administración del SGBD PostgreSQL. También se caracterizan la metodología, las tecnologías, lenguajes y herramientas a utilizar para el desarrollo de la solución.

1.1. Proceso de administración de los servidores PostgreSQL

La administración de los servidores que tienen PostgreSQL como su sistema de gestión de base de datos, es un proceso en donde se planea, se organiza, en dirección hacia el control de los recursos para lograr un objetivo eficiente y la capacidad de obtener los mayores resultados con la mínima inversión, siendo las **Bases de datos** una colección de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción común y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones (Fuentes, 2013), dichas bases de datos necesitan un servidor que las contengan.

Un **servidor de base de datos** es un sistema bajo la arquitectura cliente/servidor que proporciona servicios de gestión, administración y protección de la información (datos) a través de conexiones de red (Iruela, 2016). Es necesario un servidor de base de datos para aquellas bases de datos con múltiples usuarios donde se puede acceder a ellas desde terminales o equipos con un programa llamado cliente, siendo los **Sistemas gestores de base de datos** una colección de programas informáticos para la gestión de bases de datos con el objetivo de evitar la manipulación directa por el usuario a una base de dato, así como establecer un marco estándar para que los datos sean manipulados y accedidos desde otros programas (Benítez y Arias, 2017) y **PostgreSQL** un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema.

1.2. Instalación remota del servicio PostgreSQL y puntos de recuperación

La instalación remota del servicio PostgreSQL es un proceso en que el archivo de instalación y todas sus dependencias son transferidas desde la máquina administradora a la máquina servidora vía **SSH** que es un protocolo para el inicio de sesión remoto seguro y otros servicios de red segura a través de una red. En otras palabras, SSH es un protocolo de transporte seguro y de bajo nivel. Proporciona cifrado sólido, autenticación criptográfica de host y protección de la integridad (Ylonen y Lonvick, 2006), para posteriormente instalarlos mediante comandos de instalación enviados remotamente desde la máquina administradora a la máquina servidora.

Las copias de seguridad son copias de la información de un sistema informático realizadas para que pueda ser recuperada en caso de fallo, borrado accidental o cualquiera otra contingencia (Cervigón y Alegre, 2011), existen varias formas de hacer copias de seguridad, una de ellas es la recuperación en un punto en el tiempo.

La recuperación en un punto en el tiempo permite restaurar una base de datos en un estado en el que se encontraba en cualquier momento. Este tipo de recuperación solo se aplica a las bases de datos que se ejecutan bajo el modelo de recuperación total o de registro masivo (Omelchenko, 2015).

PostgreSQL PITR, también llamada copia de seguridad incremental de la base de datos, copia de seguridad en línea o puede ser una copia de seguridad de archivo. En todo momento, PostgreSQL mantiene un registro de escritura anticipada (WAL) en el subdirectorio *pg_wal* del directorio de datos del clúster. El registro registra cada cambio realizado en los archivos de datos de la base de datos. Este registro existe principalmente para propósitos de seguridad: si el sistema falla, la base de datos se puede restaurar a la consistencia "reproduciendo" las entradas de registro realizadas desde el último punto de control. Sin embargo, la existencia del registro permite utilizar una tercera estrategia para hacer una copia de seguridad de las bases de datos: se puede combinar una copia de seguridad a nivel del sistema de archivos con la copia de seguridad de los archivos WAL. Si es necesaria la recuperación, se restaura la copia de seguridad del sistema de archivos y luego se reproduce desde los archivos WAL respaldados para recuperar el estado inicial del sistema.

1.3. Análisis de los sistemas utilizados para la administración del SGBD PostgreSQL

Se realizó el estudio de los sistemas homólogos existentes a nivel internacional y nacional para lograr una mejor comprensión de las características necesarias para la mejora del sistema. A continuación, se describe el análisis desarrollado.

1.2.1. Internacional

Webmin

Webmin representa una herramienta para la configuración mediante la web, para GNU/Linux y otros sistemas Unix. Permite el control de los servidores no solo de forma local, sino también de forma remota. Esta herramienta posee algunas características que enriquecen la investigación, estas son:

- Instalar PostgreSQL: Esta herramienta permite la instalación de PostgreSQL mediante un módulo de instalación de paquetes que permite la herramienta buscar los paquetes e instalarlos en las máquinas que se gestionan.
- Realizar copia de respaldo de los datos: Una vez instalado PostgreSQL, y el módulo para gesionarlo, el programa permite hacer copias de respaldo de la información en el servidor.
- Recuperar los datos: Una vez instalado PostgreSQL y el módulo para gestionarlo, el programa permite hacer la recuperación de los datos a partir de un fichero de recuperación de los datos.
- Crear bases de datos relacionales: Una vez instalados PostgreSQL y PostGIS se procede a seleccionar desde la página principal la opción “Crear un nuevo enlace de base de datos” arriba o debajo de la tabla de iconos de bases de datos existentes. Una vez seleccionado, ingresa un nombre único para la base de datos en el campo “Nombre de la base de datos” y se procede a seleccionar la opción “Crear” con la cual se agregará la base de datos y volverá a la página principal del sistema.

EMS SQL Manager for PostgreSQL

EMS SQL Manager para PostgreSQL es una herramienta de alto rendimiento para la administración y desarrollo de bases de datos PostgreSQL. Esta herramienta posee funcionalidades como:

- Crear bases de datos: Una vez instalados PostgreSQL se accede al botón correspondiente de la barra de herramientas “btnCreateDB”, utilizando el asistente que guiará todo el proceso de creación de una nueva base de datos.
- Eliminar bases de datos: Se selecciona la base de datos, después la opción eliminar el elemento del menú principal de la base de datos y se confirma la caída en la ventana de diálogo correspondiente.
- Hacer copia de respaldo de los datos: permite hacer copia de respaldo de los datos en el servidor.
- Recuperar los datos: permite hacer la recuperación de los datos a partir de un fichero de recuperación de los datos.

Barman

Barman (Backup and Recovery Manager) es una herramienta de administración de código abierto para la recuperación ante desastres de servidores PostgreSQL escritos en Python. Permite a las organizaciones realizar copias de seguridad remotas de varios servidores.

La característica que posee esta herramienta que enriquece la investigación es la recuperación de los datos mediante point-in-time-recovery (PITR).

1.2.2. Nacional

Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST)

Módulo de administración para el servicio PostgreSQL en la Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST). Dicho módulo fue desarrollado en el año 2014 en la UCI. Permite administrar, a través de una interfaz de usuario, los parámetros de configuración del servidor de base de datos. Esto es posible debido a que la aplicación se conecta vía SSH al servidor y accede a los ficheros de configuración y los modifica de acuerdo a las necesidades del gestor de configuración (Casasayas, 2014). Esta herramienta posee algunas características que enriquecen la investigación, estas son:

- Instalar y desinstalar el servicio PostgreSQL: Se realiza a partir de botones con el nombre de la acción a realizar y una vez seleccionada, se visualiza una imagen animada que indica el progreso de la instalación o desinstalación.
- Iniciar, reiniciar y detener el servicio PostgreSQL: Se realiza a partir de iconos, una vez seleccionada la opción deseada. El sistema ejecuta el comando “service postgresql start” en caso de iniciar el servicio, el comando “service postgresql stop” en caso de detener el servicio y en el caso de reiniciar se ejecuta el comando “service postgresql restart”.

AD&MONPSQL 2.0

Aplicación desarrollada inicialmente en el año 2017 en la UCI. En su versión 1.0 permitía administrar y monitorear por vía SSH a los diferentes servidores de PostgreSQL alojados en los sistemas operativos Ubuntu y Debian (Caldevilla y Méndez, 2018), después en su versión actual permite gestionar las Bases de Datos Geoespaciales y configurar los certificados SSL en la administración de los servidores PostgreSQL instalados en los sistemas operativos CentOS 7 y Windows Server 2012. La misma cuenta con una organización interna de 3 capas lógicas:

- Interfaz: En esta capa están implementadas todas las interfaces gráficas con las que interactúa el usuario. Estas interfaces se relacionan directamente con las funcionalidades que se encuentran implementadas en la capa de negocio.
- Negocio: En esta capa están incluidas todas las tareas y funcionalidades que realiza el sistema.

- Base de Datos: En esta capa se encuentran las bases de datos con las que trabaja el sistema.

Dentro de las características principales del sistema, las que se presentan a continuación, son las que enriquecen la presente investigación:

- Conectarse por SSH a una máquina servidora (Caldevilla y Méndez, 2018).
- Instalar PostgreSQL (Caldevilla y Méndez, 2018).
- Gestión de base de datos geoespacial (CentOS 7 y Windows Server 2012) (Fajardo; 2019).

Una vez analizados los sistemas de administración del SGBD PostgreSQL estudiados, se realizó una tabla resumen con algunas características importantes para el desarrollo del sistema, además se refleja en la misma los elementos positivos que aporta cada una de las soluciones y las nuevas funcionalidades que debe incorporar AD&MONPSQL v3.0 (ver Tabla 1).

Tabla 1: Análisis de los sistemas utilizados para la administración del SGBD PostgreSQL

Indicadores comparativos	**Webmin**	**EMS SQL Manager para PostgreSQL**	**Barman**	**HMAST**	**AD&MONPSQL v2.0**
Gestión de Bases de datos	X	X			X
Point in time Recovery			X		
Instalación remota del PostgreSQL		X			X
Administración del servicio PostgreSQL	X	X		X	X

Con apoyo de la tabla anterior (ver Tabla 1) se determina que es factible realizar la versión 3.0 del sistema AD&MONPSQL por las razones siguientes:

- Para la herramienta Webmin, es necesaria la instalación de esta en cada máquina donde se pretende configurar PostgreSQL, por no permitir la instalación remota del servicio.
- EMS SQL Manager for PostgreSQL es necesario pagar una licencia por un monto entre 140 hasta 476 dólares americanos mensuales para su utilización y está disponible solamente para el sistema operativo Windows.
- HMAST no puede ser reutilizada debido a que no se encuentra disponible el acceso al código fuente y no hace la gestión de las bases de datos existentes en los servidores.
- Barman solo hace el respaldo de datos por la técnica de PITR, no hace la gestión de los mismos.
- AD&MONPSQL v2.0, a pesar de la aplicación estar realizada para el centro, no permite hacer el respaldo de los datos ni instalar de forma remota el servicio PostgreSQL.
- La mayoría de las herramientas carecen de las funcionalidades para hacer el PITR y la instalación remota del servicio PostgreSQL.

Teniendo de las herramientas estudiadas los siguientes aspectos positivos:

Siendo una interfaz basada en web para la gestión de sistemas UNIX y proporcionando una interfaz web sencilla pero potente, webmin aportó la idea de hacer la gestión de las bases de datos de forma rápida y sencilla y con una interfaz comprensible al usuario.

Siendo una herramienta de administración de código abierto para la recuperación ante desastres de servidores PostgreSQL, Barman aportó la idea de creación de respaldos de información a través del método de PITR que permite restaurar una base de datos en un estado en el que se encontraba en cualquier momento.

AD&MONPSQL v2.0 copia el fichero de instalación a la máquina servidora con sistema operativo Windows server para posteriormente hacer la instalación del mismo, esa práctica aportó de forma positiva en el desarrollo de la versión 3.0 del sistema AD&MONPSQL.

1.4. Metodología de desarrollo de software

Una metodología de desarrollo de software se refiere al marco que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. Una amplia variedad de dichos marcos, han evolucionado a lo largo de los años con sus propias fortalezas y debilidades reconocidas. Cada una de las metodologías disponibles se adapta mejor a tipos específicos de proyectos, en función de diversas consideraciones técnicas, organizativas, de proyecto y de equipo (Pressman, 2010).

Dentro de las metodologías más usadas, tenemos el Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos.

Metodología de desarrollo para la Actividad productiva de la UCI (Variación de AUP para la UCI).

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (Sánchez, 2015).

Con el objetivo de guiar el desarrollo de la propuesta de solución se utilizó la metodología AUP para la UCI, la cual fue seleccionada por las siguientes características:

- Para la realización del proyecto es necesaria la utilización de una metodología ágil ya que el equipo de trabajo es pequeño y de vasta experiencia en el tema a desarrollar.
- La criticidad del proyecto es baja.
- Define el ciclo de vida determinado para la actividad productiva de la UCI (Inicio, Ejecución, Cierre) y a su vez coincide con la duración del proyecto.
- Consta de una amplia documentación.
- Sus artefactos presentan una descripción del comportamiento del sistema fácil de entender.

La metodología AUP-UCI especifica las siguientes fases que guiarán el proceso de desarrollo:

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

La misma plantea cuatro escenarios para la obtención de los requisitos del software, dentro de los escenarios plantea la misma se seleccionó el escenario cuatro, dado a que:

- El negocio se encuentra bien definido.
- El cliente está siempre acompañando al equipo de desarrollo.
- El proyecto no es extenso.
- Para lograr la convergencia entre la versión a desarrollar y la anterior del sistema (AD&MONPSQL v2.0).

El escenario 4 de la variación AUP para la UCI plantea que proyectos que no modelan negocio solo pueden modelar el sistema con Historias de usuarios (HU), que se usaron para la especificación de los requisitos funcionales. Las mismas se caracterizan por ser tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento las HU pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas.

1.5. Herramientas, lenguajes y tecnologías

Las tecnologías utilizadas fueron las seleccionadas por los desarrolladores de la v1.0 y v2.0 de AD&MONPSQL con el objetivo de garantizar la compatibilidad entre ambas versiones y también para dar cumplimiento a las funcionalidades del sistema.

Lenguaje de programación del lado del servidor:

Python 3.6: Es un lenguaje de programación interpretado, orientado a objetos, multiplataforma, flexible y su implementación está bajo una licencia de código abierto PSFL [1]. Se utilizó para dar cumplimiento a las funcionalidades del sistema.

Lenguaje de programación del lado del cliente:

HTML 5: Lenguaje de Marcado de Hipertexto (HTML5, por sus siglas en inglés) es un lenguaje extensible, que se le pueden añadir características, etiquetas y funciones adicionales para el diseño de páginas web, generando un producto vistoso, rápido y sencillo. Además, provee básicamente tres características: estructura, estilo y funcionalidad (Gauchat, 2012). Se utilizó para realizar interfaces amigables y mostrar información al usuario a través del navegador web.

JavaScript: Es un lenguaje interpretado. Principalmente es utilizado para crear páginas web dinámicas. Basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet (Pérez, 2012). Se utilizó en la validación de los formularios llenados por el usuario en la realización del sistema.

CCS3: Define el diseño de un sitio web en un documento externo y eso mismo permite que modificando ese documento (la hoja CSS) se pueda cambiar el diseño entero de un sitio web. El mismo sitio web puede variar totalmente de diseño cambiando solo la CSS, sin tocar para nada los documentos HTML, JSP[2] o ASP[3] que lo componen (Gauchat, 2012). Se utilizó para estandarizar el diseño del sistema.

[1] *PSFL: Python Software Foundation License (Licencia de fundación de software).*

[2] *Jsp (siglas en ingles de Java Server Pages) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML, entre otros tipos de documentos.*

[3] *Asp (siglas en ingles de Active Server Pages) es la tecnología desarrollada por Microsoft para la creación de páginas dinámicas del servidor.*

Lenguaje de modelado:

Lenguaje Unificado de Modelado (UML[4]): Es un lenguaje para visualizar, especificar, construir y documentar sistemas de software. Se utilizó en la realización del modelo de diseño de la aplicación. El UML permitió diseñar la estructura estática y el comportamiento dinámico de los objetos del sistema mediante los diagramas estructurales (diagrama de clases, diagrama de comportamiento y diagrama de despliegue) y de comportamiento (diagrama de secuencia) generados en la fase de análisis de la presente investigación.

Marcos de Trabajo:

Django 1.11.6: Es un marco de trabajo para el desarrollo web escrito en Python, que permite construir y mantener aplicaciones web de alta calidad. Django, provee un alto nivel de abstracción para patrones comunes en el desarrollo web, incrementando la calidad de las soluciones, aumentando la productividad y disminuyendo los errores en el código (Holovaty y Kaplan-Moss, 2009), dichas características fueron utilizadas para realizar el manejo del lenguaje de programación Python apoyándose de las características antes mencionadas.

jQuery 1.9.1: Es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol Document Object Model (DOM), manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX[5] a páginas web (Sánchez y Yusbel, 2017), dichas características fueron utilizadas para facilitar el trabajo con el lenguaje interpretado JavaScript.

Bootstrap 3.3.1: Es un marco de trabajo que facilita el desarrollo de la interfaz web y está basado en los estándares de HTML, CSS y JavaScript. Llevan incorporado varias plantillas prediseñadas, formulario, botones, menús y otros componentes que facilitan y agilizan el desarrollo de las aplicaciones web (Maldonado y Mauricio, 2017), dichas características permitieron desarrollar y adaptar la interfaz del sistema.

4 *UML (Unified Modeling Language).*

[5] *AJAX (acrónimo de Asynchronous JavaScript and XML) es una técnica que permite la comunicación asíncrona entre un servidor y un navegador en formato XML mediante programas escritos en JavaScript.*

Herramienta CASE:

Visual Paradigm 8.0: Visual Paradigm para UML (VP-UML) es una poderosa herramienta CASE[6] multiplataforma (Windows/Linux/Mac OS X) que soporta el ciclo de vida completo del desarrollo de software. Está diseñado para un amplio rango de usuarios, incluyendo ingenieros de software, analistas de sistema, analistas de negocio, arquitectos de sistema y quienes estén interesados en la construcción de sistemas de software confiables mediante el uso de la Orientación a Objetos. Esta herramienta permitió el modelado del sistema mediante el UML.

IDE de desarrollo:

PyCharm 2018.1.2: Proporciona la finalización inteligente de códigos, inspecciones de códigos, resaltado de errores sobre la marcha, soluciones rápidas y capacidades de navegación avanzadas. La colección de herramientas de PyCharm incluye un depurador y corrector de prueba integrado, una terminal incorporada, integración con un Sistema Control de Versiones (SCV) principal y herramientas de base de datos incorporadas y capacidades de desarrollo remoto con intérpretes remotos (Jetbrains, 2017). Esta herramienta permitió el desarrollo del sistema utilizando los marcos de trabajo y lenguajes de programación descritos anteriormente.

1.6. Conclusiones del capítulo

Al realizar el estudio de los conceptos asociados al dominio se logró un mejor entendimiento del problema a resolver.

Luego de realizado el análisis de múltiples herramientas para la administración del sistema gestor de base de datos PostgreSQL, se decidió la elaboración de AD&MONPSQL en su versión 3.0 teniendo en cuenta que ninguna cumple del todo con las funcionalidades (Gestión de las bases de datos relacionales, respaldo de la información por PITR, Instalación del servicio PostgreSQL en sus diferentes versiones) que se requieren para la propuesta de solución. Además, se identificaron elementos positivos que estas herramientas incorporan y que permitirán enriquecer los requisitos a implementar.

6 CASE: Computer Aided Software Engineering en español (Ingeniería de Software Asistida por Computadoras).

Se utilizó como guía para el proceso de desarrollo de software la metodología: AUP para la UCI que es la adaptación de AUP que se propone para la actividad productiva de la UCI.

Por último, con el fin de mantener la compatibilidad con el sistema AD&MONPSQL v2.0 se definieron las herramientas informáticas a utilizar en la implementación del sistema con el fin de desarrollar un sistema multiplataforma, simple, libre y de código abierto.

CAPÍTULO II. ANÁLISIS Y DISEÑO DEL SISTEMA AD&MONPSQL V3.0

En el presente capítulo se diseñan los artefactos que propone la metodología de desarrollo de software seleccionada en el capítulo anterior. Para la realización de los artefactos fue necesaria la captura de los requisitos del software y a partir de estos se describen los diagramas de clases de diseño y los diagramas de secuencia. También se especifica la arquitectura del sistema, los patrones de diseño que fueron utilizados y el modelo de diseño de la aplicación.

2.1. Características de la propuesta de solución

A continuación, se presentan las funcionalidades que contiene la propuesta de solución con la que se garantiza la instalación del servicio PostgreSQL sin hacer uso de los repositorios, así como gestión de las bases de datos relacionales y el respaldo de la información existente en los servidores del Centro TLM. Para ello el usuario (administrador) accede al sistema. Una vez dentro, el sistema permite realizar varias funciones mediante conexión segura SSH a los servidores:

- Copiar en el servidor los ficheros necesarios para la instalación del servicio PostgreSQL.
- Instalar el Servicio PostgreSQL sin depender los repositorios.
- Gestionar (Listar, Crear y eliminar) un punto de recuperación.
- Restaurar la información a partir de un punto de recuperación.
- Gestionar (crear, renombrar, eliminar) las bases de datos relacionales.

A continuación de representa la propuesta de solución mediante el modelo conceptual que es una representación que se realiza para entender el proyecto donde se está trabajando. Sirve para identificar las relaciones que contienen todas las entidades de un sistema y contiene conceptos que estarán asociados tanto a su definición natural como al papel que juegan desde el punto de vista informático (Infante, 2013).

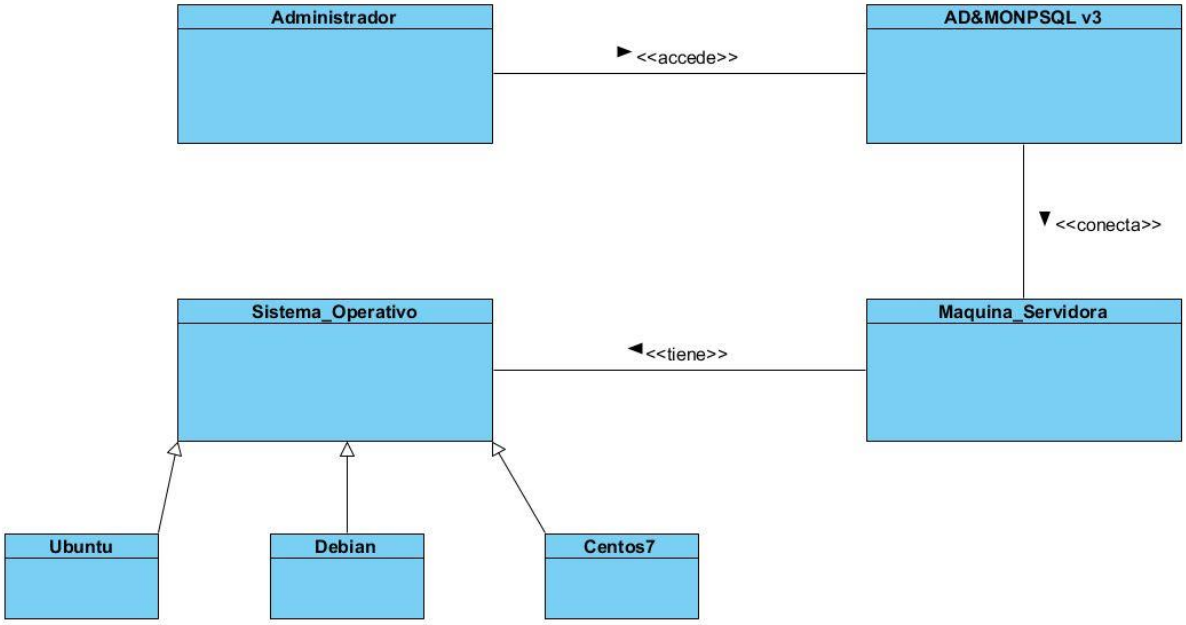


Figura 1: Propuesta de Solución

Se crean cuatro entidades (Administrador, AD&MONPSQL V3, Máquina_servidora, Sistema_Operativo), todas están relacionadas entre sí orientadas a describir la semántica y observaciones sobre la información del dominio que en este caso es la instalación del servicio PostgreSQL y la gestión de las BDR y PITR. El Administrador accede al sistema AD&MONPSQL V3 que es responsable por la administración de los servidores con SGBD PostgreSQL, este permite conectarse a la Máquina_Servidora que tiene Sistema_Operativo (CentOS 7, Ubuntu, Debian), permitiendo así al administrador instalar el servicio PostgreSQL y gestionar las bases de datos relacionales y puntos de recuperación, mediante comandos y ficheros de instalación enviados desde el sistema hacia la máquina servidora.

2.2. Requisitos del sistema

Los requisitos son capacidades y condiciones con las cuales debe ser conforme el sistema, y más ampliamente, el proyecto (Larman, 2003) La intención principal del uso de los requisitos es guiar el desarrollo hacia el sistema correcto. Estos se clasifican en funcionales y no funcionales. A continuación, se describen las técnicas utilizadas en la captura de los requisitos, así como los que fueron definidos para llevar a cabo el desarrollo de la propuesta de solución

2.2.1. Técnicas de obtención de requisitos del software

La obtención de requisitos es el proceso mediante el cual los interesados en un sistema de software obtienen los requisitos. Este proceso necesita el empleo de técnicas que permitan establecer una buena comunicación con los interesados en el producto y así lograr la satisfacción del cliente, con el objetivo de obtener los requisitos se utilizó como técnica de obtención de requisitos la entrevista porque es de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades.

Entrevistas

Las entrevistas formales o informales con participantes del sistema son una parte de la mayoría de los procesos de ingeniería de requerimientos. En estas entrevistas, el equipo de ingeniería de requerimientos formula preguntas a los participantes sobre el sistema que actualmente usan y el sistema que se va a desarrollar. Los requerimientos se derivan de las respuestas a dichas preguntas (Sommerville, 2011).

Entrevista no estructurada: Se emplea para reunir información proveniente del administrador. Durante la misma, la analista conversa con el encuestado; el cual responde en una serie de preguntas relacionadas con varios aspectos del sistema. (**Anexo 3** Guía para la realización de la entrevista).

2.2.2. Requisitos funcionales

Los requisitos funcionales son aquellos directamente relacionados con las funciones o relaciones que el sistema debe proporcionar (Cardozzo, 2016). Teniendo en cuenta las necesidades del centro de Telemática fueron definidos 30 requisitos funcionales agrupados en tres paquetes de funcionalidades, los mismos se muestran a continuación:

Paquete de funcionalidades número 1: Gestión bases de datos relacionales.

RF1 Listar bases de datos en CentOS 7.

RF2 Listar bases de datos en Ubuntu.

RF3 Listar bases de datos en Debian.

RF4 Crear base de datos relacional en CentOS 7.

RF5 Crear base de datos relacional en Ubuntu.

RF6 Crear base de datos relacional en Debian.

RF7 Renombrar base de datos relacional en CentOS 7.

RF8 Renombrar base de datos relacional en Ubuntu.

RF9 Renombrar base de datos relacional en Debian.

RF10 Eliminar base de datos relacional en CentOS 7.

RF11 Eliminar base de datos relacional en Ubuntu.

RF12 Eliminar base de datos relacional en Debian.

Paquete de funcionalidades número 2: Gestión de puntos de recuperación.

RF13 Listar puntos de recuperación en CentOS 7.

RF14 Listar puntos de recuperación en Ubuntu.

RF15 Listar puntos de recuperación en Debian.

RF16 Crear puntos de recuperación en CentOS 7.

RF17 Crear puntos de recuperación en Ubuntu.

RF18 Crear puntos de recuperación en Debian.

RF19 Restaurar punto de recuperación en CentOS 7.

RF20 Restaurar punto de recuperación en Ubuntu.

RF21 Restaurar punto de recuperación en Debian.

RF22 Eliminar punto de recuperación en CentOS 7.

RF23 Eliminar punto de recuperación en Ubuntu.

RF24 Eliminar punto de recuperación en Debian.

Paquete de funcionalidades número 3: Instalación del servicio PostgreSQL sin uso de repositorios.

RF25 Instalar de forma remota de PostgreSQL en CentOS 7.

RF26 Instalar de forma remota de PostgreSQL en Ubuntu.

RF27 Instalar de forma remota de PostgreSQL en Debian.

RF28 Instalar diferentes versiones de PostgreSQL en CentOS 7.

RF29 Instalar de diferentes versiones de PostgreSQL en Ubuntu.

RF30 Instalar de diferentes versiones de PostgreSQL en Debian.

2.2.3. Requisitos no funcionales

Los requisitos no funcionales son restricciones impuestas al funcionamiento del sistema, tales como las limitaciones de tiempo, presupuesto, proceso de desarrollo, las políticas de la organización y normas que deben adoptarse (Cardozzo, 2016). Se puede decir que los requisitos no funcionales se refieren a cualidades que imponen restricciones en el diseño y la implementación del sistema.

Eficiencia:

- El sistema responde las peticiones del usuario en 15 segundos (dependiendo del tiempo que la máquina servidora lleva a ejecutar los comandos).

Mantenibilidad:

- Se utilizan los estándares de programación (CapWords y lower_case_with_underscores) para estandarizar la definición de clases y métodos del sistema.

Interoperabilidad:

- La máquina en que se ejecute el sistema debe constar con un navegador web.
- El sistema tiene que funcionar en los sistemas operativos CentOS 7, Ubuntu server 18.4 y Debian 9.

Usabilidad:

- El sistema debe visualizar todos los mensajes en idioma español.
- Las funcionalidades principales del sistema estarán orientadas a íconos para un mayor reconocimiento por parte del usuario.

- Las máquinas a la que se desee acceder desde el sistema tienen que tener configurado el protocolo SSH.
- Se debe acceder a las máquinas servidoras utilizando el usuario root (usuario con permisos de administración).

Hardware:

- Se requiere para los servidores las siguientes condiciones mínimas:

  - Procesador: 4Ghz.

  - Memoria RAM: 8GB.

  - Disco Duro: 500MB.

- Se requiere para las estaciones de trabajo cliente las siguientes condiciones mínimas:

  - Poseer tarjeta de red.

  - Procesador: 2Ghz.

  - Memoria RAM: 2 GB.

Seguridad:

- Garantizar el acceso a la aplicación solo por los usuarios con permisos específicos.
- La información sensible (ejemplo: la contraseña de acceso al servidor) debe ser almacenada de forma no explícita.

Confiabilidad:

- Garantizar el tratamiento adecuado de las excepciones.

2.3. Validación de requisitos

Para la validación de los requisitos se empleará la técnica de Acta de conformidad del cliente. Esta consiste en recoger los requisitos funcionales detectados. Luego de ser revisada y firmada por el cliente se da fin a este ciclo (**Anexo 1** Acta de aceptación de los requisitos por parte del cliente).

2.4. Especificación de los requisitos funcionales

Cada HU es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Para efectos de planificación, las HU pueden ser de una a tres semanas de tiempo de programación A continuación, se muestran algunas de las HU correspondientes al paquete “Gestionar_Puntos_de_Recuperación”). El resto de las HU se pueden visualizar en el **Anexo 2** Historias de usuario.

Tabla 2: HU Nº 13: Listar puntos de recuperación en CentOS 7

Número:13	**Nombre del requisito:** Listar puntos de recuperación en CentOS 7
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1
Prioridad: Alta	**Puntos estimados:** 0.4
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 día
Descripción: Funcionalidad que permite al usuario listar los puntos de recuperación existentes en el servidor con sistema operativo CentOS 7 al que se está conectado. Se debe mostrar un listado de los puntos de recuperación existentes en el servidor con la opción de adicionar un punto de recuperación visible para el usuario, mostrar también para cada punto de recuperación la opción de recuperar y eliminar. En caso de no existir ningún punto de recuperación se muestra un listado vacío.	
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL en una versión mayor que la 9.4.	
Prototipo de interfaz:	

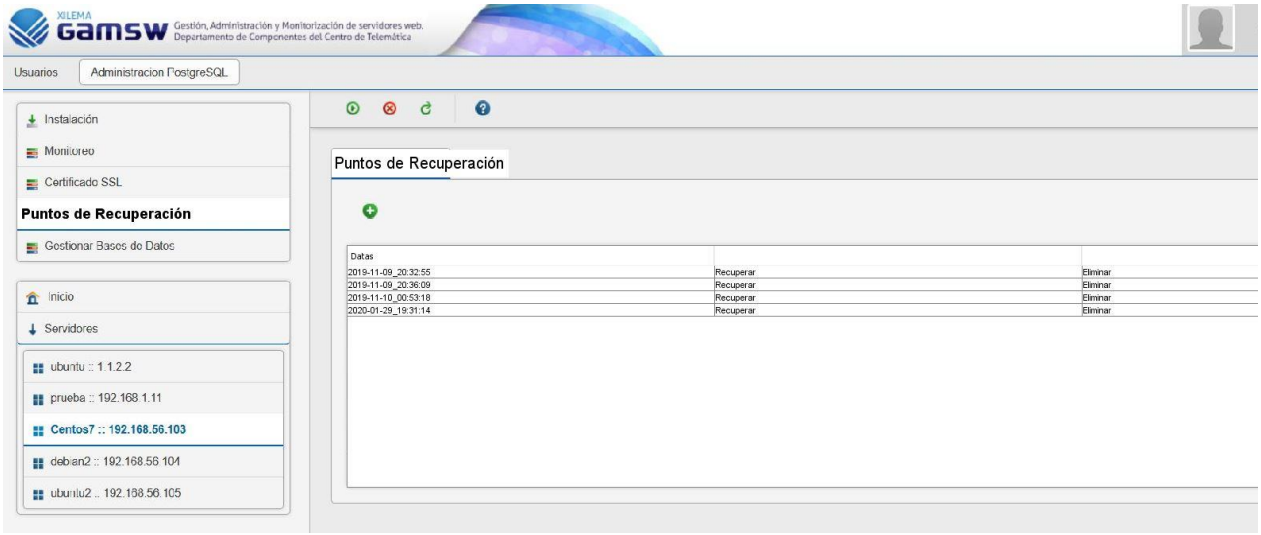


Tabla 3: HU Nº 20: Restaurar punto de recuperación en Ubuntu

Número:20	**Nombre del requisito:** Restaurar punto de recuperación en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.8
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 días

Descripción: Funcionalidad que permite restaurar las configuraciones y datos del servidor Ubuntu al que se está gestionando, al dar pulsar el icono de restauro se debe abrir una caja de confirmación del restauro de la información, se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Recuperación realizada con éxito!” o “Error en el servidor”).

Botones:

- Recuperar: Opción prosigue con la operación, recuperando todos los datos y configuraciones.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

Prototipo de interfaz:

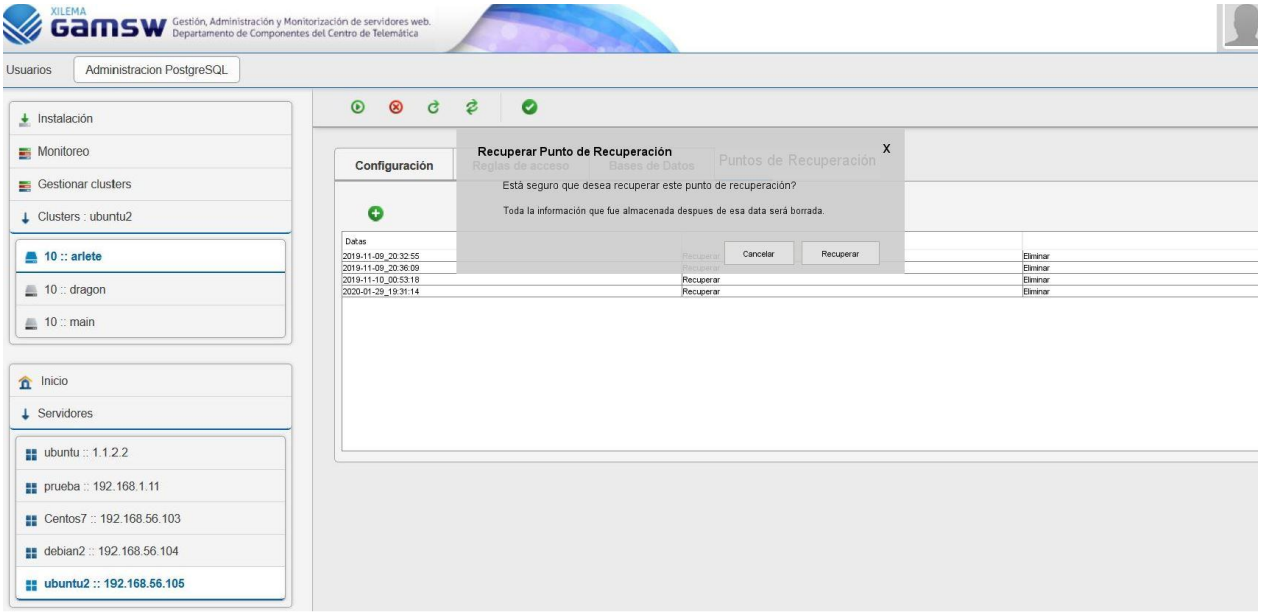


Tabla 4: HU Nº 21: Restaurar punto de recuperación en Debian

Número:21	**Nombre del requisito:** Restaurar punto de recuperación en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.8
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 días
Descripción: Funcionalidad que permite el usuario restaurar las configuraciones y datos de PostgreSQL en el servidor Debian al que se está gestionando, al dar pulsar el icono de restauro se debe abrir una caja de confirmación del restauro de los datos donde se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Recuperación realizada con éxito!” o “Error en el servidor”). **Botones:**	

- Recuperar: Opción prosigue con la operación, recuperando todos los datos y configuraciones.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

Prototipo de interfaz:

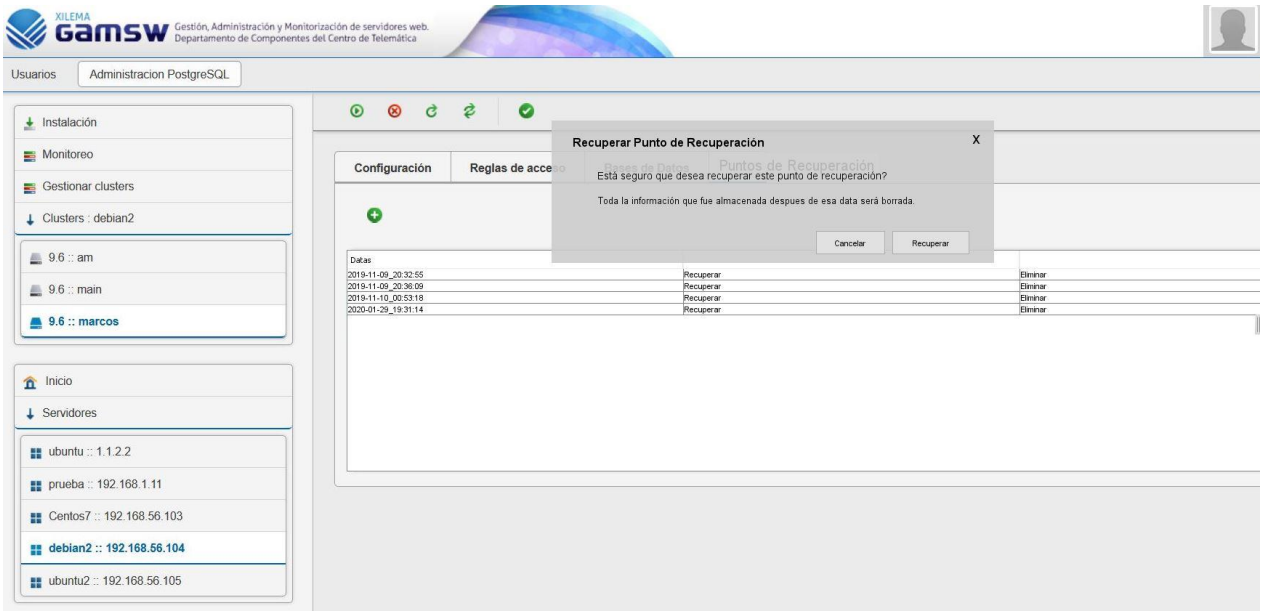


Tabla 5: HU Nº 24: Eliminar punto de recuperación en Debian

Número:24	**Nombre del requisito:** Eliminar punto de recuperación en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 días

Descripción: Funcionalidad que permite eliminar un punto de recuperación en el servidor Debian al que se está gestionando, al pulsar el icono de eliminación, se debe mostrar una caja de confirmación de la operación, en esa caja se debe mostrar la opción de eliminar o cancelar la

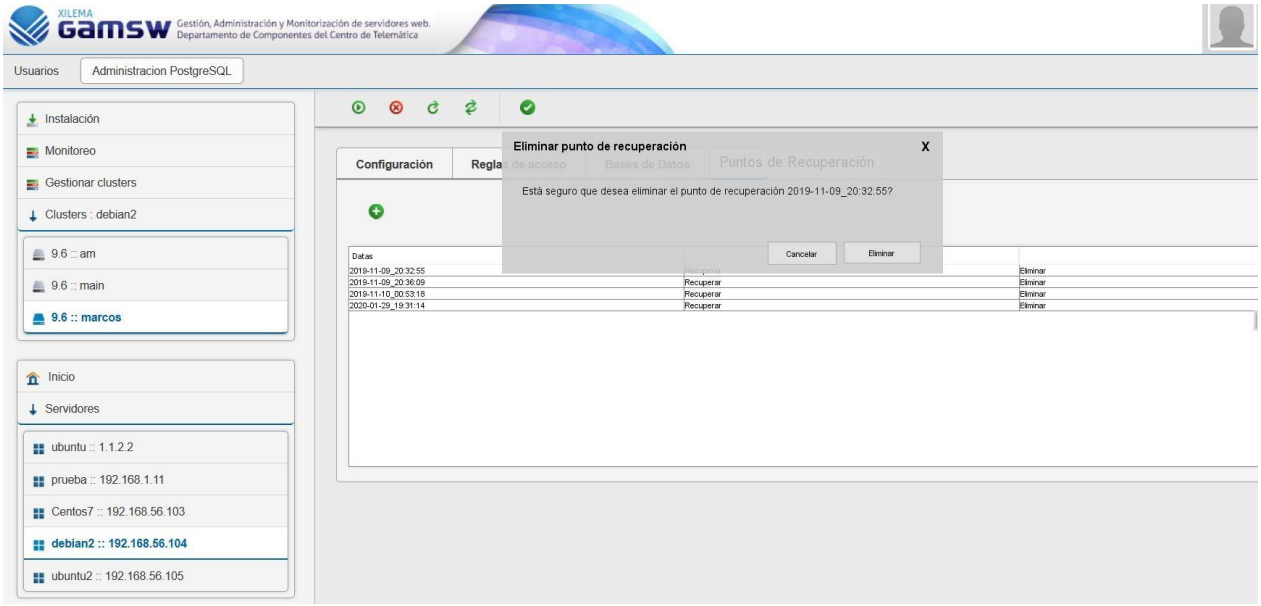
operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación borrado con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina el punto de recuperación seleccionado.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

Prototipo de interfaz:



2.5. Especificación de los requisitos no funcionales

La propuesta de solución cuenta con un total de 15 requisitos no funcionales agrupados en 6 atributos de calidad mostrados a continuación.

Tabla 6: Especificación RNF Eficiencia

Atributo de Calidad	*Eficiencia*
Sub-atributos/Sub-características	*Cumplimiento de eficiencia:* *El sistema debe responde las peticiones del usuario en 15 segundos (dependiendo del tiempo que la máquina servidora lleva a ejecutar los comandos).*
Objetivo	*Disminuir el tiempo de espera de las peticiones del usuario.*
Origen	*Administrador*
Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
1.A	
Al realizar una operación en el sistema (Ejemplo: Adicionar base de datos relacional) el mismo debe notificar al usuario que se está realizando la acción.	*Se muestra un mensaje de información o notificación (Ejemplo: La base de datos relacional fue creada con éxito) al realizar cualquier acción en el sistema.*
Medida de respuesta	
El sistema responde las peticiones del usuario en un tiempo entre 10 y 15 segundos (dependiendo del tiempo que la máquina servidora lleva a ejecutar los comandos).	

Tabla 7: Especificación RNF Mantenibilidad

Atributo de Calidad	*Mantenibilidad*
Sub-atributos/Sub-características	*Cumplimiento de mantenibilidad:* *Se deben utilizar estándares de programación*
Objetivo	*Estandarizar la programación del sistema.*
Origen	*Programador*
Artefacto	*Código*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
Nº 2. A	
N/A	*N/A*
Medida de respuesta	
Velar por los estándares de programación seguidos	

Tabla 8: Especificación RNF Interoperabilidad

Atributo de Calidad	*Interoperabilidad*
Sub-atributos/Sub-características	*Cumplimiento de interoperabilidad:* *El sistema tiene que funcionar en los sistemas operativos CentOS 7, Ubuntu server 18.4 y Debian 9.*
Objetivo	*Lograr la ejecución del sistema en los sistemas operativos CentOS 7, Ubuntu server 18.4 y Debian 9.*
Origen	*Administrador*
Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
Nº 3. A	
Ejecutar una máquina servidora (CentOS 7, Ubuntu server 18.4 y Debian 9).	*Se redirecciona al usuario a una página en la cual se muestran las opciones que se pueden realizar en la máquina servidora ejecutada.*
Medida de respuesta	
El sistema funciona en los sistemas operativos CentOS 7, Ubuntu server 18.4 y Debian 9.	

Tabla 9: Especificación RNF Usabilidad

Atributo de Calidad	*Usabilidad*
Sub-atributos/Sub-características	*Cumplimiento de Usabilidad:* *Visualizar todos los mensajes en idioma español.* *Las funcionalidades principales del sistema deben estar orientadas a íconos.* *Las máquinas a la que se desee acceder desde el sistema tienen que tener configurado el protocolo SSH.* *La máquina en que se ejecute el sistema debe constar de un navegador web.* *Se debe acceder a las máquinas servidoras utilizando el usuario root de las máquinas.*
Objetivo	*Lograr que el sistema sea entendible para las personas de habla hispana.*
Origen	*Administrador*
Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
Nº 4. A	
Seleccionar la opción instalar PostgreSQL.	*Se muestra un mensaje de notificación (en idioma español) en cualquier caso (correcto o incorrecto).*
Nº 4. B	

Seleccionar el icono del cesto de basura cuando se desee eliminar una base de datos relacional o un punto de recuperación.	*Se muestra un mensaje de notificación (en idioma español) en cualquier caso (correcto o incorrecto).*
Medida de respuesta	
Utilización de las características señaladas para la ejecución del sistema.	

Tabla 10: Especificación RNF Hardware.1

Atributo de Calidad	*Hardware*
Sub-atributos/Sub-características	*Utilización de recursos*
Objetivo	*Ejecutar el sistema en los servidores con las siguientes condiciones mínimas:* *- Procesador: 4Ghz.* *- Memoria RAM: 8GB.* *- Disco Duro: 1GB.*
Origen	*Externo al sistema*
Artefacto	*Servidor de aplicación*
Entorno	*Operación normal*
Estímulo	*Respuesta: Flujo de eventos*
NA	*NA*

Tabla 11: Especificación RNF Hardware.2

Atributo de Calidad	*Hardware*
Sub-atributos/Sub-características	*Utilización de recursos*
Objetivo	*Ejecutar las funcionalidades del sistema en los servidores de base de datos con las siguientes condiciones mínimas:* *- Poseer tarjeta de red.* *- Procesador: 2Ghz.* *- Memoria RAM: 2 GB.*
Origen	*Externo al sistema*
Artefacto	*Servidor de aplicación*
Entorno	*Operación normal*
Estímulo	*Respuesta: Flujo de eventos*
NA	*NA*

Tabla 12: Especificación RNF Seguridad.1

Atributo de Calidad	*Seguridad*
Sub-atributos/Sub-características	*Acceso restringido:* *Garantizar el acceso a la aplicación solo por los usuarios con permisos específicos*
Objetivo	*Garantizar la seguridad del sistema.*
Origen	*Administrador*

Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
Nº 5. A	
Autenticarse en el sistema.	*Solo permitir la entrada al sistema a los usuarios administradores.*
Medida de respuesta	
Denegar el acceso a los usuarios sin permiso de entrada al sistema.	

Tabla 13: Especificación RNF Seguridad.2

Atributo de Calidad	*Seguridad*
Sub-atributos/Sub-características	*Resistente a manipulación:* *La información sensible debe ser almacenada de forma no explicita.*
Objetivo	*Garantizar la seguridad del sistema.*
Origen	*Programador*
Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
N/A	*N/A*
Medida de respuesta	

Resistencia del sistema a la manipulación.

Tabla 14: Especificación RNF Confiabilidad

Atributo de Calidad	*Confiabilidad*
Sub-atributos/Sub-características	*Garantizar el tratamiento adecuado de las excepciones.*
Objetivo	*Garantizar el adecuado funcionamiento del sistema.*
Origen	*Programador*
Artefacto	*El sistema*
Entorno	*El sistema está funcionando correctamente.*
Estímulo	***Respuesta: Flujo de eventos***
Nº 6. A	
Garantizar que para cada acción halla una respuesta del sistema.	*El sistema funciona correctamente.*
Medida de respuesta	
Verificar respuesta del sistema.	

2.6. Arquitectura del sistema

El estándar 1471-2000 del Instituto de Ingeniería Eléctrica y Electrónica (IEEE, según sus siglas en inglés) describe que la arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. Un mismo sistema puede utilizar varias arquitecturas, en este caso por las características del sistema es necesario utilizar las siguientes:

Modelo Cliente-Servidor

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas. En este modelo intervienen únicamente dos entidades: El Cliente y El Servidor (Marini, 2012).

- El papel del cliente lo desempeña la aplicación final del usuario, que implementará todas las funciones correspondientes a la lógica de presentación, más algunas de las funciones relacionadas con la lógica del negocio, como pueden ser determinadas validaciones de datos y condiciones de recuperación.
- El papel del servidor lo desempeña el propio SGBD, el cual se ocupará de todas las funciones correspondientes a la lógica de datos, más las restantes funciones correspondientes a la lógica del negocio.

Modelo Vista Plantilla (MVT)

El marco de trabajo Django, utilizado para la realización de la solución, implementa el patrón arquitectónico Modelo Vista Plantilla (MVT). La misma es una implementación especial del Modelo-Vista-Controlador. A continuación, se muestra una imagen donde se puede visualizar la variación entre los patrones (ver figura 2).

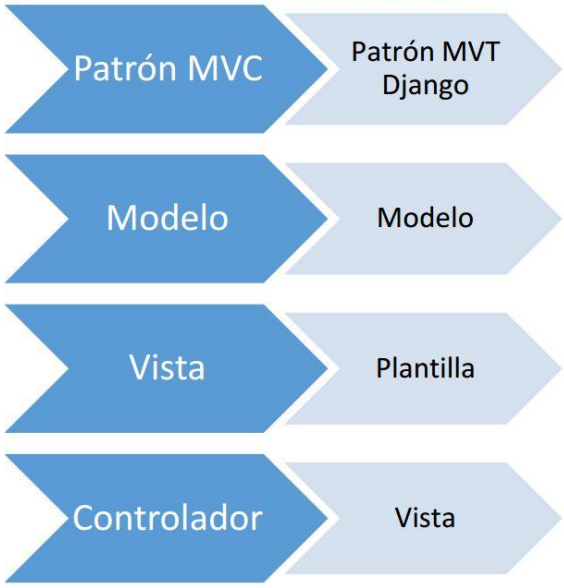


Figura 2: Variación del MVC a MVT

Esta arquitectura permite una separación entre interfaz, lógica del negocio y de presentación, lo que permite una gran reutilización de componentes, así como facilidad para desarrollar prototipos y realizar pruebas unitarias de los componentes. La siguiente imagen muestra con más detalles una representación del modelo MVT adaptado al sistema a desarrollar (ver figura 3).

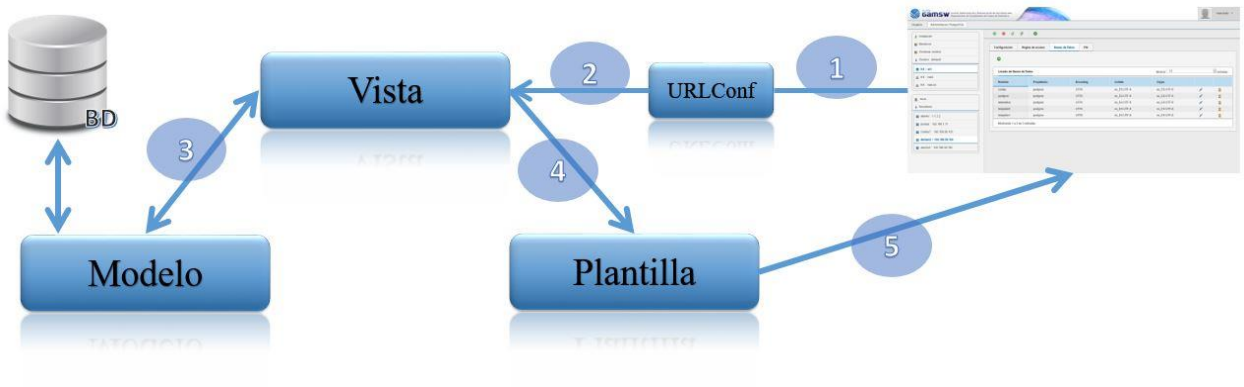


Figura 3: Funcionamiento de la arquitectura MVT (George, 2019)

El cliente desde su navegador realiza una solicitud de un servicio web de la aplicación, por lo que se pone en contacto con la capa de la vista.

La capa de la vista necesita tener contacto directo con la del modelo ya que es de ella de donde cogerá los datos que solicita el usuario. Por lo que entra en acción la capa del modelo. Ésta

última capa, gracias a las consultas a la base de datos devuelve los objetos a la capa de la vista, que ella ya se encargará de filtrar y devolver explícitamente los que el usuario ha solicitado en la petición web.

El siguiente paso es que la capa de la vista envíe a la capa de la plantilla los datos requeridos por el usuario, tras previo procesamiento de los mismos. En este paso, ha actuado indirectamente el modelo, ya que es quien proporcionó los datos a la vista.

En último lugar, la capa de la plantilla es la encargada de cómo se deben mostrar los datos al usuario. Por lo que, con los datos recibidos de la vista, los incluye en su respectiva plantilla para que los navegadores puedan procesarlo y el usuario visualice correctamente la información

2.7. Diagrama de componentes

El diagrama de componentes representa cómo un sistema de software es dividido en elementos y muestra las dependencias entre estos elementos. A continuación, se muestra el diagrama de componentes correspondiente al sistema AD&MONPSQL v3.0.

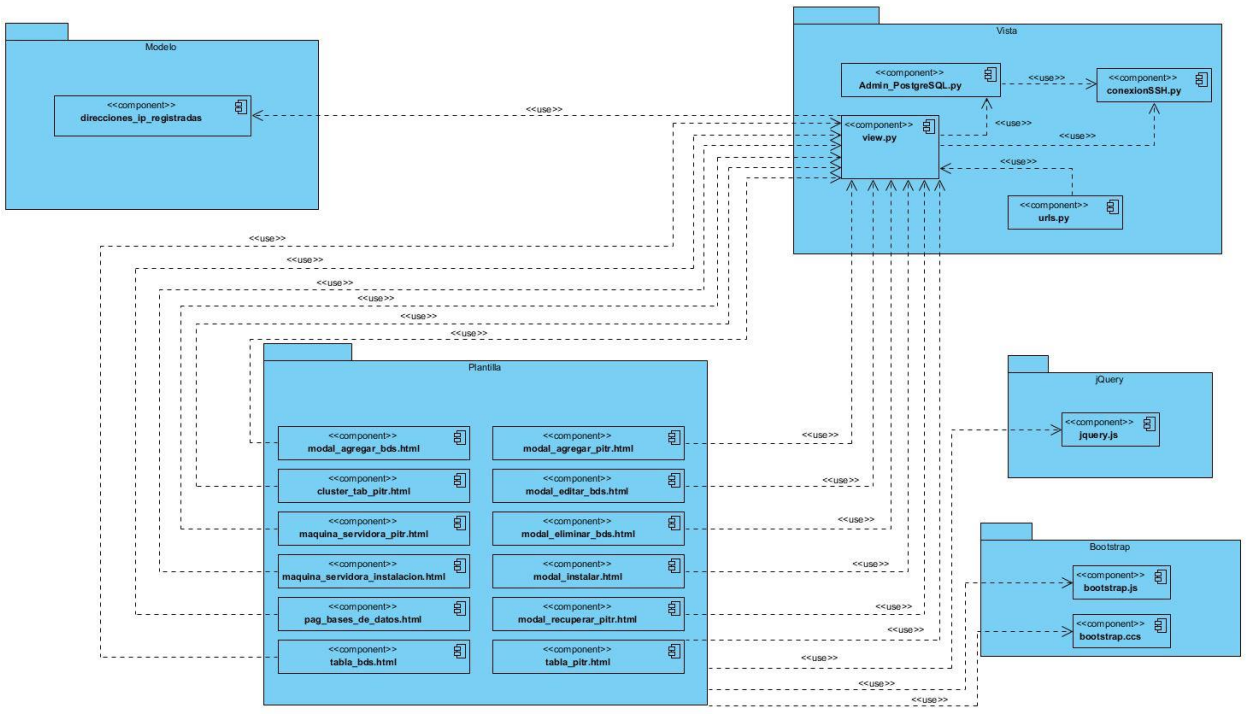


Figura 4: Diagrama de componentes

El diagrama de componentes está representado según el patrón MVT, incluyendo los marcos de trabajo Bootstrap y JQuery utilizados en la implementación de la herramienta, dicho diagrama queda estructurado de la siguiente forma:

• En el paquete “Plantilla” se representan los componentes “.HTML”, este paquete contiene las decisiones relacionadas a la presentación, en este caso se presentan en páginas web de formato html.

• En el paquete “Vista” se representan los componentes “.py” y contiene las clases controladoras del sistema, este paquete contiene la lógica que accede al modelo y delega la información a la plantilla apropiada.

• En el paquete “Modelo” se representan los componentes que son las entidades encargadas de la manipulación de los datos en el sistema.

2.8. Patrones de diseño

Son principios generales de soluciones que emplean ciertos estilos que ayudan a la creación de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (Larman, 2003).

Patrones GRASP:

Son patrones generales de software para asignar responsabilidades, los patrones GRASP describen los principios fundamentales para la asignación de responsabilidades a objetos, expresado en forma de patrones (Larman, 2003). A continuación, son descritos los de este tipo que son utilizados para el diseño de la solución propuesta.

Experto: Es un patrón que permite asignar la responsabilidad de realizar una labor a la clase que contiene toda la información necesaria para cumplir la tarea encomendada. En el caso de la presente investigación se tiene la clase “Admin_PostgreSQL” que es experta en realizar las configuraciones de PostgreSQL ya que contiene toda la información necesaria para realizar esa tarea.

Creador: Es un patrón que tiene responsabilidades relacionadas con la creación de objetos. El propósito fundamental del mismo, es encontrar un creador que se debe conectar con el objeto

producido en cualquier evento. Uno de los ejemplos del patrón en la aplicación se visualiza a continuación, en la clase “BDR” la cual es la encargada de crear objetos de tipo “BDR”.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras. Acoplamiento bajo significa que una clase no depende de muchas otras. El uso de este patrón permite que las clases no se afecten por cambios de otros componentes, haciendo posible que sean fáciles de entender y de reutilizar. En el caso de la presente investigación se muestra este patrón en el diagrama de clase de diseño (ver figura 7), donde se puede notar que las clases tienen solo dos o menos dependencias y esto proporciona que se reduzca el impacto de cambios posteriores en el sistema.

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. Se hizo necesario la utilización de este patrón en el sistema en cuestión con el fin de controlar la complejidad de cada clase utilizada para mantener un buen comportamiento de las mismas. Por esta razón, las que se identificaron con una amplia cantidad de funcionalidades, se dividieron en otras clases siguiendo el propósito de distribuir de forma equitativa el peso de la complejidad, manteniendo, además, la coherencia entre ellas.

Patrones GOF:

Los patrones GOF constituyen una herramienta fundamental para cualquier programador. Se utilizan para solucionar problemas de creación de instancias, ayudando a encapsular y abstraer dicha creación. Los patrones GOF presentan tres tipos de clasificaciones (patrones de comportamiento, patrones de creación y patrones estructurales) los cuales son utilizados en dependencia del propósito que se quiere alcanzar, a continuación se describe el patrón utilizado en la presente investigación.

Patrones de Comportamiento:

Tratan la interacción y la cooperación entre clases u objetos (Larman 1999).

El patrón de comportamiento GoF utilizado en el desarrollo de las funcionalidades es uno de los que usa el marco de trabajo Django, patrón de comando (Command Pattern por sus siglas en inglés), encapsula una petición como un objeto, permitiendo ejecutar dicha operación sin

necesidad de conocer el contenido de la misma (Moran y Araujo 2018). Este patrón evidenciado en el marco de trabajo django que encapsula sus peticiones, permite que la información viaje de forma segura por el sistema.

2.9. Modelo de diseño

El modelo de diseño expande y detalla los modelos de análisis tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente durante la fase de implementación (Pressman, 1998). El modelo de diseño se realizó haciendo uso de la herramienta Visual Paradigm donde se definió una estructura de paquetes para la organización del trabajo mostrada a continuación.

2.9.1. Diagrama de clases de diseño

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones. Los diagramas de clase son el pilar básico del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño) (…) (Larman, 2003). En estos se muestran las clases que componen un sistema y cómo se relacionan entre sí. A continuación, se muestra el diagrama de clases de diseño correspondiente con el paquete “Gestionar_Puntos_de_Recuperación” (ver figura 7), el resto de los diagramas pueden encontrarse en el **Anexo 4 Diagramas de clases de diseño.**

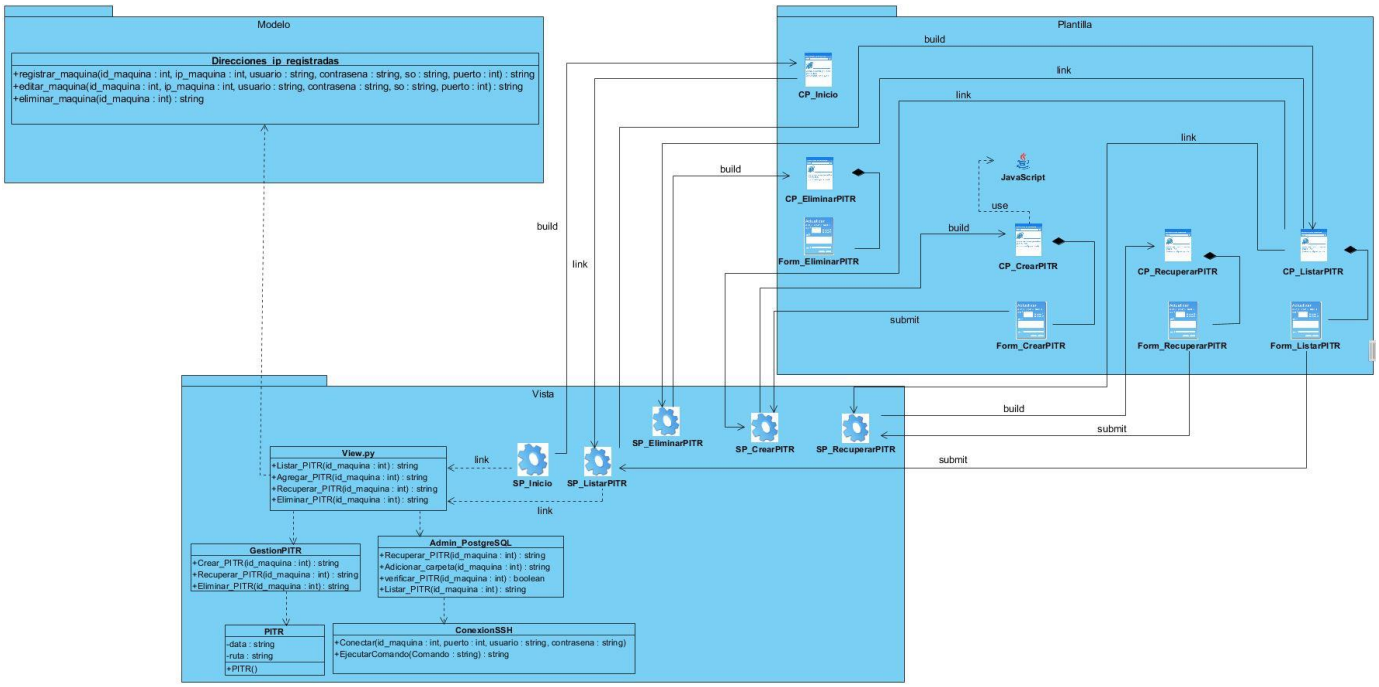


Figura 5: Diagrama de clase de diseño del paquete Gestionar_Puntos_de_Recuperación

Descripción de las Clases:

Client page (CP): Representan una página web, con formato HTML. Son interpretadas por el navegador y encargadas de mostrar los formularios (Form) e información al usuario.

Form: Colección de elementos de entrada que son parte de una página cliente. Sus atributos son los elementos de entrada del formulario.

Server page (SP): Son clases encargadas de generar las Client Page, representan la página web que tiene código que se ejecuta en el servidor.

View.py: Representa la lógica del proyecto.

AdministracionPostgreSQL: Es la clase encargada de realizar todas las funcionalidades relacionadas con el servicio PostgreSQL.

ConexionSSH: Es la clase por la cual se comunica la aplicación con el protocolo SSH configurado en la máquina virtual servidora.

GestionPITR: Es la clase encargada de gestionar los puntos de recuperación.

PITR: Es la clase encargada de generar los objetos de tipo “puntos de recuperación”.

Direcciones_ip_registradas: Es la clase encargada de acceder a los datos de la base de datos para proporcionar al sistema la dirección ip de las máquinas servidoras.

2.10. Diagrama de secuencia

Para una acción determinada en el sistema este diagrama representa la interacción entre diversos objetos a través de las llamadas ejecutadas y de retorno, lo que permite ver la secuencia de llamadas a través del tiempo (Cardozzo, 2016). Con dichos diagramas se muestran las interacciones entre las clases mediante la transferencia de mensajes. A continuación, se muestran los diagramas de secuencia correspondientes con el paquete Gestionar_Puntos_de_Recuperación (ver Figura 6, Figura 7, Figura 8 y Figura 9), el resto de los diagramas pueden encontrarse en el **Anexo 5 Diagramas de secuencia**.

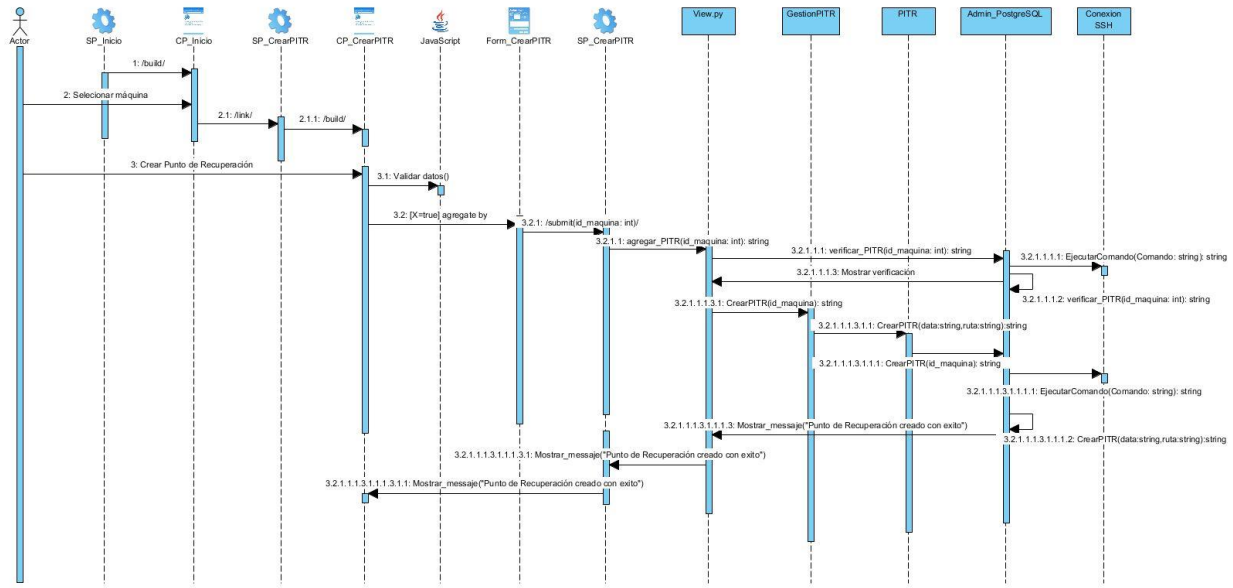


Figura 6: Diagrama de secuencia: Escenario crear punto de recuperación

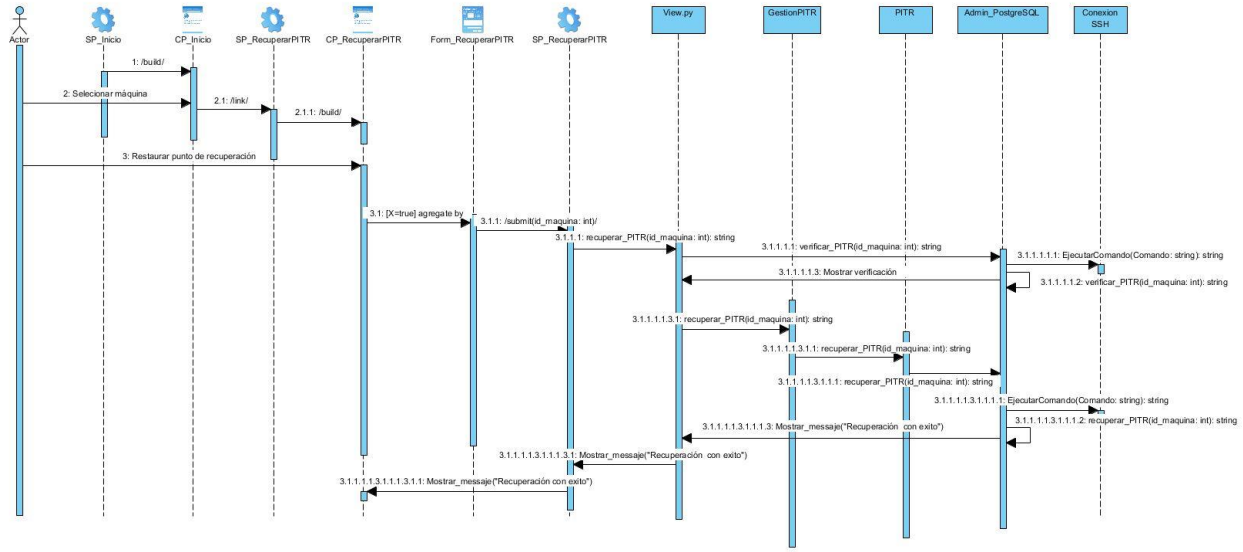


Figura 7: Diagrama de secuencia: Escenario restaurar punto de recuperación

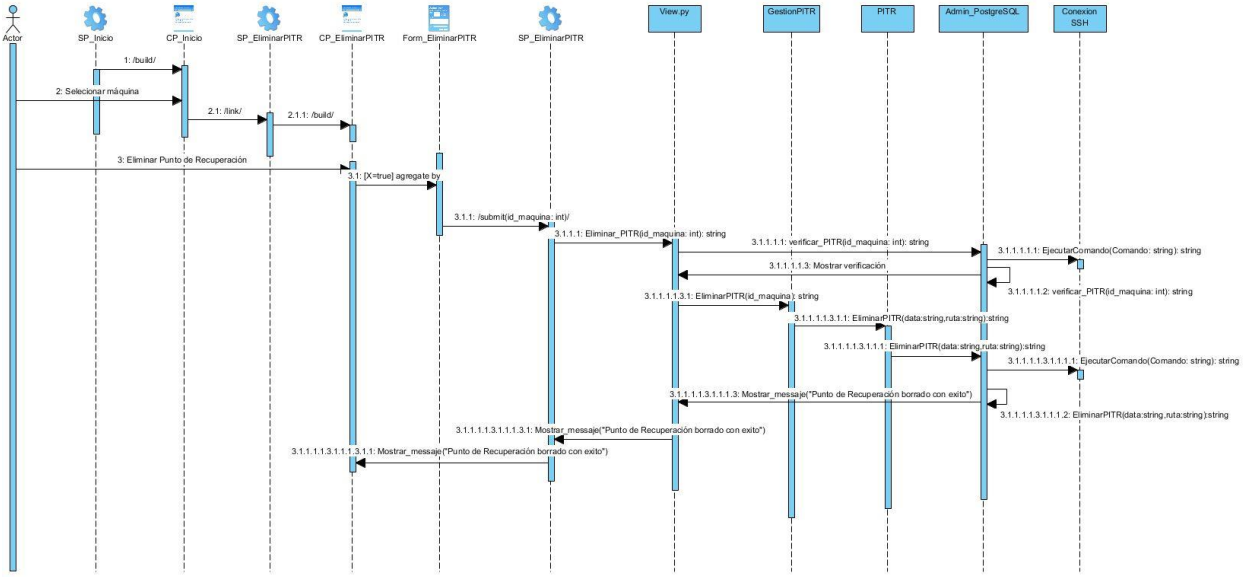


Figura 8: Diagrama de secuencia: Escenario Eliminar punto de recuperación

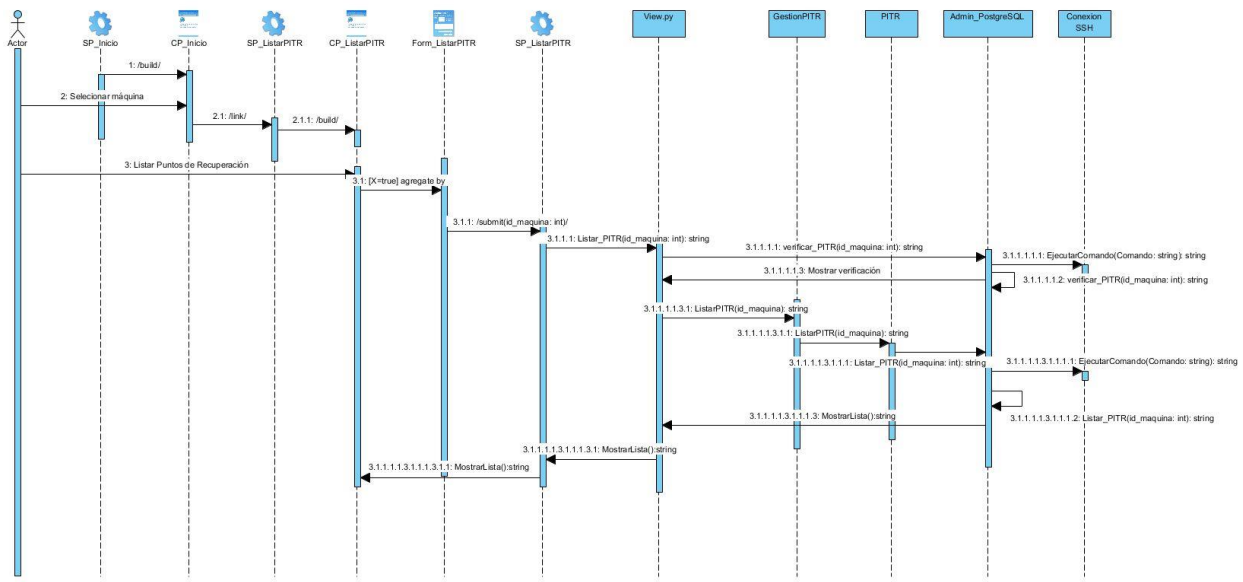


Figura 9: Diagrama de secuencia: Escenario listar puntos de recuperación

2.11. Conclusiones del capítulo

A partir de la aplicación de la técnica de obtención de requisito entrevista no estructurada se obtuvieron un total de 30 requisitos funcionales agrupados en tres paquetes de funcionalidades. Estos fueron especificados en un total de 30 Historias de Usuarios. Se obtuvieron además un conjunto de 9 requisitos no funcionales. Estos fueron especificados por atributos de calidad siguiendo las especificaciones de la metodología AUP asumida. Todos estos requisitos fueron validados a partir del acta de conformidad del cliente.

Se definieron como patrones de diseño a utilizar los patrones GRAP: Experto, Creador, Bajo acoplamiento y Alta cohesión. Ellos permitieron especificar la estructura y comportamiento de las clases, presentando una solución que atiende los niveles de reutilización y mantenimiento.

Se definió el modelo de diseño con el cual se obtuvieron un total de 3 diagramas de clases de diseño para mostrar lo que el sistema puede hacer (análisis), y para mostrar cómo puede ser construido (diseño). Se obtuvieron además 27 diagramas de secuencia para mostrar las interacciones entre las clases mediante la transferencia de mensajes.

Por consiguiente, una vez precisadas las funcionalidades y la estructura de la solución es posible dar inicio a la fase de implementación.

CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA

El presente capítulo muestra los artefactos ingenieriles relacionados con la implementación y validación del sistema a desarrollar. Entre los principales elementos se encuentran el modelo de implementación con los respectivos artefactos generados a partir de ellos. Además, se detallan las pruebas realizadas al sistema.

3.1. Modelo de implementación

Tiene como objetivo implementar cada una de las clases significativas del diseño.

Estándares de codificación

Los estándares de codificación se definen por el equipo de desarrollo para lograr la estandarización en la programación del software. Se basan en la estructura y apariencia física de un programa, con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Con el fin de facilitar el entendimiento del código y fijar un modelo a seguir, se establecieron los siguientes estándares de codificación:

CapWords: Para nombrar las clases, definiendo que las mismas deben comenzar con letra mayúscula, y en caso de estar compuesta por más de una palabra, todas deben iniciar con mayúscula. A continuación, se muestra una imagen que demuestra la utilización del estándar en el sistema.

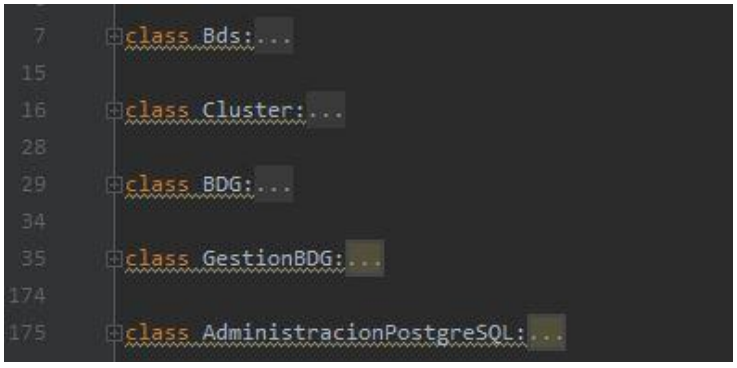


Figura 10: Ejemplo de uso del estándar de codificación CapWords

lower_case_with_underscores: Para nombrar los métodos del programa principal, definiendo que las palabras son escritas en minúsculas separadas por guion bajo. A continuación, se muestra una imagen que demuestra la utilización del estándar en el sistema.

```
808     def eliminar_pitr(self, cnombre, datapitr,cpuerto):...
824
825     def listar_versiones_disp(self, so):...
838
839     def subir_fichero_instalacion_local(self, vers):...
887
888
```

Figura 11: Ejemplo de uso del estándar de codificación lower_case_with_underscores

3.2. Diagrama de despliegue

Un diagrama de despliegue es utilizado para representar la distribución física de un sistema. El mismo muestra la configuración y la comunicación entre las instancias de los componentes y objetos que residen en ellos, A continuación, se presenta el diagrama de despliegue del sistema AD&MONPSQL, visto que para suportar las nuevas funcionalidades no fue necesario hacer cambios en el mismo:

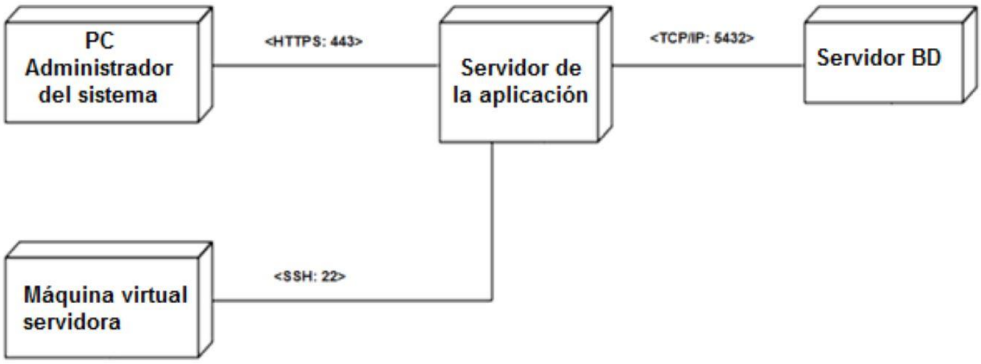


Figura 12: Diagrama de despliegue

Para la distribución física del sistema se utilizó un nodo “PC_Administrador del sistema” que representa la computadora del usuario. La PC_Administrador del sistema se conecta por el protocolo seguro de transferencia de hipertexto (https por sus siglas en inglés) al nodo “Servidor de la aplicación” web. Con el nodo del Servidor de la aplicación se puede acceder a la “Máquina

servidora” con el protocolo SSH y con el nodo “Servidor de BD” mediante el protocolo diseñado para facilitar la reutilización de código de base de datos (TCP/IP).

3.3. Pruebas del software

Las pruebas del software constituyen una etapa imprescindible durante el proceso de desarrollo del software. Su objetivo principal es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. Es importante considerar que las pruebas de software no garantizan que un sistema esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles para su debida corrección.

Estrategia de prueba

Trazar una estrategia de prueba es primordial para realizar la ejecución de las pruebas de un software. Mediante esta quedan plasmado los niveles de prueba a tratar, los tipos de pruebas que se deben llevar a cabo por cada nivel, los métodos de pruebas a aplicar y las técnicas a utilizar por el método.

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requieran. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (Pressman, 2010).

3.4. Disciplinas de pruebas

La metodología AUP para la UCI plantea varias disciplinas para guiar el desarrollo de las pruebas, a continuación, se describen las disciplinas presentes en la investigación:

Pruebas internas: En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.

Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas (Sánchez, 2015).

Pruebas de Aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Sánchez, 2015).

3.5. Niveles de Prueba

A continuación, se puntualizan los niveles de pruebas que corresponden a las disciplinas de pruebas descritas anteriormente. Dichos niveles propiciaron la detección de los errores existentes.

Unitarias: Las pruebas de unidad se concentran en la lógica del procesamiento interno y en las estructuras de datos de los límites de un componente. El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas (Rumbaugh y Jacobson, 2014). Se puede decir que consiste en aislar cada parte del programa (clases, módulos, objetos, paquetes, subsistemas) para comprobar que cada una de esas partes funciona correctamente por separado.

Aceptación: En este nivel se evidencian las pruebas realizadas por el usuario en un entorno muy similar al de producción para demostrar que el sistema cumple las especificaciones funcionales y requisitos del cliente. Son las últimas pruebas realizadas donde el cliente prueba el software y verifica que cumpla con sus expectativas, estas son fundamentales por lo cual deben incluirse obligatoriamente en el plan de pruebas de software (Rumbaugh y Jacobson, 2014).

Sistema: Tienen como propósito fundamental ejercitar profundamente el sistema desarrollado, con el objetivo de verificar que se hayan integrado correctamente todos los elementos del sistema y que realizan correctamente las funciones descritas. Este tipo de pruebas estudia el producto completo para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad o rendimiento (Rumbaugh y Jacobson, 2014).

3.6. Método de prueba

La metodología propone 2 métodos fundamentales que serán usados en el proceso de desarrollo de las pruebas al sistema, los cuales serán puntualizados a continuación.

Pruebas de caja negra:

Las pruebas de cajas negras o funcionales son realizadas a la interfaz del software y para aplicarlas solo se necesita conocer la funcionalidad del mismo. Estas pruebas se centran en los requisitos funcionales, permitiendo derivar conjuntos de condiciones de entradas que ejercitarán por completo dichos requerimientos.

Para la realización de los casos de prueba del sistema propuesto se selecciona el criterio de **partición de equivalencia** que es una técnica que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software y permite examinar los valores válidos e inválidos de las entradas existentes.

Prueba de caja blanca:

Estas pruebas, también suelen ser llamadas estructurales o de cobertura lógica. En ellas se pretende investigar sobre la estructura interna del código, exceptuando detalles referidos a datos de entrada o salida. Realizan un seguimiento del código fuente según se va ejecutando los casos de prueba, determinándose de manera concreta las instrucciones y/o bloques; que han sido ejecutados. A continuación, se describen las técnicas utilizadas para la realización del método.

Detección y corrección de errores: Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (Moreno y Sánchez, 2013).

Ruta o trayectoria básica (camino básico): Permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (Pressman, 2010).

3.7. Tipos de pruebas

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el sistema cumple con todos los requisitos identificados en el proceso de análisis y comprobar su correcto funcionamiento. A continuación, se explican los tipos de pruebas seleccionados:

Pruebas unitarias: Enfocan los esfuerzos de verificación en la unidad más pequeña del diseño de software. Al usar la descripción del diseño de componente como guía, las rutas de control importantes se prueban para descubrir errores dentro de la frontera del módulo. Se enfocan en la lógica de procesamiento interno y de las estructuras de datos dentro de las fronteras de un componente. Este tipo de pruebas puede realizarse en paralelo para múltiples componentes (Pressman, 2010).

Pruebas funcionales: Son un proceso de control de calidad que consiste en asegurar el cumplimiento de un sistema o componente con requerimientos funcionales. El objetivo principal de las pruebas funcionales es analizar el producto terminado y determinar si cumple con todo lo que debería hacer y si lo hace correctamente. Este tipo de pruebas entran dentro de lo que se llaman pruebas de caja negra: aquí no se centra en cómo se generan las respuestas del sistema, solo se analizan los datos de entrada y los resultados obtenidos (Oterino, 2014).

Pruebas de aceptación: Son realizadas con el cliente y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, sino una vez pasadas todas las pruebas por parte del desarrollador. Pueden realizarse durante un período de semanas o meses, y mediante ellas implementación y pruebas del componente propuesto se descubren errores acumulados que con el tiempo puedan degradar el sistema. La mayoría de los constructores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer sólo el usuario final es capaz de encontrar (Pressman, 2010).

3.8. Diseños de Casos de Prueba

Un Diseño de Caso de Prueba (DCP) está compuesto por un conjunto de entradas, respuestas (que emite el sistema de acuerdo a esas entradas) y el flujo central que indica el camino del escenario descrito. Estos son desarrollados para verificar el cumplimiento total o parcial de un requisito. Las entradas representan las variables que se pueden especificar y las mismas contienen: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor de la variable en un determinado caso, ya que es irrelevante.

A continuación, se muestran los casos de pruebas generados para el paquete de funcionalidades “Gestionar_Base_de_Datos_Relacional”:

Tabla 15: Caso de Prueba Gestionar BDR. Sección: Crear BDR

Escenario	**Descripción**	**Nombre**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear base de datos relacional satisfactoriamente.	Se selecciona la opción Bases de datos. Se selecciona la opción adicionar base de datos, se llenan los datos requeridos y se selecciona la opción crear.	V	El sistema muestra un mensaje informando que la base de datos fue creada exitosamente.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar crear base de datos.
EC 1.2 Error en la validación	Se selecciona la opción Bases de datos. Se selecciona la opción adicionar	I	El sistema muestra un mensaje informando que el nombre de la	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina

	base de datos, se llenan los datos requeridos y se selecciona la opción crear.		base de datos es inválido.	servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar crear base de datos.
EC 1.3 Seleccionar la opción cancelar.	El usuario selecciona la opción cancelar en el formulario de creación de la base de datos.	NA	El sistema vuelve a la página anterior	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.
EC 1.4 Campos vacíos.	El usuario selecciona la opción aceptar sin llenar los campos.	I	El sistema muestra un mensaje informando que existen campos vacíos.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.

Tabla 16: Descripción la variable correspondiente a la sección crear BDR

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Cadena de caracteres con un máximo de 50 elementos

Tabla 17: Caso de Prueba Gestionar BDR. Sección: Editar BDR

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
EC 2.1 Editar base de datos relacional satisfactoriamente.	Se selecciona la opción Bases de datos. Se selecciona la opción editar base de datos, se llenan los datos requeridos y se selecciona la opción aceptar.	V	El sistema muestra un mensaje informando que el cambio en la base de datos fue realizado exitosamente.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.
EC 2.2 Error en la validación	Se selecciona la opción Bases de datos. Se selecciona la opción editar base de datos, se llenan los datos requeridos y se	I	El sistema muestra un mensaje informando que el nombre de la base de datos es inválido.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos.

	selecciona la opción aceptar.			4-Seleccionar editar base de datos.
EC 2.3 Seleccionar la opción cancelar.	El usuario selecciona la opción cancelar en el formulario de editar base de datos.	NA	El sistema vuelve a la página anterior	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.
EC 2.3 Campos vacíos.	El usuario selecciona la opción aceptar sin llenar los campos.	I	El sistema muestra un mensaje informando que existen campos vacíos.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.

A continuación, se muestra la descripción las variables correspondientes con el caso de prueba “Editar BDR”.

Tabla 18: Descripción de la variable correspondiente a la sección editar BDR

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	Cadena de caracteres con un máximo de 50 elementos

Tabla 19: Caso de Prueba Gestionar BDR. Sección: Eliminar BDR

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
EC 3.1 Eliminar base de datos relacional satisfactoriamente.	Se selecciona la opción Bases de datos. Se selecciona la opción eliminar base de datos, selecciona la opción eliminar.	NA	El sistema muestra un mensaje informando que la base de datos fue eliminada exitosamente.	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar en la lista de BD la que se quiere eliminar. 5-Seleccionar eliminar base de datos.
EC 3.2 Seleccionar la opción cancelar.	El usuario selecciona la opción cancelar en el formulario de creación de	NA	El sistema vuelve a la página anterior	1-Autenticarse en el sistema. 2- Seleccionar la opción máquina

	la base de datos.			servidora. 3-Seleccionar la opción Bases de datos. 4-Seleccionar editar base de datos.

3.9. Resultados de las pruebas

Resultados de la aplicación de la técnica de camino básico asociada al método de caja blanca:

A continuación, se muestra el caso de prueba correspondiente a la función “verificar_pitr”, el cual se localiza en la clase Admin_PostgreSQL. El objetivo de este método es verificar si existen puntos de recuperación en el servidor. Se selecciona este método teniendo en cuenta la importancia que representa para el resultado de este trabajo. Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Para poder elaborar el grafo de flujo, primero se deben enumerar las sentencias del código. A continuación, se muestra el código fuente referente al método descrito anteriormente y cómo fue aplicada la Técnica de Camino Básico al mismo:

```
def verificar_pitr(self, cnombre,cpuerto):
    vers = self.Obtener_clusterp(cpuerto)
    esta = False
    if self.so == 'debian' or self.so == "ubuntu":
        salida, errores = self.conex.EjecutarComando("find /var/lib/postgresql/"+vers.version+"/recovery/ -name "+cnombre+"")
        if len(salida) != 0:
            esta = True
    if self.so == 'centos7':
        salida, errores = self.conex.EjecutarComando(
            "find /var/lib/pgsql/9.6/ -name recovery")
        if len(salida) != 0:
            esta = True

    return esta
```

Figura 13: Código del método verificar_pitr()

Posteriormente se procede a la elaboración del grafo de flujo teniendo en cuenta dicha enumeración:

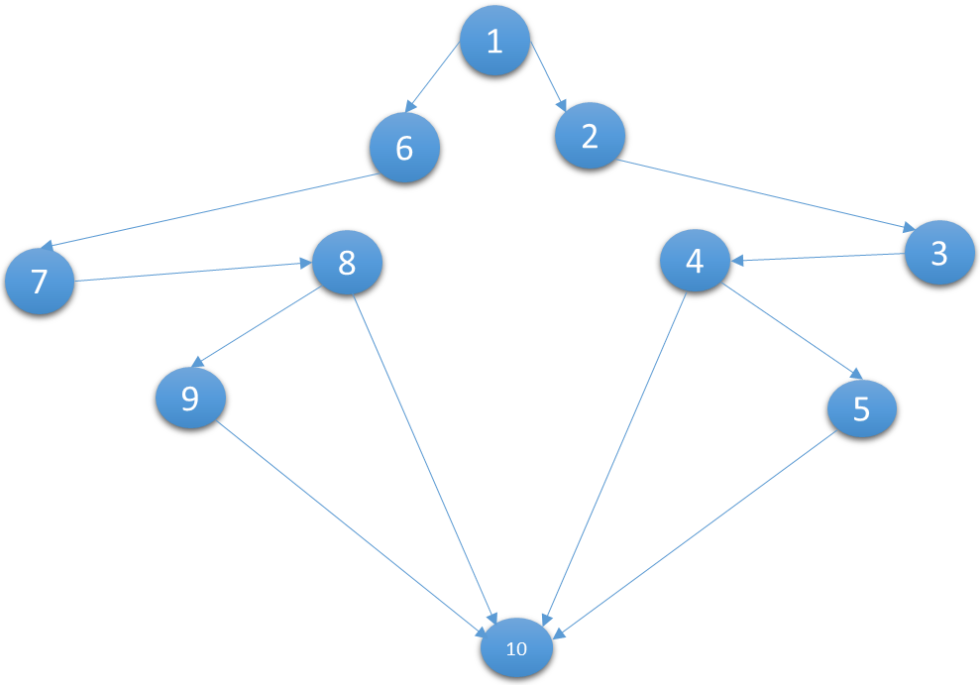


Figura 14: Representación del grafo de flujo de camino básico del método verificar_pitr()

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (Pressman, 2010). La complejidad ciclomática se basa en la teoría gráfica y para que el cálculo sea correcto, todos deben arrojar el mismo resultado:

El número de regiones corresponde a la complejidad ciclomática “V (G)”.

V (G)= R Donde R es la cantidad total de regiones.

V (G)= 4

V (G)= E – N + 2 Donde E es el número de aristas y N número de nodos.

V (G)= 12 – 10 + 2

V (G)= 4

V (G)= P + 1 Donde P es la cantidad de nodos predicados.

V (G)= 3 + 1 = 4

El valor V (G) expresa la cantidad de caminos linealmente independientes de la estructura de control del programa, por lo que se definen los siguientes 4 caminos:

Camino básico 1: 1-2-3-4-10

Camino básico 2: 1-2-3-4-5-10

Camino básico 3: 1-6-7-8-10

Camino básico 4: 1-6-7-8-9-10

Cada camino independiente es un caso de prueba a realizar, lo que se garantiza que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. En el caso anterior se calcularon dos caminos básicos, por tanto, surge la necesidad de efectuar igual número de casos de prueba. A continuación, se muestran los casos de pruebas realizados:

Tabla 20: Caso de prueba de caja blanca para el camino básico 1

Entrada:	N/A
Resultados Esperados:	No se encuentran puntos de recuperación en el sistema Debian o Ubuntu.
Condiciones:	Se modifican los ficheros y se crean las carpetas en la máquina servidora para la creación de puntos de recuperación.

Tabla 21: Caso de prueba de caja blanca para el camino básico 2

Entrada:	N/A
Resultados Esperados:	Se encuentran puntos de recuperación en el sistema Debian o Ubuntu.
Condiciones:	Se crea un nuevo punto de recuperación en las carpetas existentes en la máquina servidora.

Tabla 22: Caso de prueba de caja blanca para el camino básico 3

Entrada:	N/A
Resultados Esperados:	No se encuentran puntos de recuperación en el sistema CentOS 7.
Condiciones:	Se modifican los ficheros y se crean las carpetas en la máquina servidora para la creación de puntos de recuperación.

Tabla 23: Caso de prueba de caja blanca para el camino básico 4

Entrada:	N/A
Resultados Esperados:	No se encuentran puntos de recuperación en el sistema CentOS 7.
Condiciones:	Se modifican los ficheros y se crean las carpetas en la máquina servidora para la creación de puntos de recuperación.

Una vez realizadas las pruebas unitarias mediante la técnica de camino básico al código seleccionado, se puede afirmar que la prueba concluye de forma satisfactoria, por tanto, se obtuvo el resultado esperado.

Resultados de la aplicación de la técnica de partición de equivalencia asociada al método de caja negra:

A través del método de caja negra, y apoyados en el diseño de Casos de Prueba se realizaron tres iteraciones de pruebas internas pertenecientes al nivel de sistema. Dichas pruebas fueron realizadas con el objetivo de detectar y corregir errores que impidieran el correcto funcionamiento de la solución.

Para evaluar la solución se realizaron 3 iteraciones donde se probó el software íntegramente, finalmente se realizó una prueba final donde se comprobó la resolución de todas las no conformidades detectadas. A continuación, se presentan los resultados arrojados durante las diferentes pruebas aplicadas:

Tabla 24: Resultados de la prueba de caja negra 1ra iteración

Casos de pruebas	No conformidades			
	Alta	Media	Baja	Total
Gestión BDR	1	4	3	8
Gestión PITR	3	1	2	6
Instalación PostgreSQL	1	3	6	10
Total	5	8	11	24

Tabla 25: Resultados de la prueba de caja negra 2ra iteración

Casos de pruebas	No conformidades			
	Alta	Media	Baja	Total
Gestión BDR	0	3	2	5
Gestión PITR	2	1	1	4
Instalación PostgreSQL	1	2	3	6
Total	3	6	6	15

Tabla 26: Resultados de la prueba de caja negra 3ra iteración

Casos de pruebas	No conformidades			
	Alta	Media	Baja	Resuelta
Gestión BDR	0	1	0	1
Gestión PITR	1	0	1	2
Instalación PostgreSQL	1	2	1	4
Total	2	3	2	7

Las pruebas se centraron en el cumplimiento de los paquetes de funcionalidades descritos en el capítulo II del presente documento. Las no conformidades se clasificaron en Alta, Media o Baja en dependencia del impacto que tuvieran, generalmente las altas responden a errores técnicos relacionados directamente con la funcionalidad interna del paquete de funcionalidad y las bajas tienden a ser errores ortográficos y validaciones.

La figura muestra en una gráfica de barras los principales resultados de las pruebas de caja negra realizadas. Como se puede apreciar, se realizaron 3 iteraciones de pruebas, detectando en la primera, segunda y tercera iteración un total de 24, 15 y 7 no conformidades respectivamente. Comparando los resultados entre dichas iteraciones se evidencia una reducción considerable de los principales errores detectados en el sistema una vez ejecutada la 3ra iteración. Finalmente se realizó una última prueba donde se corrigieron todas las no conformidades. Se recomienda realizar la solicitud a Calidad Centro y Calidad UCI para realizar las correspondientes pruebas de liberación, las cuales no forman parte del alcance de este trabajo de diploma, debido al tiempo que demoran las mismas.

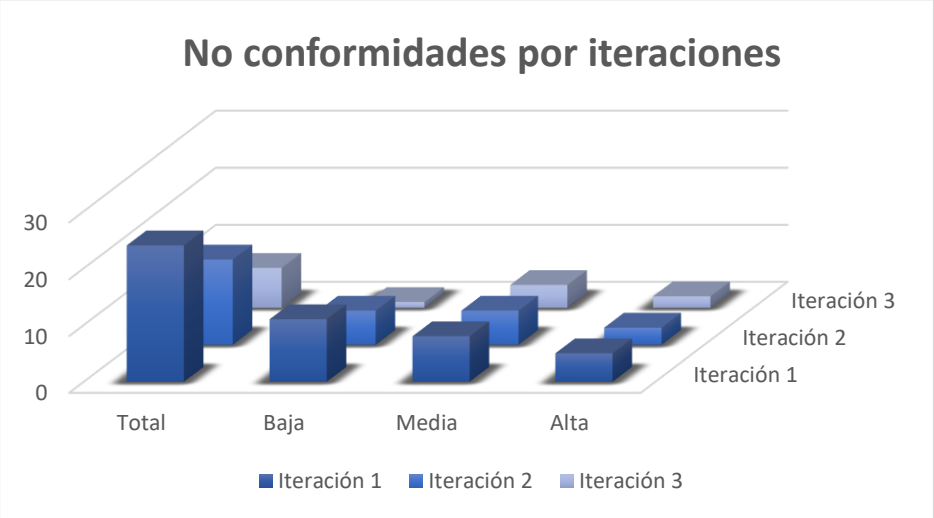


Figura 15: No conformidades detectadas contra cantidad de iteraciones de prueba.

Pruebas de aceptación:

Se empleó la técnica de prueba alfa, por el cliente y se evaluaron las funcionalidades del sistema basándose en la especificación de requisitos del expediente de proyecto. Los interesados verificaron cada requisito funcional y comprobaron si estos correspondían con los requerimientos planteados. (Anexo 1 Acta de aceptación de desarrollo de productos informáticos).

Al ser concluida las pruebas, se liberó el sistema de administración para PostgreSQL, entregándole al equipo de desarrollo el acta de aceptación en la que consta que el sistema puede ser utilizado y cumple con las expectativas planteadas por el cliente.

3.10. Conclusiones del capítulo

1. Al realizar la implementación del sistema siguiendo los estándares de codificación definidos, se desarrolló un código reutilizable y comprensible por los integrantes del equipo de desarrollo.
2. La elaboración del modelo de despliegue permitió representar la distribución física del sistema. Mediante los casos de prueba funcionales.
3. Los diferentes métodos de pruebas aplicados a la solución durante el ciclo de vida del sistema permitieron comprobar los errores existentes y mejorar la calidad de los resultados.
4. El sistema desde el punto de vista funcional cumple con los requerimientos definidos y especificados en las primeras etapas de desarrollo a partir de las necesidades del cliente.

CONCLUSIONES

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriban a las siguientes conclusiones:

- A partir del reconocimiento de la problemática existente dentro del Centro de Telemática y la realización del estudio de las herramientas informáticas homólogas existentes, se demostró la viabilidad del sistema propuesto ya que ninguna cumplía con las necesidades específicas del área.
- El estudio realizado sobre las diferentes metodologías de software, definió a AUP-UCI como la más acertada para guiar el proceso de diseño e implementación del sistema que se desarrolló.
- La selección de las herramientas informáticas y lenguajes permitió establecer las bases para la implementación correcta del sistema.
- Se realizó una descripción del proceso que se realiza dentro del centro y a partir del entendimiento del mismo se propuso una solución capaz de satisfacer las necesidades del cliente. Se puntualizaron las pautas a seguir en la implementación del sistema posibilitando una mayor agilidad, claridad y organización en cuanto al código. Así como la realización de las pruebas que define la metodología AUP-UCI que permitieron validar y certificar el correcto funcionamiento del sistema.
- El producto obtenido permite realizar los procesos de administración (Instalar PostgreSQL, gestionar BDR, gestionar PITR) de servidores PostgreSQL de manera centralizada a través de una interfaz web.

Por todo lo anteriormente expuesto, se concluye que el objetivo propuesto para el presente trabajo ha sido cumplido satisfactoriamente. El sistema desarrollado contribuirá de manera significativa para la administración de servidores PostgreSQL en el centro de Telemática de la Facultad 2.

RECOMENDACIONES

Tomando como base la investigación realizada y la experiencia acumulada durante la realización de este trabajo de diploma, se proponen las siguientes recomendaciones:

- Desarrollar la funcionalidad que permita la creación de relaciones entre las bases de datos haciendo mejoras en la gestión de las mismas.
- Desarrollar las funcionalidades necesarias para que el sistema pueda gestionar bases de datos y hacer el respaldo de información en las máquinas que tienen como sistema operativo Windows server.
- Desarrollar la funcionalidad necesaria para que el sistema sea capaz de brindar al usuario la opción de cambiar de versión a gestionar en la máquina servidora con sistema operativo CentOS 7.

REFERENCIAS BIBLIOGRÁFICAS

1. **Caldevilla, A. y Méndez, J.** Sistema para la administración y monitoreo de servidores PostgreSQL. Trabajo de Diploma, Universidad de las Ciencias Informáticas, Havana, 2018.
2. **Benítez, M. y Arias, Á.,** Curso de Introducción a la Administración de Bases de Datos: 2ª Edición, IT Campus Academy, 2017. 194.
3. **Casasayas, L.** Módulo de administración para el servicio PostgreSQL en la Herramienta para la Migración y Administración de Servicios Telemáticos. Trabajo de Diploma, Universidad de las Ciencias Informáticas, Havana, 2014.
4. **LARMAN, C.** UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado, Madrid, Pearson Educación, S.A., 2003, 624.
5. **Pérez, J.** Introducción a javascript. 2012, [visitado 2019-11-19]. Disponible en: http://www.librosweb.es/javascript**.**
6. **Gauchat, D.** El gran libro de HTML5, CSS3 y Javascript. 2012, [visitado 2019-10-15]. Disponible en: www.minkbooks.com.
7. **Fuentes, M.** Bases de Datos. 2013, [visitado 2019-12-16]. [En línea] 2013. Disponible en:http://www.cua.uam.mx/pdfs/conoce/libroselec/Notas_del_curso_Bases_de_Datos.pdf.
8. **León, H.; Coello, S., et al.** El proceso de investigación científica. s.l. : Editorial Universitaria del MINISTERIO DE EDUCACIÓN SUPERIOR, 2012.
9. **Holovaty, J.; Kaplan-Moss, A.,** The Definitive Guide To Django. Ed. Por APRESS,Editorial, 2009.
10. **Iruela, Juan. 2016.** Los gestores de bases de datos más usados. Revista digital INESEM.
11. **Marini, E.** El Modelo Cliente/Servidor, [visitado 2019-12-16], 2012. [En línea]. Disponible en: https://www.linuxito.com/docs/elmodelo-cliente-servidor.pdf
12. **Moreno, A.; Sánchez, C.** Manejo de sistemas de información para la organización de procesos de la gerencia de protección y aseguramiento de ingresos de la compañía Claro Colombia SA. 2013.
13. **Oterino, A.** ¿Pruebas de integración, funcionales, de carga? ¿Qué diferencias hay?, 2014
14. **Pressman, R. S.** Ingeniería del software. Un enfoque práctico. Cuarta edición,1998.

15. **Pressman, R.S.** SOFTWARE ENGINEERING A PRACTITIONER´S APPROACH SEVENTH EDITION. New York, McGRAW-HILL Education, 2010. 736.
16. **Cardozzo, D.** Desarrollo de Software: Requisitos, Estimaciones y Análisis. España, s.l.: Createspace Independent Pub, 2016, 145.
17. **PARNELL, B.,** De terabytes a zettabytes: el crecimiento de los datos mundiales. Think Progress, 2015. [online] [visitado 2019-11-05]. Disponible en: www.think-progress.com/es/blog/
18. **Rumbaugh J.; Jacobson I., et al.** El Proceso Unificado de Desarrollo de Software. Manual de Referencia. Addison Wesley. 2014.
19. **Sánchez, T.** Metodología de desarrollo para la Actividad productiva de la UCI. La Habana, 2015
20. **Sánchez, Y.; Yusbel L., et al.** Sistema Web Para La Gestión Del Control De Almacén En La Mini-Industria El Mambí Del Municipio De Florencia En La Provincia De Ciego De Ávila. [En línea] 2017, [visitado 2019-11-05] disponible en: http: //revistas.unica.cu/index.php/uciencia/article/view/302/1090.
21. **Ylonen, C.; Lonvick, T.** The secure shell (SSH) connection protocol, 2006.
22. **LARMAN, C.** UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado, Madrid, Pearson Educación, S.A., 2003, 624.
23. **INFANTE, C.A.F., 2013.** *GUÍA DE ELABORACIÓN DE MODELOS*. Bogotá D.C, Colombia: s.n.

BIBLIOGRAFÍA

1. **Omelchenko, A.,** Point-in-time recovery. *Sql Server Backup Academy* [en línea]. [Consulta: 3 diciembre 2019]. Disponible en: https://sqlbak.com/academy/point-in-time-recovery.
2. **Böszörményi, Z.,** *PostgreSQL Replication*. S.l.: Packt Publishing Ltd, 2013.
3. **Caldevilla, A.; Méndez, J.** Sistema para la administración y monitoreo de servidores PostgreSQL. Trabajo de Diploma, Universidad de las Ciencias Informáticas, Havana, 2018.
4. **Fajardo, S.** Sistema de administración de servidores PostgreSQL para el Departamento de Desarrollo de Componentes del Centro Telemática V2.0. Trabajo de Diploma, Universidad de las Ciencias Informáticas, Havana, 2019.
5. **PostgreSQL: Documentation: 9.6:** Continuous Archiving and Point-in-Time Recovery (PITR). [en línea], [sin fecha]. [Consulta: 3 diciembre 2019]. Disponible en: https://www.postgresql.org/docs/9.6/continuous-archiving.html.
6. **Stones, R.; Matthew, N.,** *Beginning Databases with PostgreSQL: From Novice to Professional*. S.l.: Apress. 2006. ISBN 978-1-4302-0018-5.
7. **Iruela, J.** *Los gestores de bases de datos más usados*. S.l.: Revista digital INESEM, 2016.
8. **Sánchez, T.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana. Cuba, 2015.
9. **Cervigón Hutaro, A.G. Y Alegre Ramos, M.D.P., 2011.** *Seguridad Informatica Ed.11 Paraninfo* [En Línea]. S.L.: S.N. [Consulta: 8 Diciembre 2019]. Disponible En: Https://Books.Google.Com/Books/About/Seguridad_Informatica_Ed_11_Paraninfo.Html?Hl=Pt-Pt&Id=C8kni5g2yv8c.
10. **Sommerville, I., 2011.** *Software Engineering*. 9. Ed. Harlow: Addison-Wesley. International Computer Science Series. Isbn 978-0-201-39815-1.
11. **Cardozzo, R,.** Desarrollo de Software: Requisitos, Estimaciones y Análisis. Createspace Independent Pub, 2016
12. **Zapata, Carlos M., Palacio, C. et al.** UNC-Analista: Lacia la captura de un corpus de requisitos a partir de la aplicación del experimento mago de oz. s.l. : Revista EIA, Junio 2007. ISSN 1794-1237

13. **Marini, Emiliano. 2012.** El Modelo Cliente/Servidor. [En línea]. https://www.linuxito.com/docs/elmodelo-cliente-servidor.pdf
14. **George, N., 2019.** *Build a Website With Django 3: A complete introduction to Django 3*. S.l.: GNW Independent Publishing.
15. **LARMAN, C.** UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado, Madrid, Pearson Educación, S.A., 2003, 624.
16. **Pressman, R.S. 2010.** Ingeniería de Software, un enfoque práctico. 2010. Séptima Edición
17. **Rumbaugh J.; Jacobson I., et al.** El Proceso Unificado de Desarrollo de Software. Manual de Referencia. Addison Wesley. 2014.
18. **Moreno, A.; Sánchez, C.** Manejo de sistemas de información para la organización de procesos de la gerencia de protección y aseguramiento de ingresos de la compañía Claro Colombia SA. 2013.
19. **Oterino, A.M. del C.G. 2014.** ¿Pruebas de integración, funcionales, de carga?¿Qué diferencias hay?
20. **Benítez, M. y Arias, Á.,** Curso de Introducción a la Administración de Bases de Datos: 2ª Edición, IT Campus Academy, 2017. 194.
21. **Caldera Vergara, Roberto. 2017.** Estudio del framework de desarrollo web Django. [En línea]. https://ebuah.uah.es/dspace/handle/10017/32018
22. **Casasayas, L.** Módulo de administración para el servicio PostgreSQL en la Herramienta para la Migración y Administración de Servicios Telemáticos. Trabajo de Diploma, Universidad de las Ciencias Informáticas, Havana, 2014.
23. **Condori A., Luis, J.** Phython – Django Framework de desarrollo web para perfeccionistas Basado en el Modelo MTV, 2012.
24. **DB-Engines. 2018.** DB-Engines. [En línea] 2018. https://db-engines.com/en/ranking
25. **Pérez, J.** Introducción a javascript. 2012, [visitado 2019-11-19]. Disponible en:http://sedici.unlp.edu.ar/bitstream/handle/10915/52183/Documento_completo..pdf?sequence=3.
26. **España, S.; Fernando, H.,** Documentación y análisis de los principales frameworks de arquitectura de software en aplicaciones empresariales. 2012, [visitado 2019-11-19]. Disponible en: http://www.librosweb.es/javascript**.**
27. **FOUNDATION, PYTHON SOFTWARE. 2018.** PEP 8 – Style Guide for Python Code. [En línea] 2018. https://www.python.org/dev/peps/pep-0008/

28. **Gauchat, Juan Diego. 2012.** El gran libro de HTML5, CSS3 y Javascript. 2012
29. **Gómez Fuentes, Dra. María del Carmen. 2013.** [En línea] 2013. http://www.cua.uam.mx/pdfs/conoce/libroselec/Notas_del_curso_Bases_de_Datos.pdf.
30. **Group, The PostgreSQL Global Development. 2018.** [En línea]. https://www.PostgreSQL.org/about/.
31. **Group, Visual Paradigm. 2018. Visual Paradigm. [En línea] 2018.** https: //www.visual-paradigm.com/.
32. **León, H.; Coello, S., et al.** El proceso de investigación científica. s.l. : Editorial Universitaria del MINISTERIO DE EDUCACIÓN SUPERIOR, 2012.
33. **Holovaty, J.; Kaplan-Moss, A.,** The Definitive Guide To Django. Ed. Por APRESS,Editorial, 2009.
34. **Iruela, J.** *Los gestores de bases de datos más usados*. S.l.: Revista digital INESEM, 2016.
35. **Jimenez, M. C. 2014.** Bases de datos relacionales y modelado de datos. s.l. : UF1471. IC Editorial, 2014
36. **Marini, Emiliano. 2012.** El Modelo Cliente/Servidor. [En línea]. https://www.linuxito.com/docs/elmodelo-cliente-servidor.pdf.
37. **Oterino, A.M. del C.G. 2014.** ¿Pruebas de integración, funcionales, de carga?¿Qué diferencias hay?
38. **Pressman, Roger S. 1998.** Ingeniería del software. Un enfoque práctico. 1998. Cuarta edición.
39. **Cardozzo, R,.** Desarrollo de Software: Requisitos, Estimaciones y Análisis. Createspace Independent Pub, 2016.
40. **Rumbaugh J.; Jacobson I., et al.** El Proceso Unificado de Desarrollo de Software. Manual de Referencia. Addison Wesley. 2014.
41. **SOLUTIONS, EMS DATABASE MANAGEMENT. 2016.** EMS SQLManager for PostgreSQL. [En línea] 2016. www.sqlmanager.net/products/PostgreSQL/manager
42. **Webmin.** Webmin. [En línea] 2017. https://doxfer.webmin.com/Webmin/Introduction.
43. **Ylonen, C.; Lonvick, T.** The secure shell (SSH) connection protocol, 2006.

ANEXOS

Anexo 1 Acta de aceptación de los requisitos

UCI Universidad de las Ciencias Informáticas

Acta de Aceptación de Requisitos
Calidad Centro Telemática

Sistema de administración de servidores PostgreSQL para el Centro de Telemática

Versión: 3.0

Mediante el presente documento se certifica que los 30 requerimientos funcionales del Sistema de administración de servidores PostgreSQL para el Centro de Telemática, quedan aprobados y validados por el cliente.

Observaciones:

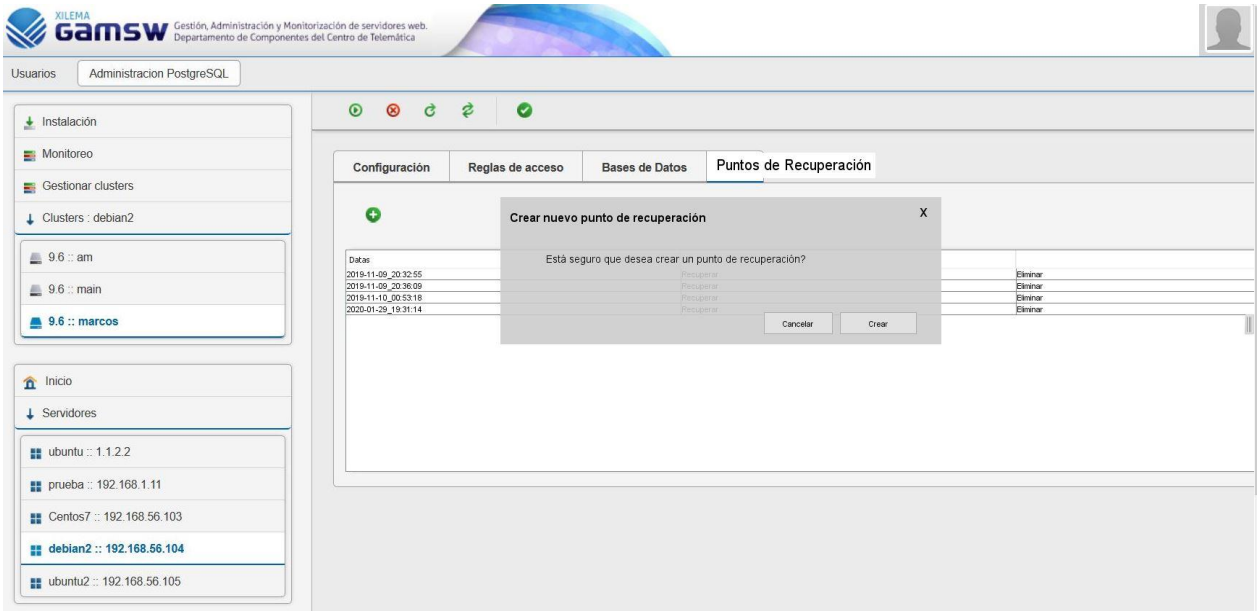
Recomendaciones:

Nombre del cliente: Antonio Hernández Dominguez

Firma:

Anexo 2 Historias de Usuarios

Número: 18	**Nombre del requisito:** Crear punto de recuperación en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.8
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días
Descripción: Funcionalidad que permite el usuario crear un punto de recuperación en el servidor Debian usando PITR, al dar pulsar la opción crear se debe abrir una caja de confirmación de la creación donde se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación creado con éxito!” o “Error en el servidor”). **Botones:** Cancelar: Opción que permite regresar al listado de puntos de recuperación. Crear: Opción que prosigue con la creación de un punto de recuperación.	
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL en una versión mayor que la 9.4.	
Prototipo de interfaz:	



Número: 16	**Nombre del requisito:** Crear punto de recuperación en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.8
Riesgo en Desarrollo: • Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite el usuario crear un punto de recuperación en el servidor CentOS 7 usando PITR, al dar pulsar la opción crear se debe abrir una caja de confirmación de la creación donde se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación creado con éxito!” o “Error en el servidor”).

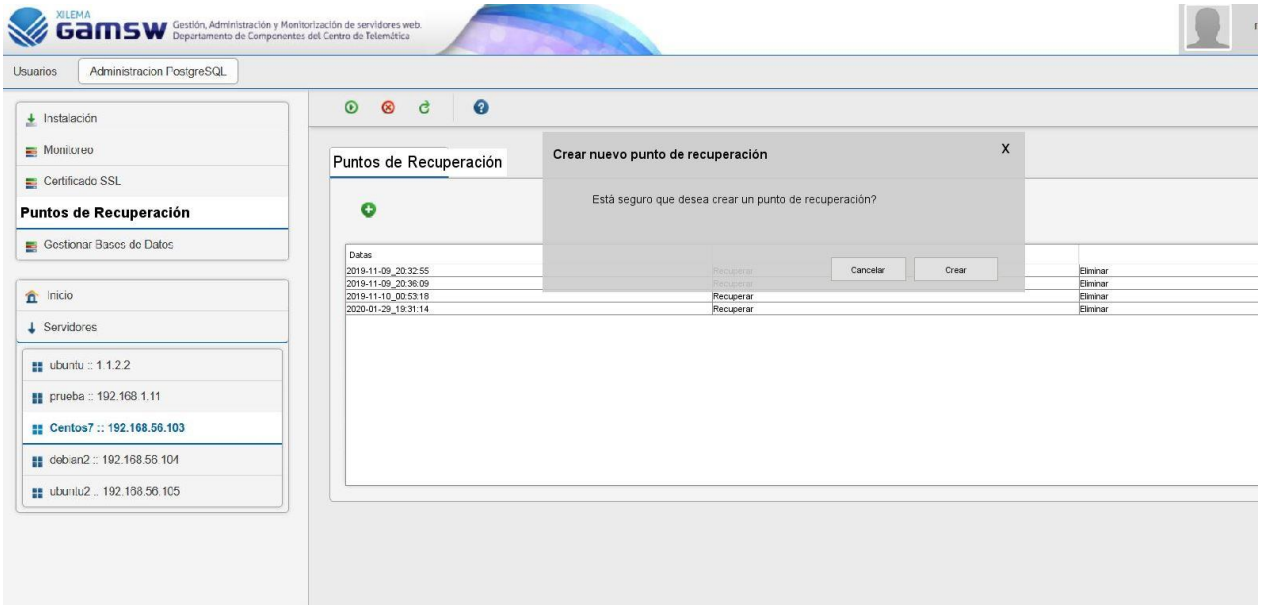
Botones:

Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Crear: Opción que prosigue con la creación de un punto de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL en una versión mayor que la 9.4.

Prototipo de interfaz:



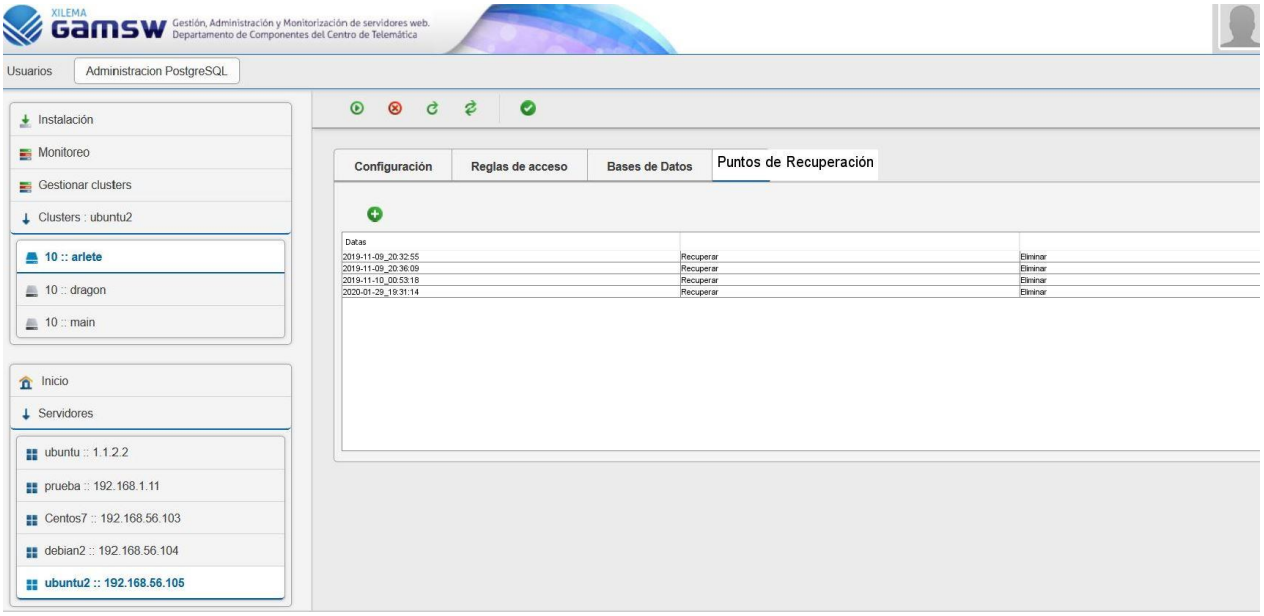
Número:14	**Nombre del requisito:** Listar Puntos de Recuperación en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1
Prioridad: Alta	**Puntos Estimados:** 0.4
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 día

Descripción: Funcionalidad que permite al usuario listar los puntos de recuperación existentes en el servidor con sistema operativo Ubuntu al que se está conectado. Se debe mostrar un listado de los puntos de recuperación existentes en el servidor con la opción de adicionar un punto de recuperación visible para el usuario, mostrar también para cada punto de recuperación la opción

de recuperar y eliminar, En caso de no existir ningún punto de recuperación se muestra un listado vacío.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL en una versión mayor que la 9.4.

Prototipo de interfaz:

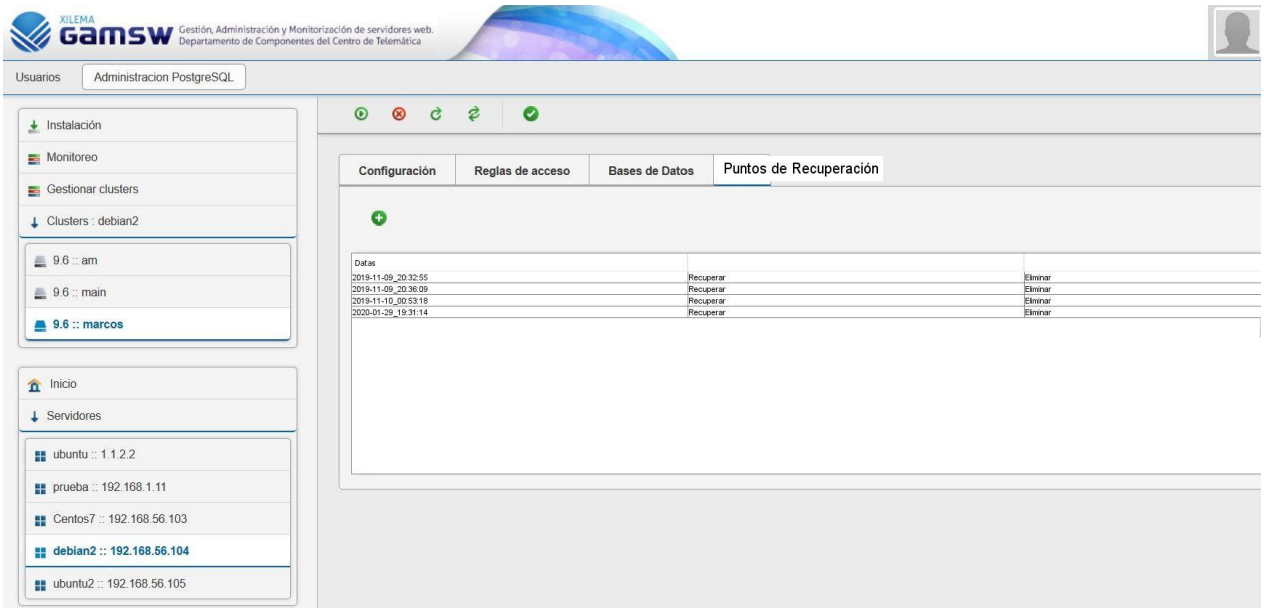


Número:15	**Nombre del requisito:** Listar Puntos de Recuperación en Debian.	
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1	
Prioridad: Alta	**Puntos Estimados:** 0.4	
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 día	

Descripción: Funcionalidad que permite al usuario listar los puntos de recuperación existentes en el servidor con sistema operativo Debian al que se está conectado. Se debe mostrar un listado de los puntos de recuperación existentes en el servidor con la opción de adicionar un punto de recuperación visible para el usuario, mostrar también para cada punto de recuperación la opción de recuperar y eliminar, En caso de no existir ningún punto de recuperación se muestra un listado vacío.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL en una versión mayor que la 9.4.

Prototipo de interfaz:



Número:22	**Nombre del requisito:** Eliminar punto de recuperación en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1

Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: • Bajo	**Tiempo Real:** 1 día

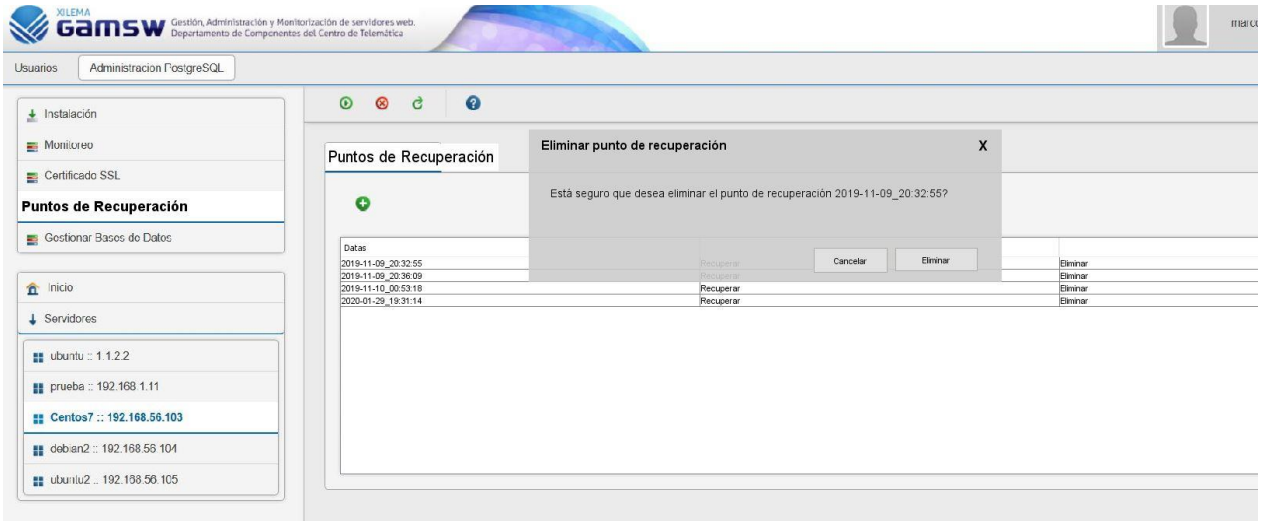
Descripción: Funcionalidad que permite el usuario eliminar un punto de recuperación en el servidor CentOS 7 al que se está gestionando, al pulsar el icono de eliminación, se debe mostrar una caja de confirmación de la operación, en esa caja se debe mostrar la opción de eliminar o cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación borrado con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina el punto de recuperación seleccionado.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

Prototipo de interfaz:



Número:23	**Nombre del requisito:** Eliminar punto de recuperación en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1

Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 días

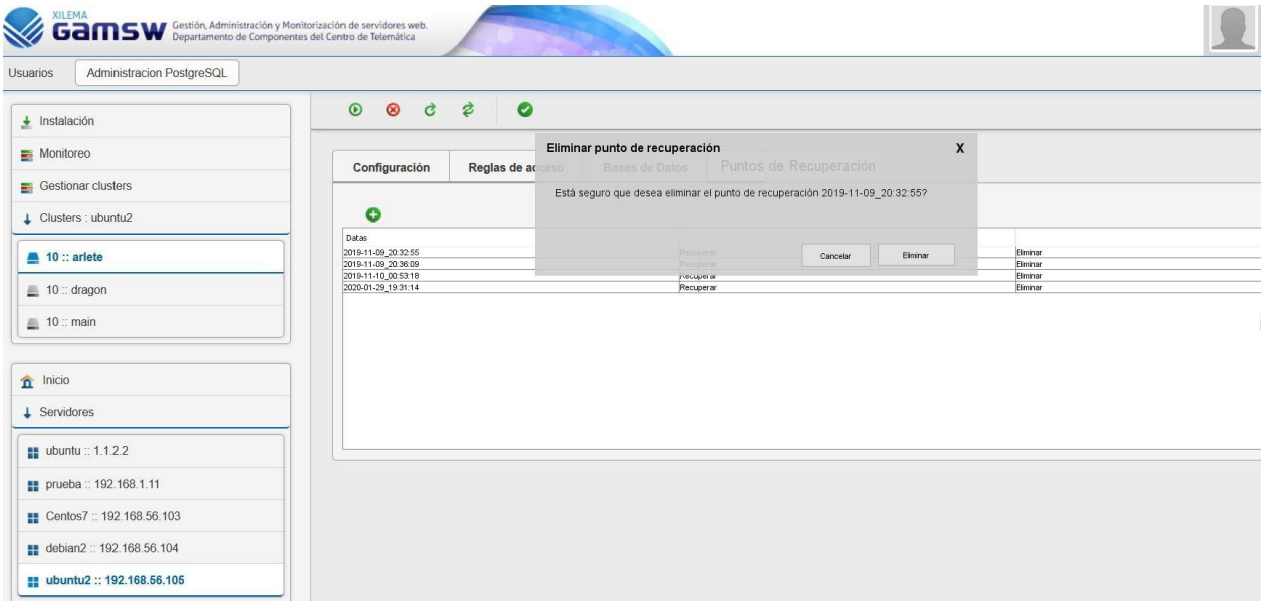
Descripción: Funcionalidad que permite eliminar un punto de recuperación en el servidor Ubuntu al que se está gestionando, al pulsar el icono de eliminación, se debe mostrar una caja de confirmación de la operación, en esa caja se debe mostrar la opción de eliminar o cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación borrado con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina el punto de recuperación seleccionado.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

Prototipo de interfaz:



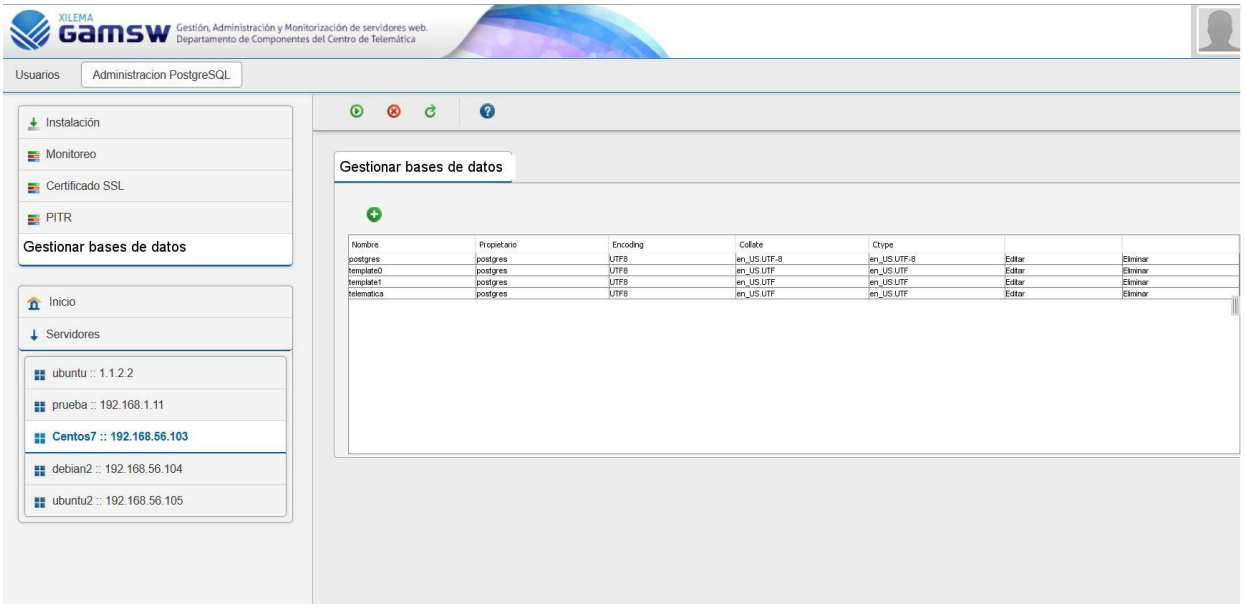
Número:1	**Nombre del requisito:** Listar Bases de datos relacionales en CentOS 7.

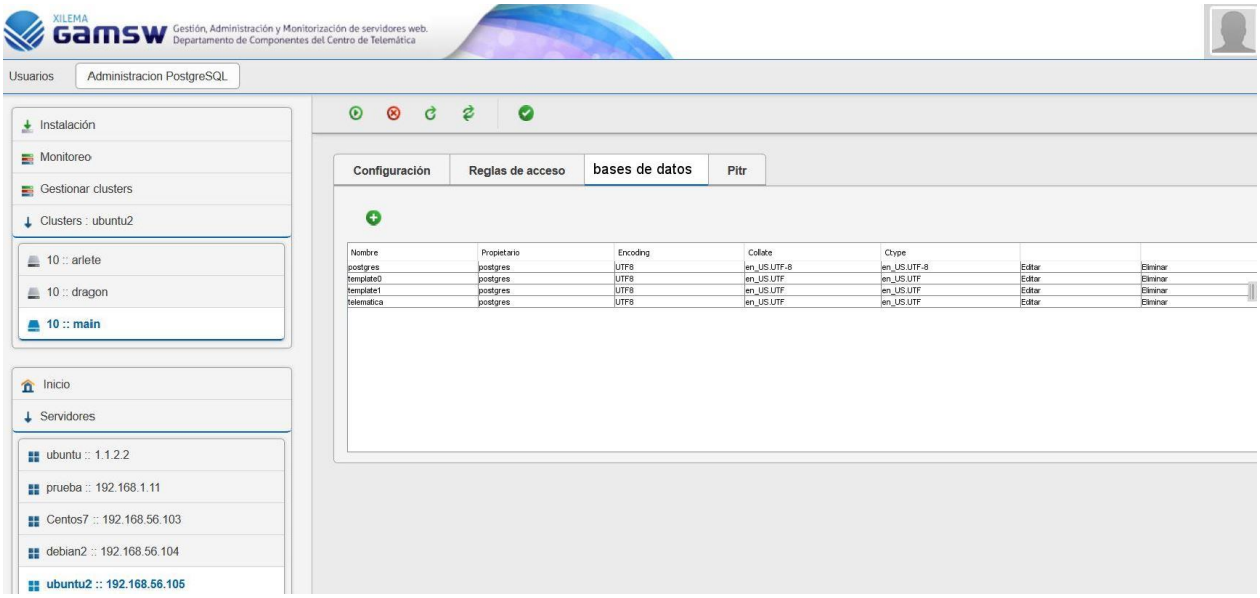
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1
Prioridad: Alta	**Puntos Estimados:** 0.4
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 días

Descripción: Funcionalidad que permite listar las bases de datos existentes en el servidor con sistema operativo CentOS 7 al que se está conectado, se debe mostrar un listado con las bases de datos existentes en el servidor, se debe dejar visible también las opciones de crear, renombrar y eliminar las bases de datos. En caso de no existir ninguna base de datos se muestra una tabla vacía.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL.

Prototipo de interfaz:



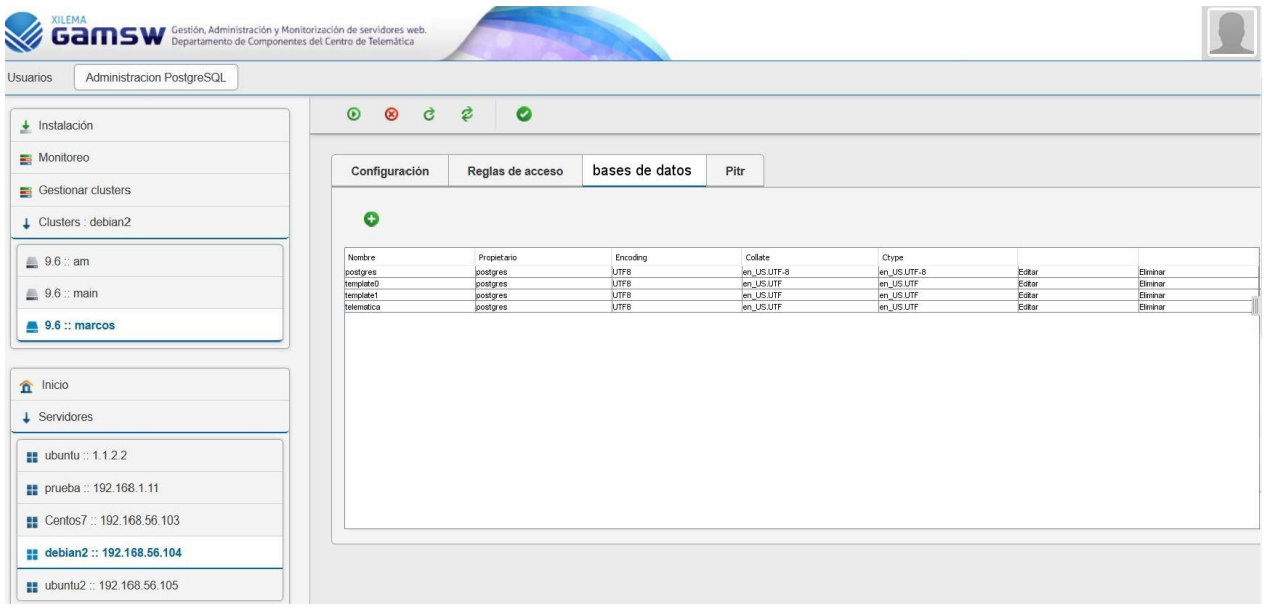
Número:2	**Nombre del requisito:** Listar Bases de datos relacionales en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1
Prioridad: Alta	**Puntos Estimados:** 0.4
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 días
Descripción: Funcionalidad que permite listar las bases de datos existentes en el servidor con sistema operativo Ubuntu al que se está conectado, se debe dejar visible también las opciones de crear, renombrar y eliminar las bases de datos. En caso de no existir ninguna base de datos se muestra una tabla vacía.	
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL.	
Prototipo de interfaz: 	

Número:3	**Nombre del requisito:** Listar Bases de datos relacionales en Debian.
Programador: Marcos do Rosário da Costa Fernandes	Iteración Asignada: 1
Prioridad: Alta	**Puntos Estimados:** 0.4
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 2 días

Descripción: Funcionalidad que permite listar las bases de datos existentes en el servidor con sistema operativo Debian al que se está conectado, se debe dejar visible también las opciones de crear, renombrar y eliminar las bases de datos. En caso de no existir ninguna base de datos se muestra una tabla vacía.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor debe tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 4	**Nombre del requisito:** Crear base de datos relacional en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.6
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite crear bases de datos relacionales en el servidor CentOS 7, al pulsar la opción “crear”, se debe enseñar al usuario un formulario donde se va a insertar el nombre de la base de datos a crear, después de la validación del nombre, el sistema debe proseguir con la creación de la mismo, el formulario debe tener la opción de cancelar la operación caso se desee. Se debe mostrar un mensaje de confirmación en cualquier caso ejemplo: “Base de datos creada con éxito!” o “Error en el servidor”).

Campos:

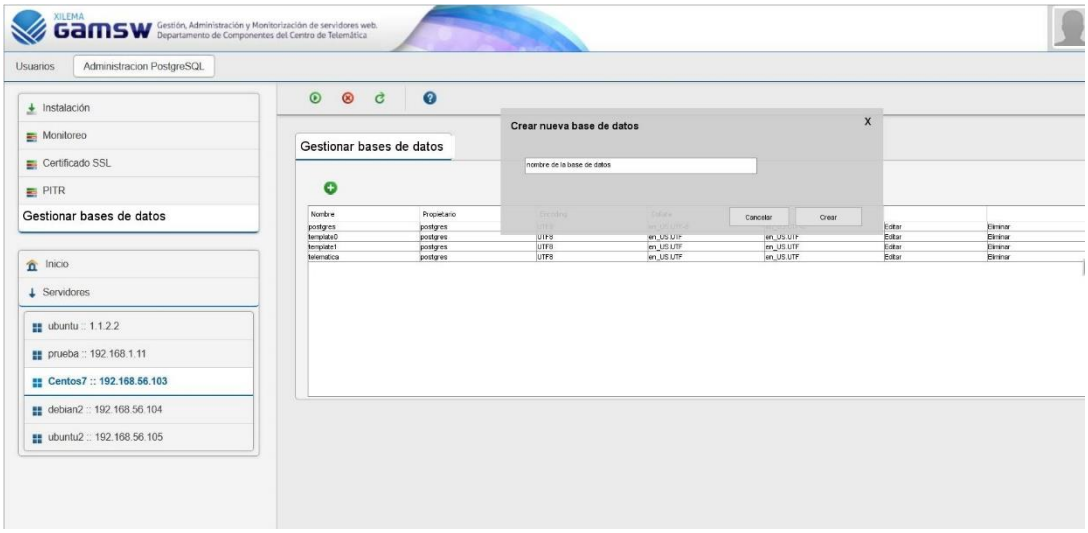
• Nombre: Campo de texto que permite nombrar la base de datos geoespacial (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Crear: Opción que prosigue con la creación de una base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 5	**Nombre del requisito:** Crear base de datos relacional en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.6
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite crear bases de datos relacionales en el servidor Ubuntu, al pulsar la opción “crear”, se debe enseñar al usuario un formulario donde se va a insertar el nombre de la base de datos a crear, después de la validación del nombre, el sistema debe proseguir con la creación de la mismo, el formulario debe tener la opción de cancelar la operación caso se desee. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos creada con éxito!” o “Error en el servidor”).

Campos:

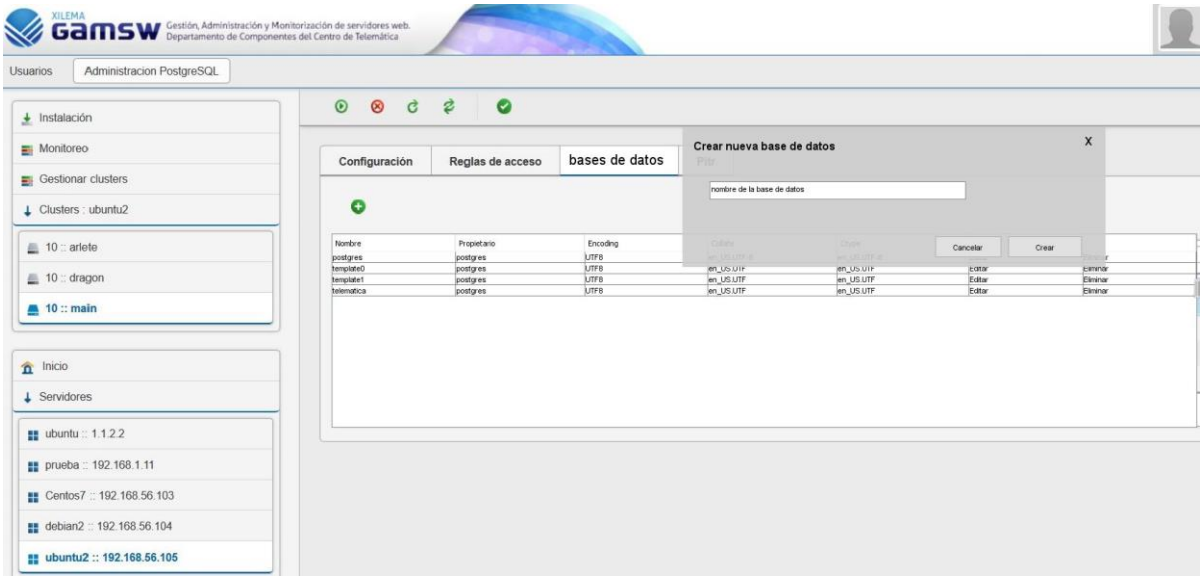
• Nombre: Campo de texto que permite nombrar la base de datos relacional (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Crear: Opción que prosigue con la creación de una base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 6	**Nombre del requisito:** Crear base de datos relacional en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.6
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite crear bases de datos relacionales en el servidor Debian, al pulsar la opción “crear”, se debe enseñar al usuario un formulario donde se va a insertar el nombre de la base de datos a crear, después de la validación del nombre, el sistema debe proseguir con la creación de la mismo, el formulario debe tener la opción de cancelar la operación caso se desee. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos creada con éxito!” o “Error en el servidor”).

Campos:

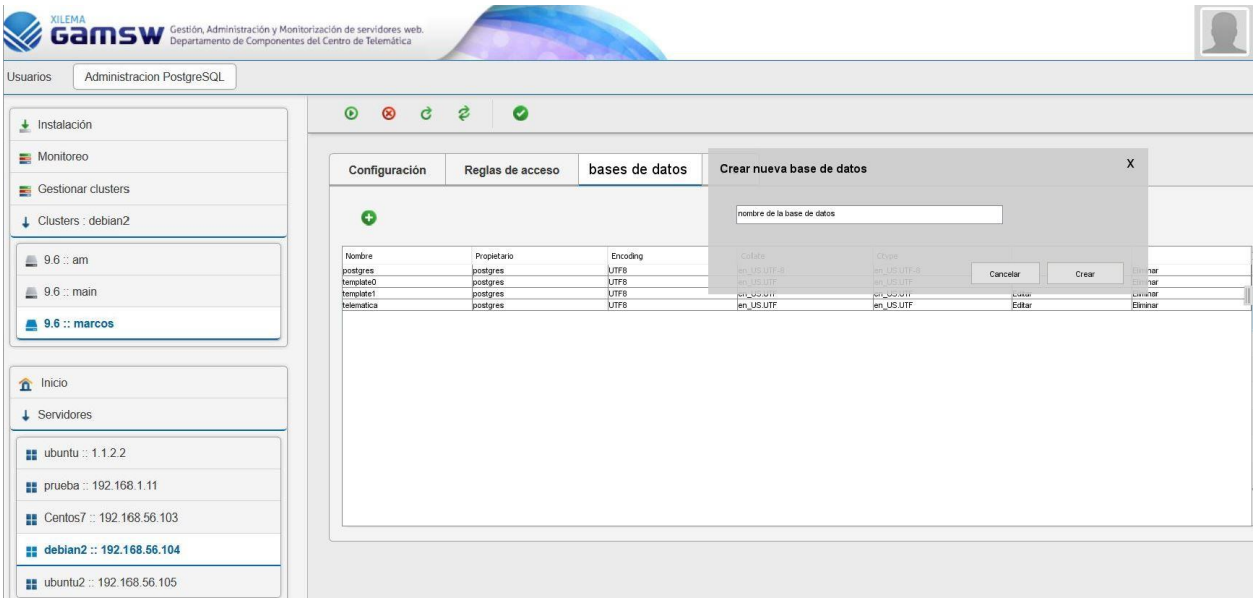
• Nombre: Campo de texto que permite nombrar la base de datos relacional (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Crear: Opción que prosigue con la creación de una base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 7	**Nombre del requisito:** Renombrar base de datos relacional en CentOS 7.	
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1	
Prioridad: Alta	**Puntos Estimados:** 0.6	
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días	
Descripción: Funcionalidad que permite renombrar bases de datos relacionales en el servidor CentOS 7, al pulsar la opción renombrar se debe enseñar el formulario donde se va a insertar el		

nuevo nombre de la base de datos, después de la validación del nombre, el sistema debe seguir con los cambios, el formulario debe tener visible la opción de cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos renombrada con éxito!” o “Error en el servidor”).

Campos:

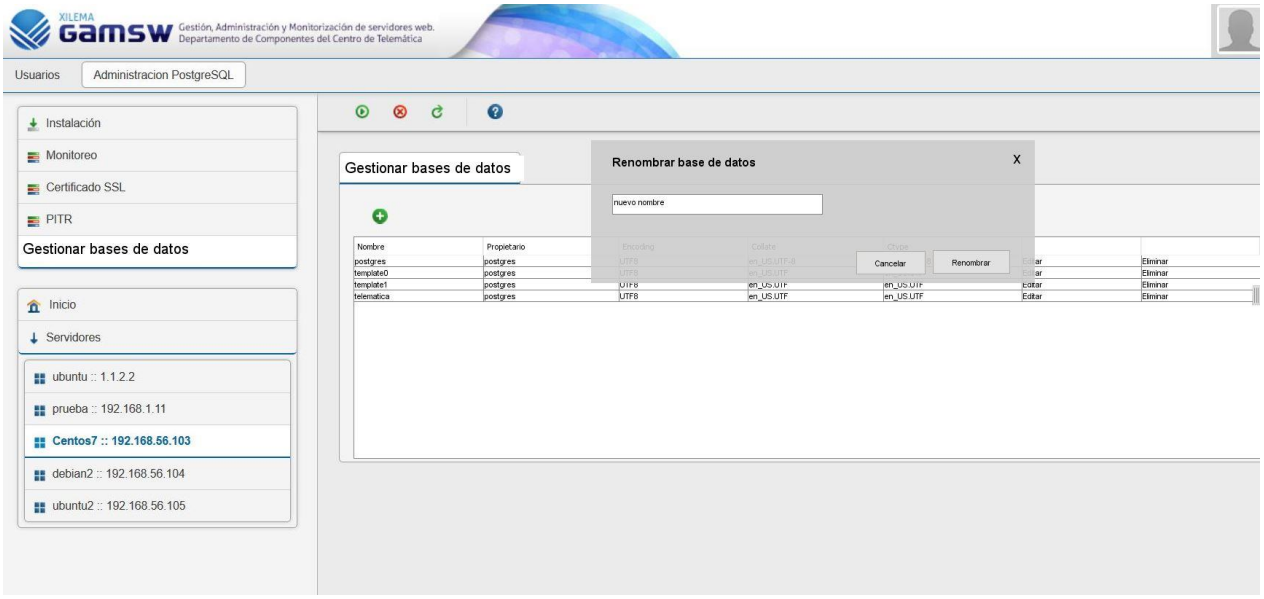
• Nombre: Campo de texto que permite nombrar la base de datos relacional (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Renombrar: Opción que prosigue con el cambio de nombre de la base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 8	**Nombre del requisito:** Renombrar base de datos relacional en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1

Prioridad: Alta	**Puntos Estimados:** 0.6
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite renombrar bases de datos relacionales en el servidor Ubuntu, al pulsar la opción renombrar se debe enseñar el formulario donde se va a insertar el nuevo nombre de la base de datos, después de la validación del nombre, el sistema debe seguir con los cambios, el formulario debe tener visible la opción de cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos renombrada con éxito!” o “Error en el servidor”).

Campos:

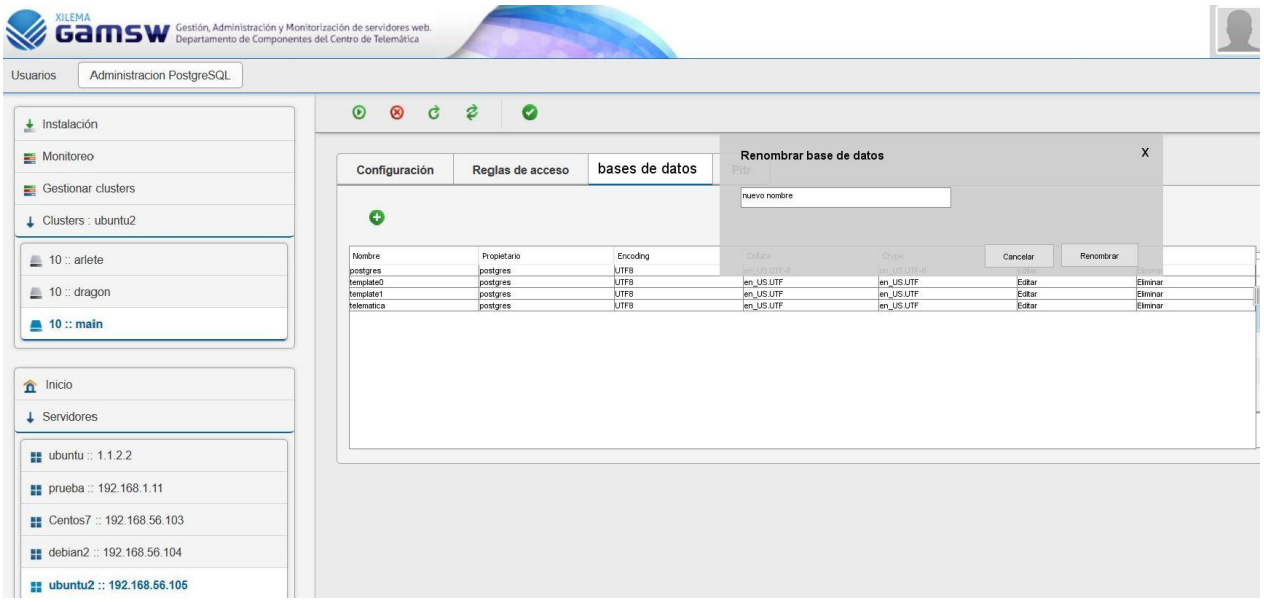
• Nombre: Campo de texto que permite nombrar la base de datos relacional (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Renombrar: Opción que prosigue con el cambio de nombre de la base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número: 9	**Nombre del requisito:** Renombrar base de datos relacional en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.6
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite renombrar bases de datos relacionales en el servidor Debian, al pulsar la opción renombrar se debe enseñar el formulario donde se va a insertar el nuevo nombre de la base de datos, después de la validación del nombre, el sistema debe seguir con los cambios, el formulario debe tener visible la opción de cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos renombrada con éxito!” o “Error en el servidor”).

Campos:

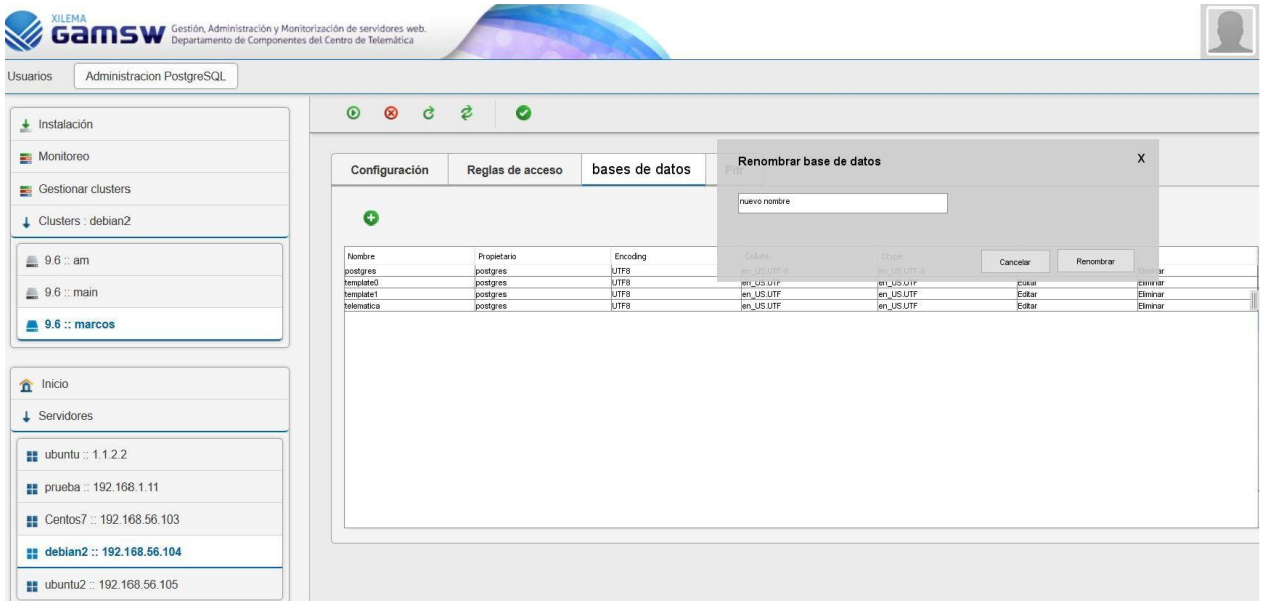
• Nombre: Campo de texto que permite nombrar la base de datos relacional (solo puede contener letras).

Botones:

- Cancelar: Opción que permite regresar al listado de bases de datos.
- Renombrar: Opción que prosigue con el cambio de nombre de la base de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL.

Prototipo de interfaz:



Número:10	**Nombre del requisito:** Eliminar base de datos relacional en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 días

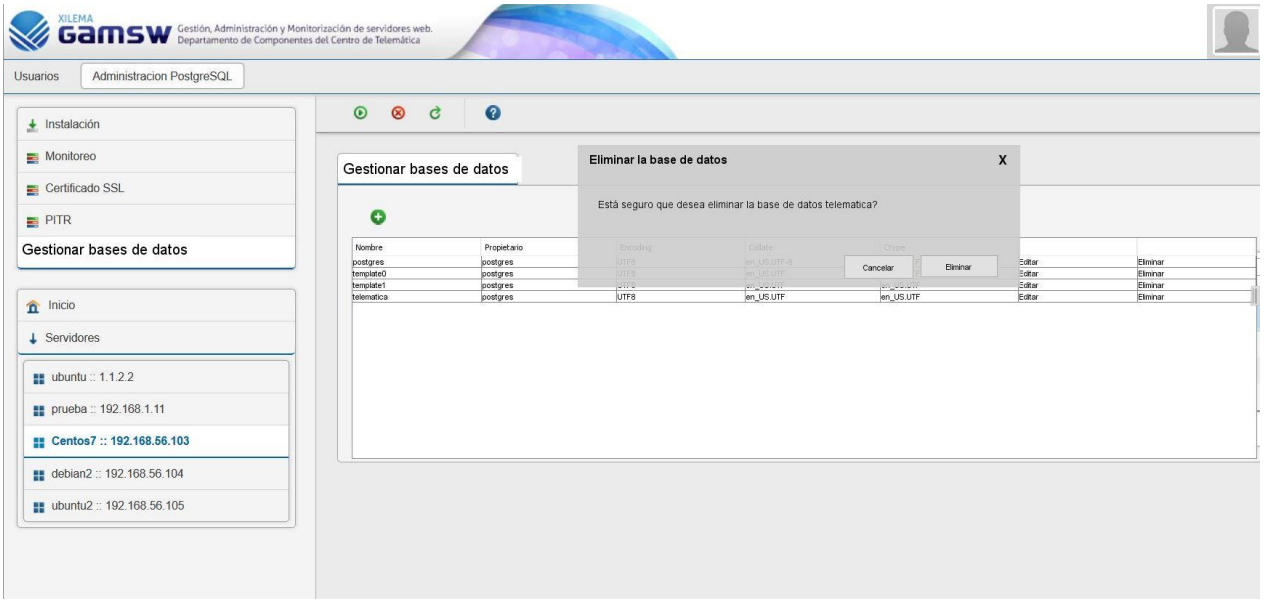
Descripción: Funcionalidad que permite eliminar una base de datos relacional en el servidor CentOS 7 al que se está gestionando, al pulsar la opción eliminar, se debe enseñar al usuario una caja de confirmación de la operación, en la misma se debe tener visible la opción de cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos borrada con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina la base de datos seleccionada.
- Cancelar: Opción que permite cancelar la operación y regresar al listado de bases de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber al menos una base de datos en el listado.

Prototipo de interfaz:



Número:11	**Nombre del requisito:** Eliminar base de datos relacional en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 días

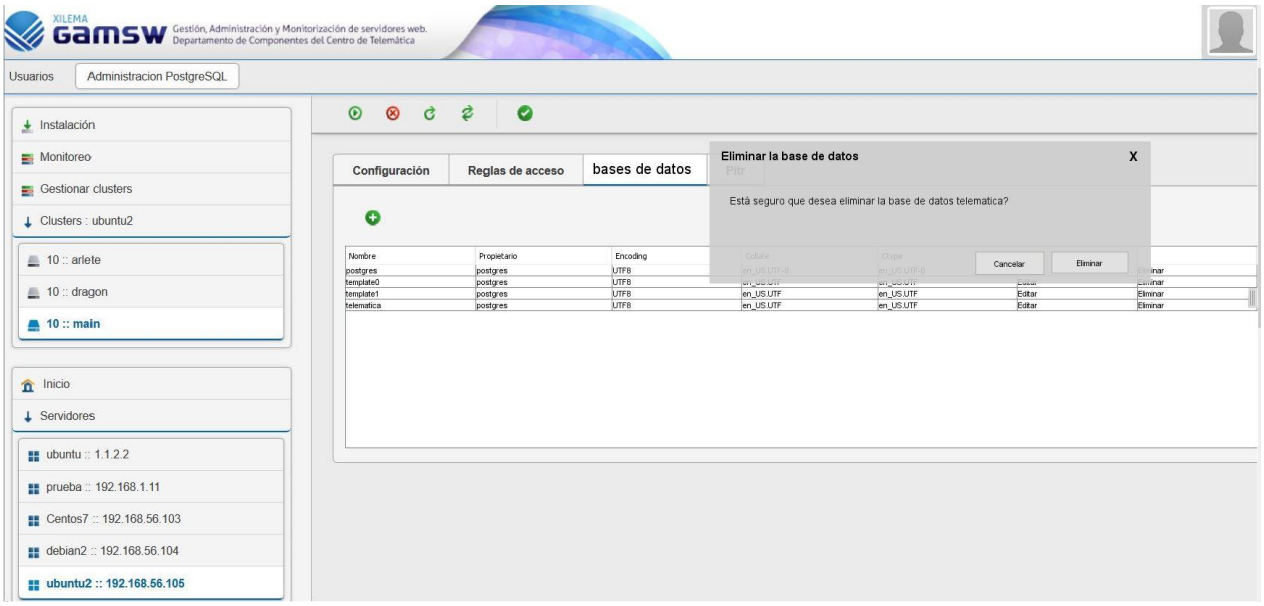
Descripción: Funcionalidad que permite eliminar una base de datos relacional en el servidor Ubuntu al que se está gestionando, al pulsar la opción eliminar, se debe enseñar al usuario una caja de confirmación de la operación, en la misma se debe tener visible la opción de cancelar la operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos borrada con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina la base de datos seleccionada.
- Cancelar: Opción que permite cancelar la operación y regresar al listado de bases de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber al menos una base de datos en el listado.

Prototipo de interfaz:



Número:12	**Nombre del requisito:** Eliminar base de datos relacional en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.2
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 1 día

Descripción: Funcionalidad que permite eliminar una base de datos relacional en el servidor Debian al que se está gestionando, al pulsar la opción eliminar, se debe enseñar al usuario una caja de confirmación de la operación, en la misma se debe tener visible la opción de cancelar la

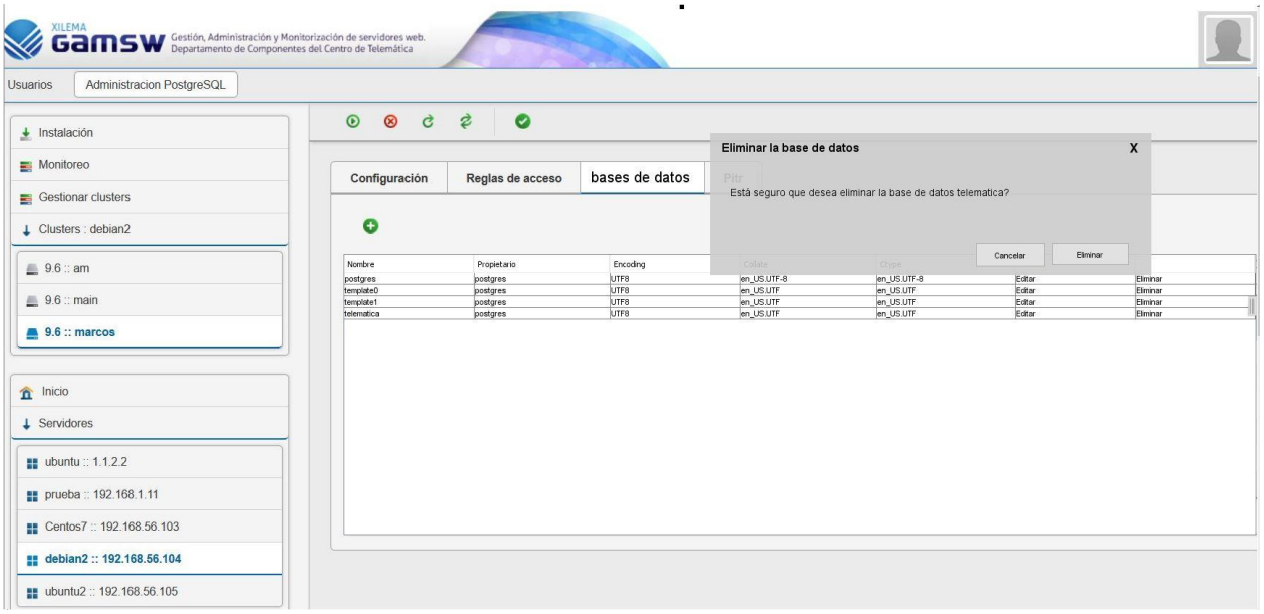
operación. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Base de datos borrada con éxito!” o “Error en el servidor”).

Botones:

- Eliminar: Opción que elimina la base de datos seleccionada.
- Cancelar: Opción que permite cancelar la operación y regresar al listado de bases de datos.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber al menos una base de datos en el listado.

Prototipo de interfaz:



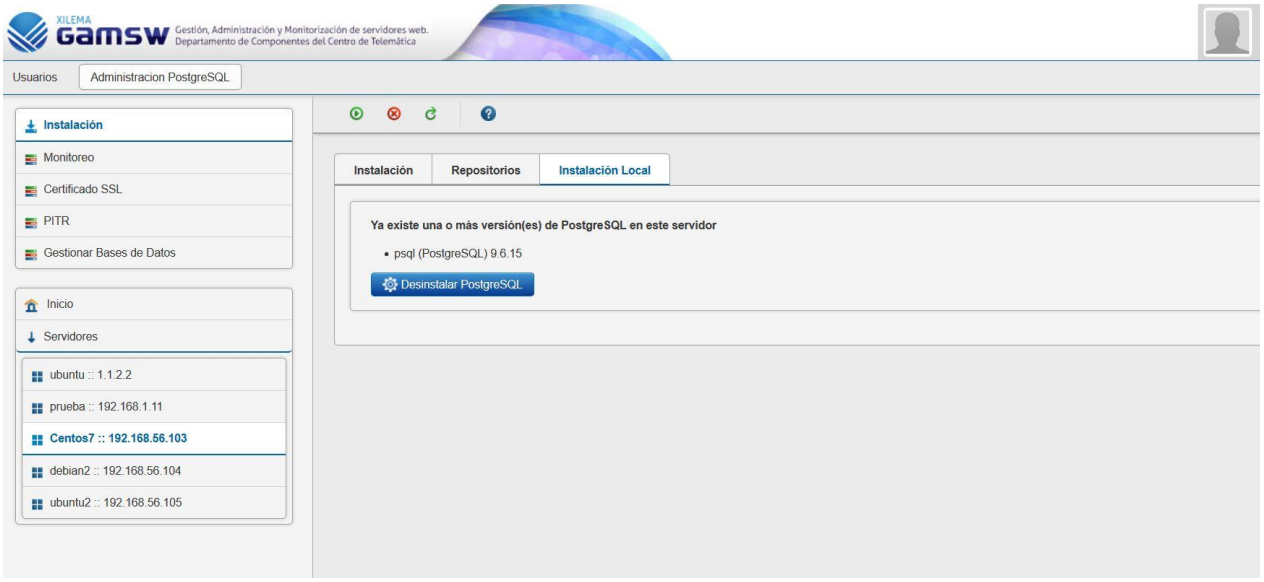
Número:25	**Nombre del requisito:** Instalación remota del PostgreSQL en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo:	**Tiempo Real:** 4 días

Bajo	

Descripción: Funcionalidad que permite instalar el servicio PostgreSQL en el servidor CentOS 7 al que se está gestionando, una vez pulsada la opción “instalar PostgreSQL”, el sistema debe seguir con la instalación del servicio PostgreSQL en la máquina que se gestiona. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Servicio instalado con éxito!” o “Error en la instalación”).

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.

Prototipo de interfaz:



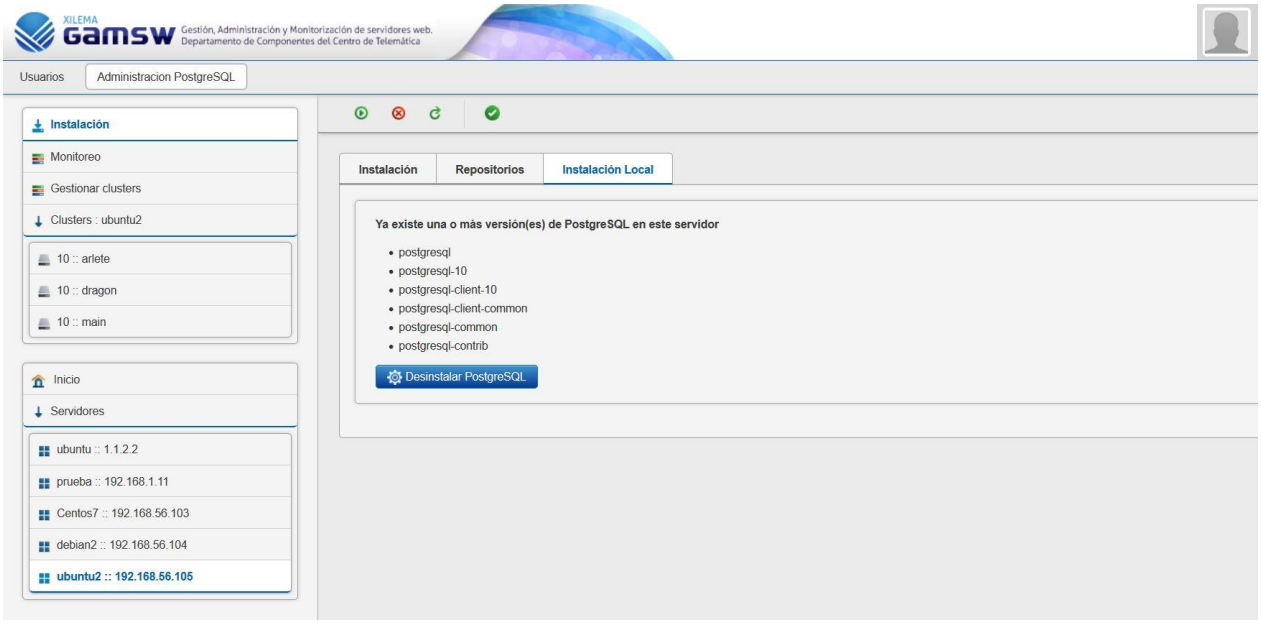
Número:26	**Nombre del requisito:** Instalación remota del PostgreSQL en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 5 días

Descripción: Funcionalidad que permite instalar el servicio PostgreSQL en el servidor Ubuntu al que se está gestionando, una vez pulsada la opción “instalar PostgreSQL”, el sistema debe seguir

con la instalación del servicio PostgreSQL en la máquina que se gestiona. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Servicio instalado con éxito!” o “Error en la instalación”).

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.

Prototipo de interfaz:



Número: 17	**Nombre del requisito:** Crear punto de recuperación en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 0.8
Riesgo en Desarrollo: • Bajo	**Tiempo Real:** 3 días

Descripción: Funcionalidad que permite el usuario crear un punto de recuperación en el servidor Ubuntu usando PITR, al dar pulsar la opción crear se debe abrir una caja de confirmación de la creación donde se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe

mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Punto de recuperación creado con éxito!” o “Error en el servidor”).

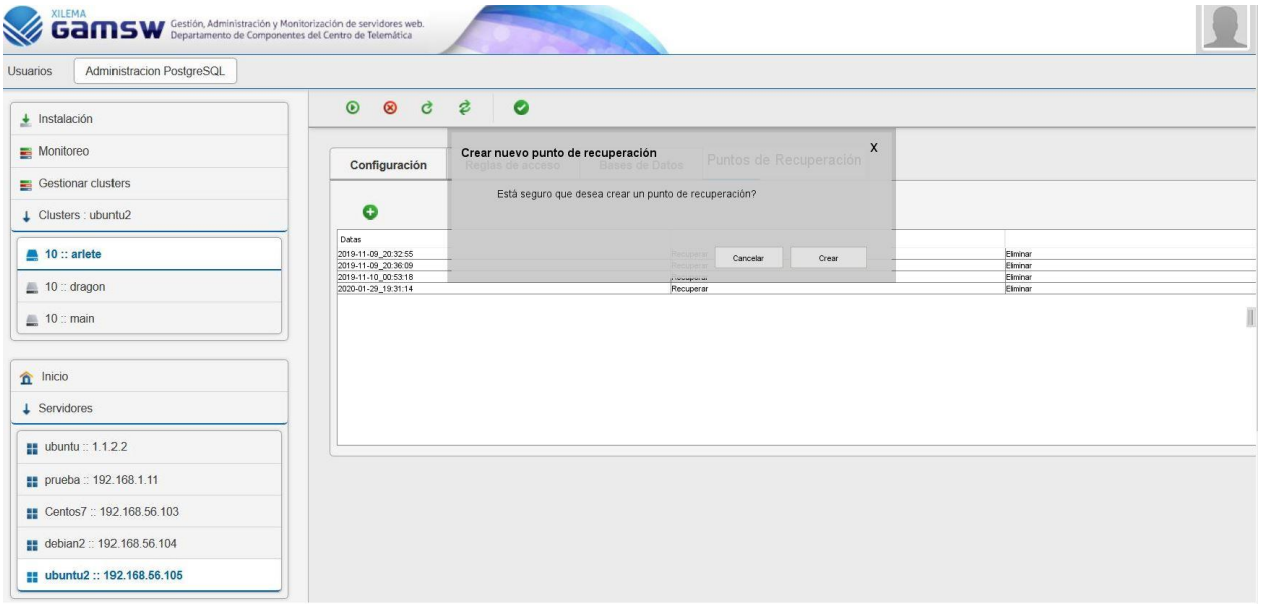
Botones:

Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Crear: Opción que prosigue con la creación de un punto de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL en una versión mayor que la 9.4.

Prototipo de interfaz:



Número:19	**Nombre del requisito:** Restaurar punto de recuperación en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo:	**Tiempo Real:** 4 días

Bajo	

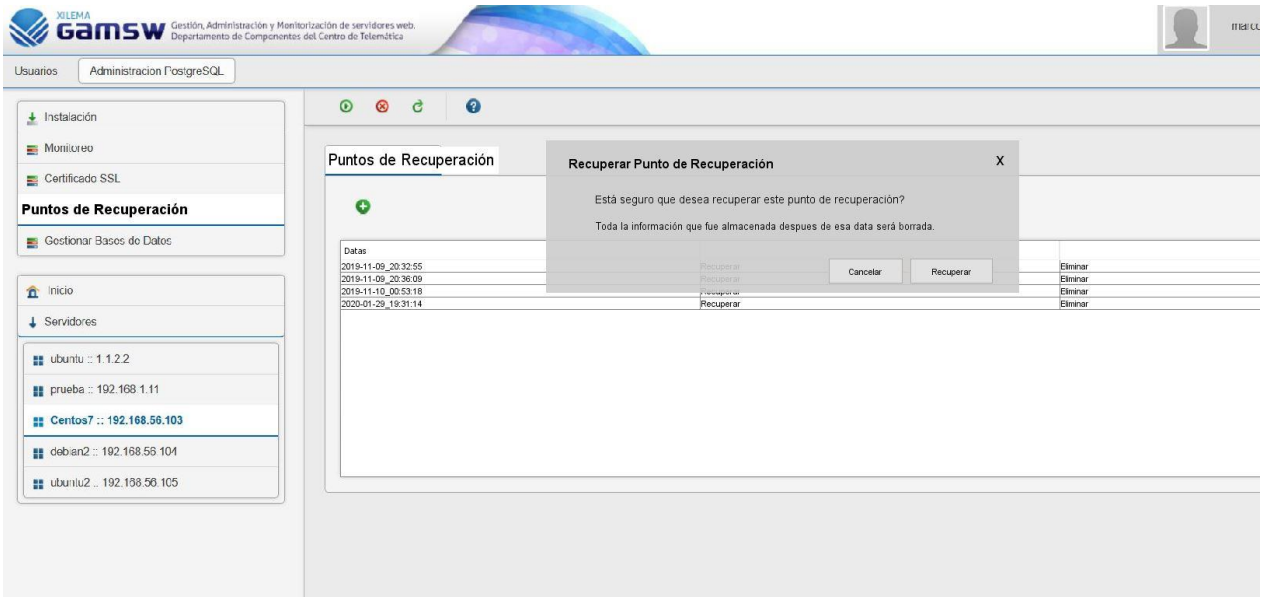
Descripción: Funcionalidad que permite el usuario restaurar la información guardada en un punto de recuperación en el servidor CentOS 7, al dar pulsar el icono de restauro se debe abrir una caja de confirmación del restauro de la información, se debe dar la posibilidad de seguir con la operación o cancelarla. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Recuperación realizada con éxito!” o “Error en el servidor”).

Botones:

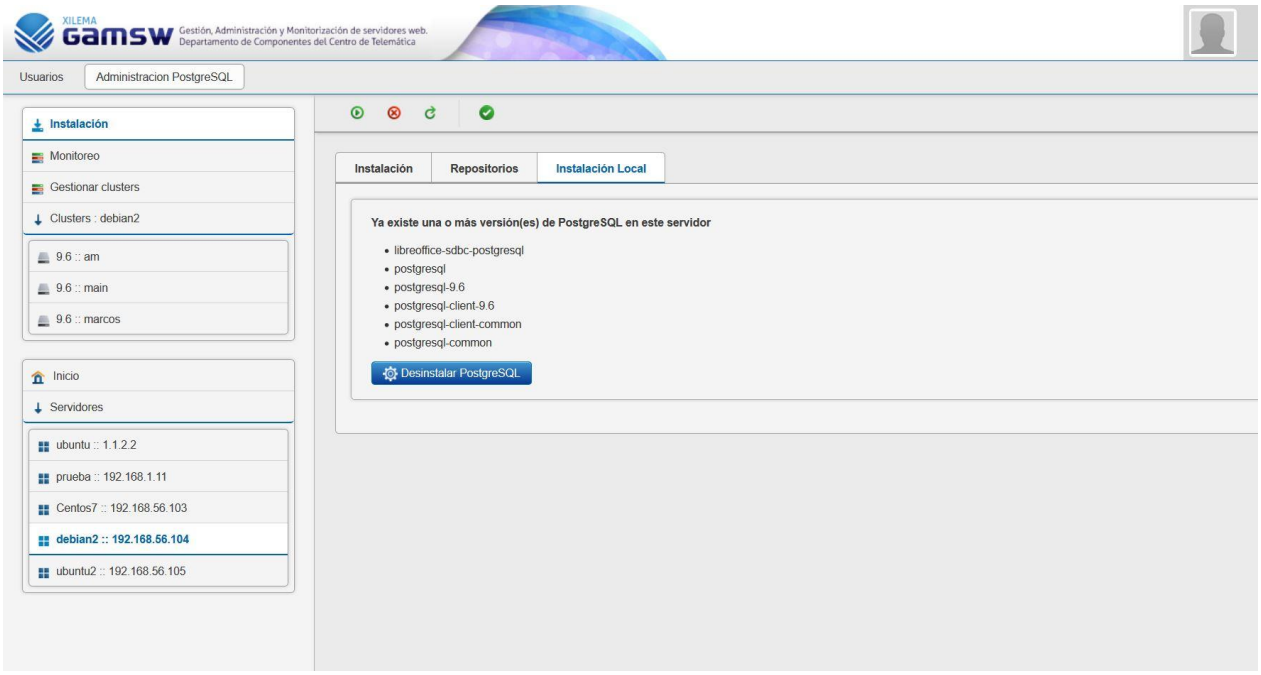
- Recuperar: Opción prosigue con la operación, recuperando todos los datos y configuraciones.
- Cancelar: Opción que permite regresar al listado de puntos de recuperación.

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor, y el servidor tiene que tener instalado el PostgreSQL, Debe haber por lo menos un punto de recuperación en el servidor.

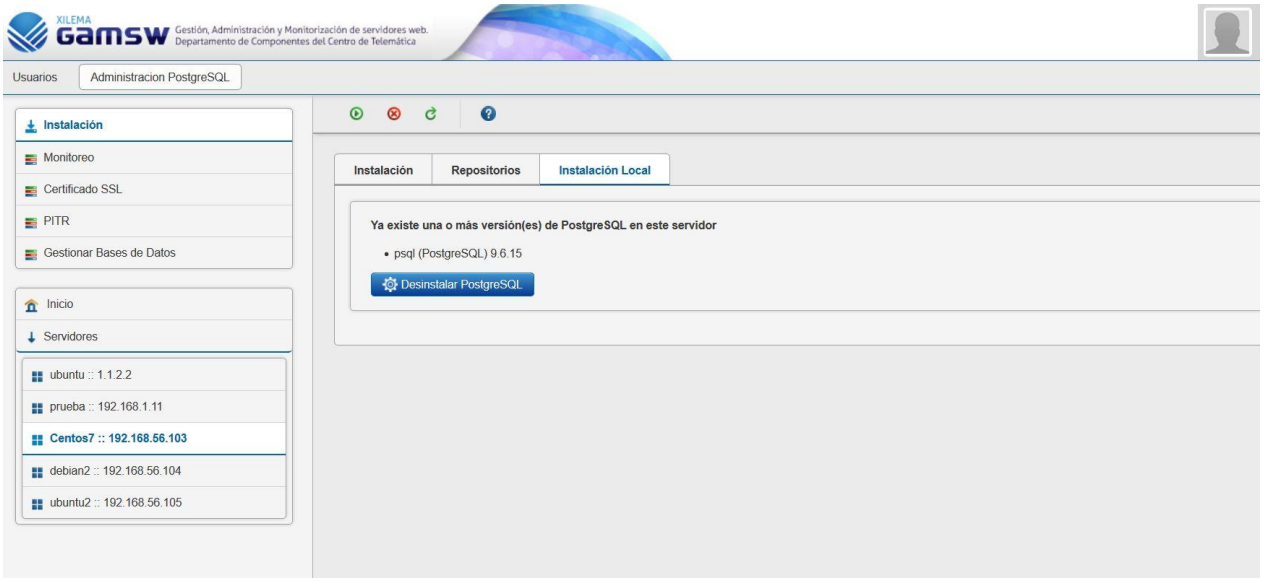
Prototipo de interfaz:



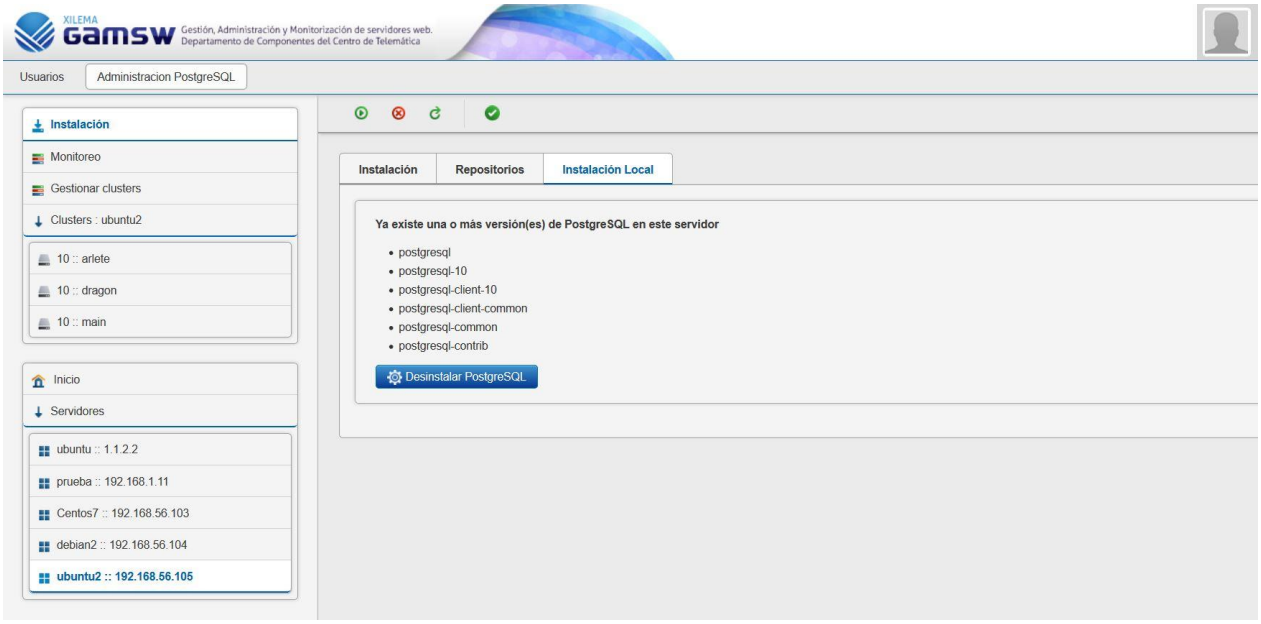
Número:27	**Nombre del requisito:** Instalación remota del PostgreSQL en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1

Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 días
Descripción: Funcionalidad que permite instalar el servicio PostgreSQL en el servidor Debian al que se está gestionando, una vez pulsada la opción “instalar PostgreSQL”, el sistema debe seguir con la instalación del servicio PostgreSQL en la máquina que se gestiona. Se debe mostrar un mensaje de confirmación en cualquier caso (ejemplo: “Servicio instalado con éxito!” o “Error en la instalación”).	
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.	
Prototipo de interfaz: 	

Número:28	**Nombre del requisito:** Instalación de diferentes versiones del PostgreSQL en CentOS 7.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1

Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 5 días
Descripción: Funcionalidad que permite Instalar diferentes versiones del servicio PostgreSQL en el servidor CentOS 7 al que se está gestionando, una vez detectadas diferentes versiones del servicio PostgreSQL, el sistema debe ser capaz de listarlas.	
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.	
Prototipo de interfaz: 	

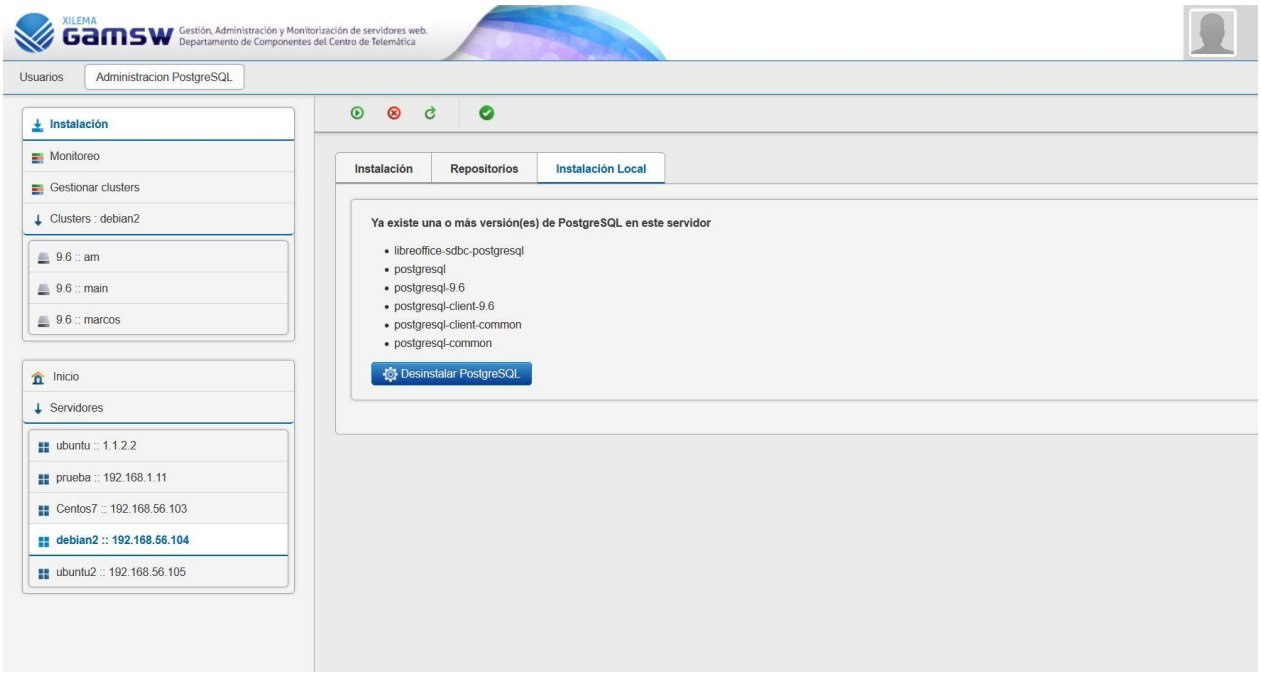
Número:29	**Nombre del requisito:** Instalación de diferentes versiones del PostgreSQL en Ubuntu.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 4 días

Descripción: Funcionalidad que permite Instalar diferentes versiones del servicio PostgreSQL en el servidor Ubuntu al que se está gestionando, una vez detectadas diferentes versiones del servicio PostgreSQL, el sistema debe ser capaz de listarlas.
Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.
Prototipo de interfaz: 

Número:30	**Nombre del requisito:** Instalación de diferentes versiones del PostgreSQL en Debian.
Programador: Marcos do Rosário da Costa Fernandes	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Riesgo en Desarrollo: Bajo	**Tiempo Real:** 5 días
Descripción: Funcionalidad que permite Instalar diferentes versiones del servicio PostgreSQL en el servidor Debian al que se está gestionando, una vez detectadas diferentes versiones del servicio PostgreSQL, el sistema debe ser capaz de listarlas.	

Observaciones: El usuario debe estar autenticado en el sistema, el sistema debe estar conectado al servidor.

Prototipo de interfaz:



Anexo 3 Guía para la realización de la entrevista

Ok ¿Quién usará las funcionalidades desarrolladas?

Ok ¿Hay otro origen para las funcionalidades que se necesitan?

Ok ¿Cuál sería una "buena" salida generada por una funcionalidad exitosa?

Ok ¿Qué problemas resolverían estas funcionalidades?

Ok ¿Puede mostrar (o describir) el ambiente de negocios en el que se usarían las funcionalidades?

Ok ¿Hay aspectos especiales del desempeño o restricciones que afecten el modo en el que se enfoque las funcionalidades a desarrollar?

Anexo 4 Diagramas de clases de diseño.

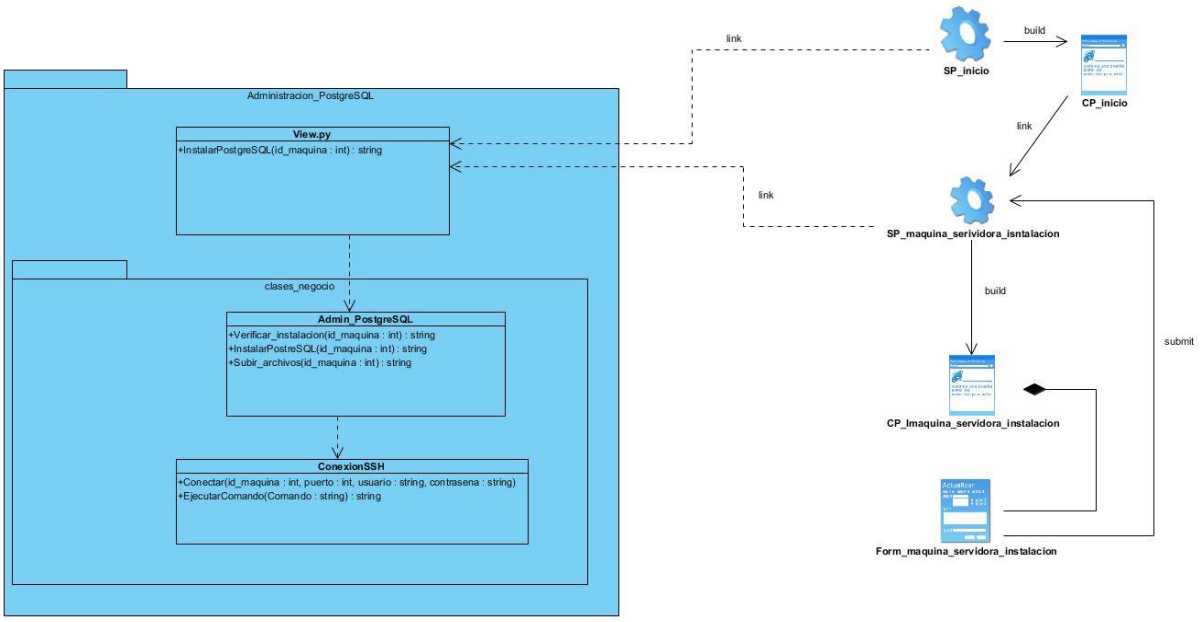


Diagrama de clase del paquete Instalación remota de servicio PostgreSQL

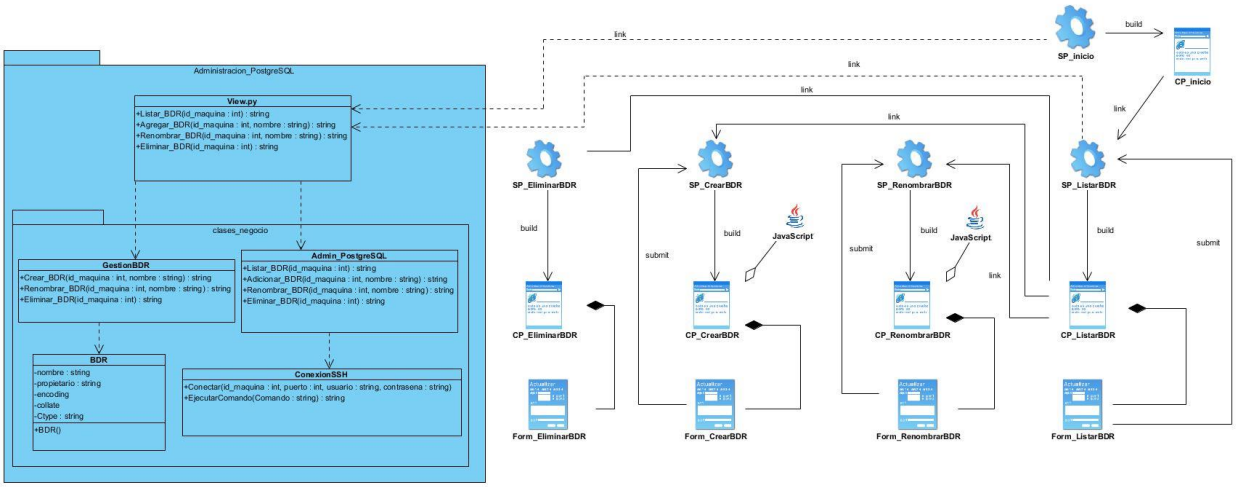


Diagrama de clase del paquete Gestionar Bases de datos relacionales

Anexo 5 Diagramas de secuencia.

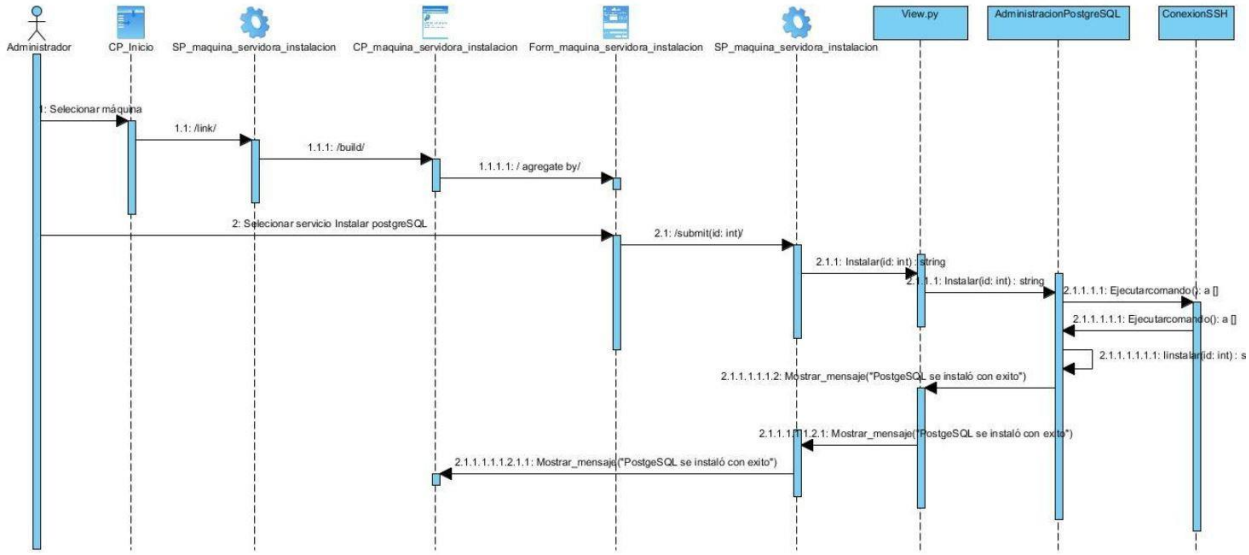


Diagrama de secuencia: HU 26 instalación remota del PostgreSQL en CentOS 7

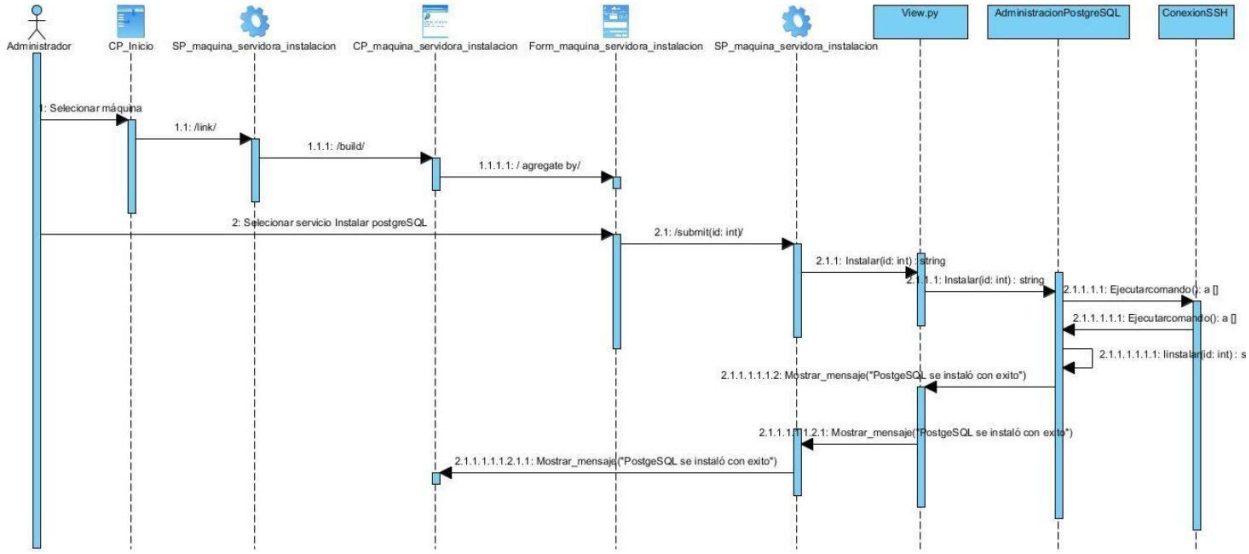


Diagrama de secuencia: HU 27 instalación remota del PostgreSQL en Ubuntu

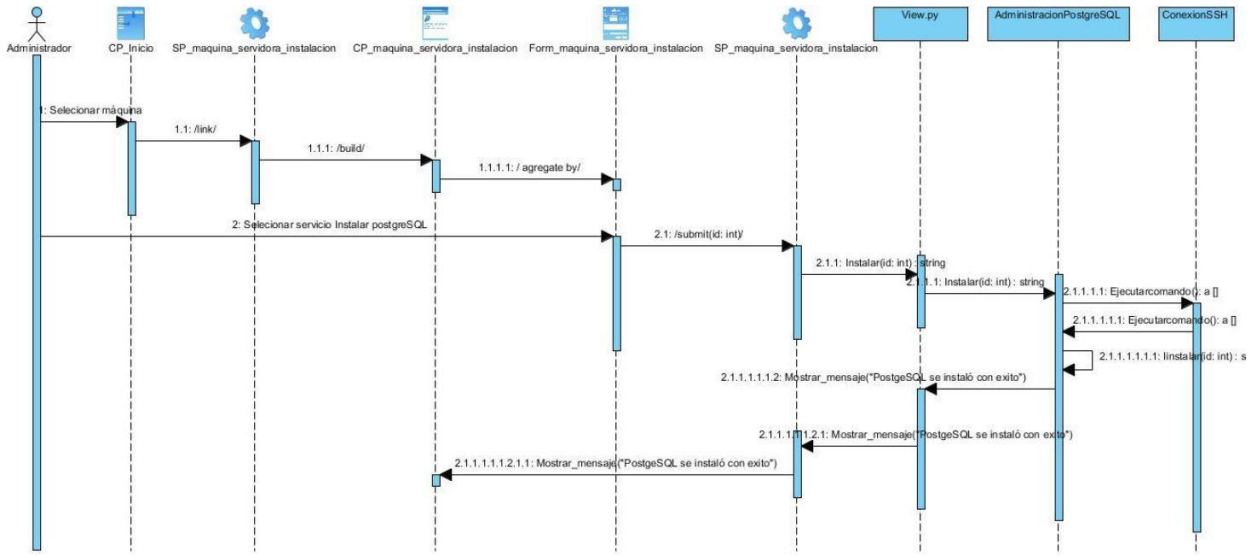


Diagrama de secuencia: HU 28 instalación remota del PostgreSQL en Debian

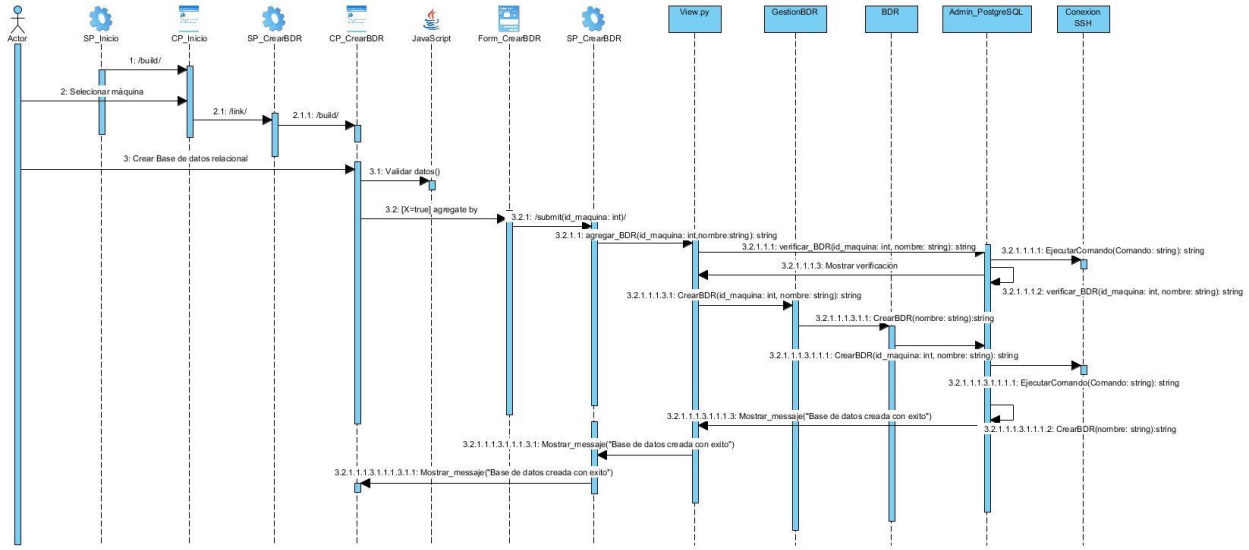


Diagrama de secuencia: HU 4 Crear base de datos relacional en CentOS 7

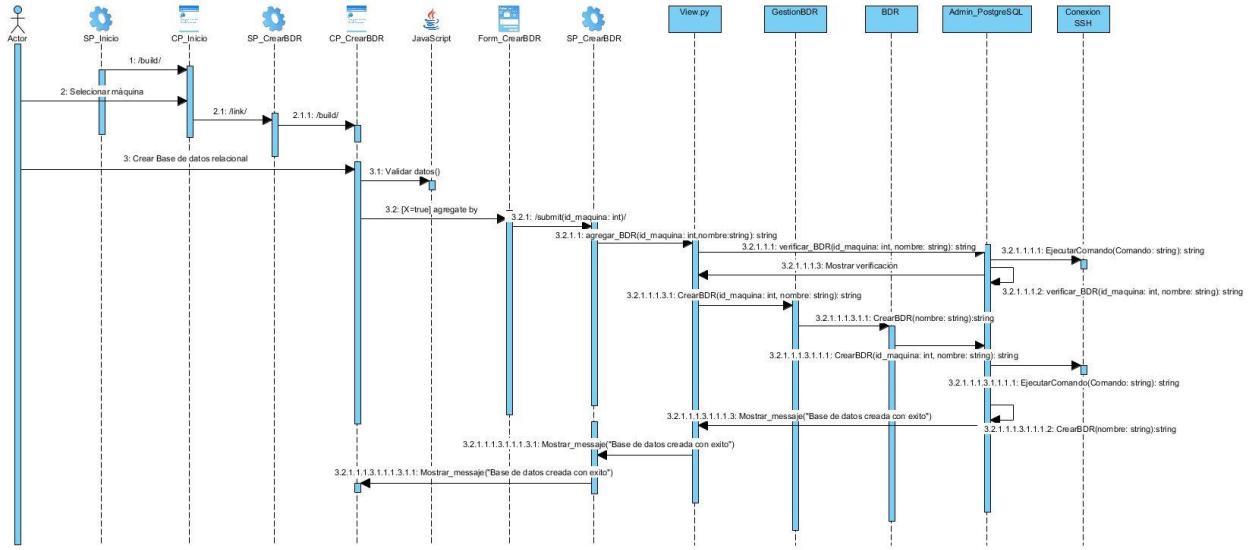


Diagrama de secuencia: HU 5 Crear base de datos relacional en Ubuntu

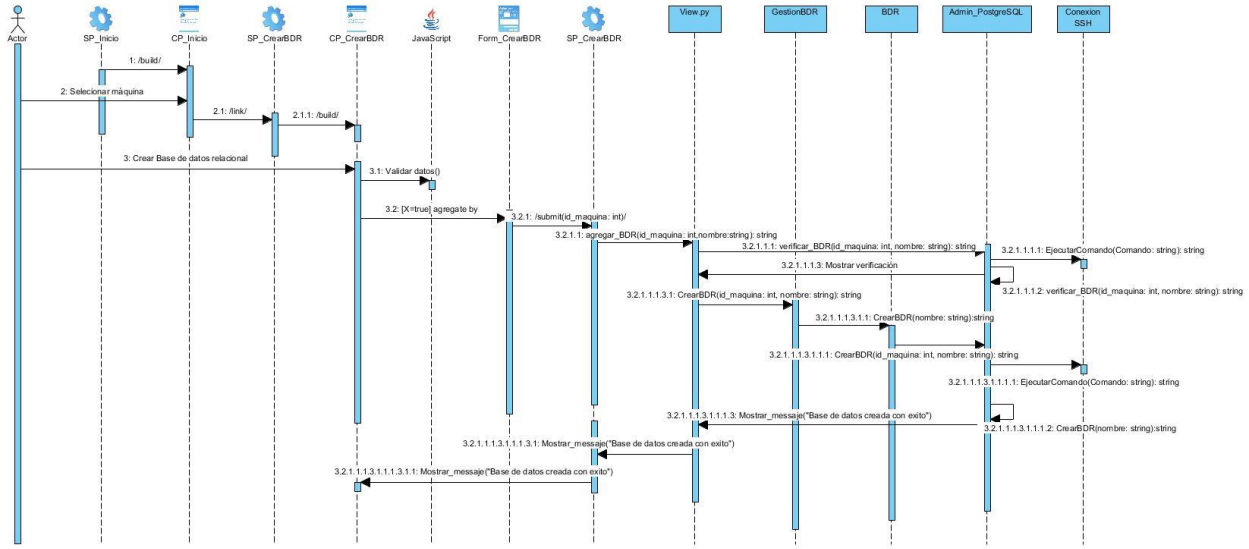


Diagrama de secuencia: HU 6 Crear base de datos relacional en Debian

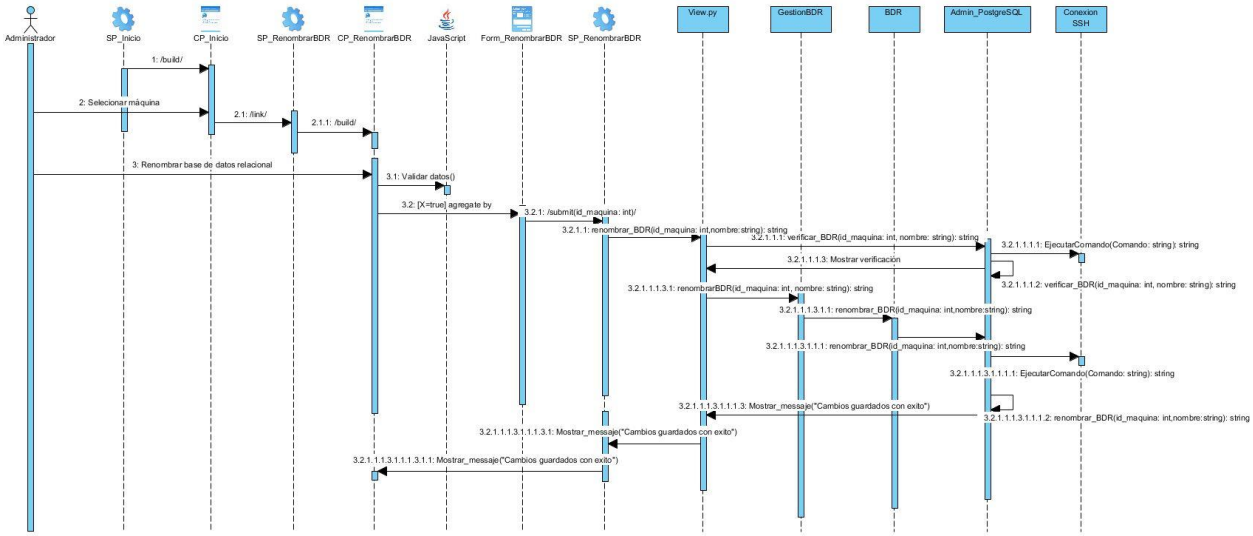


Diagrama de secuencia: HU 7 Renombrar base de datos relacional en CentOS 7

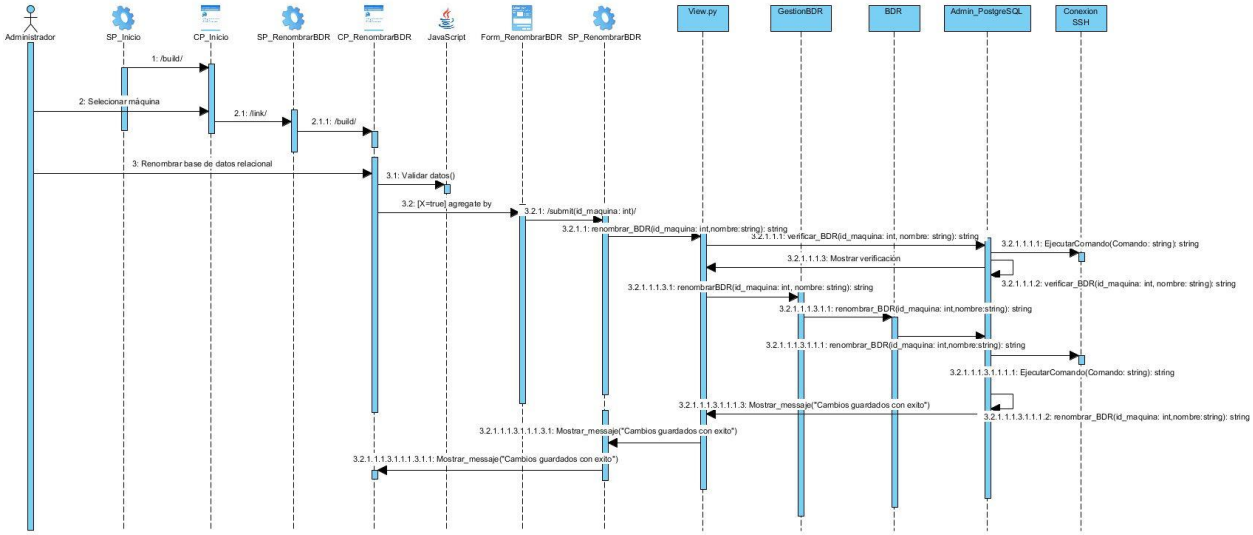


Diagrama de secuencia: HU 8 Renombrar base de datos relacional en Ubuntu

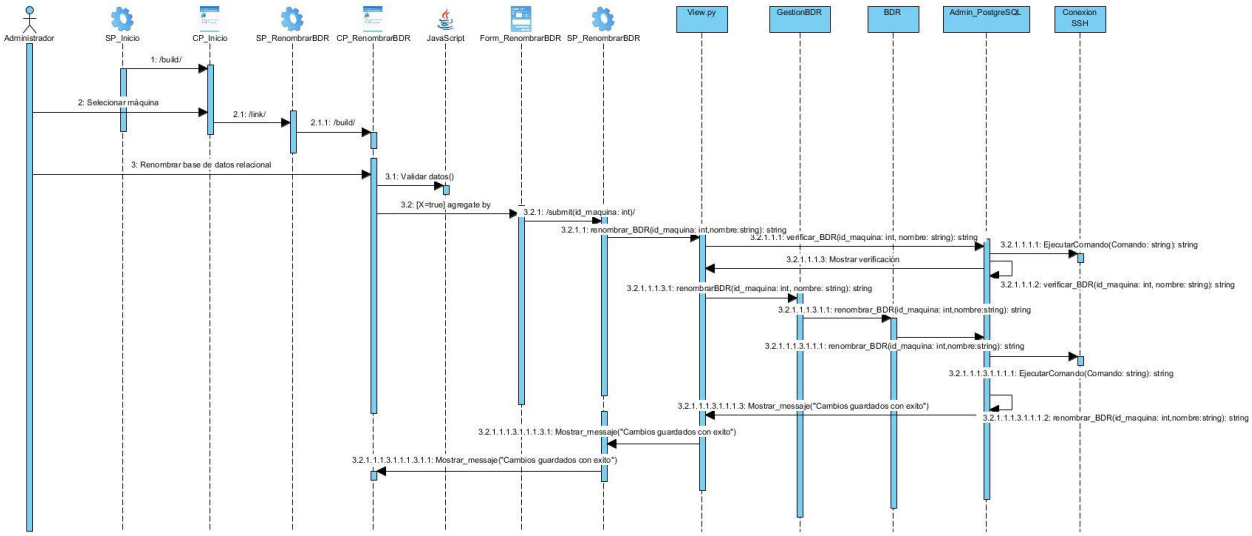


Diagrama de secuencia: HU 9 Renombrar base de datos relacional en Debian

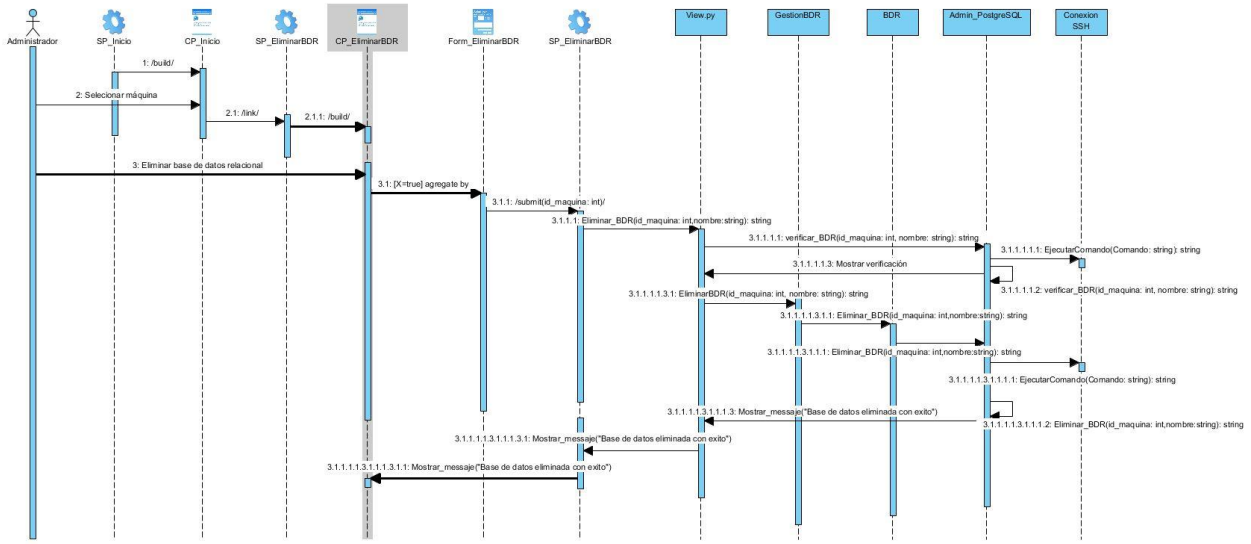


Diagrama de secuencia: HU 10 Eliminar base de datos relacional en CentOS 7

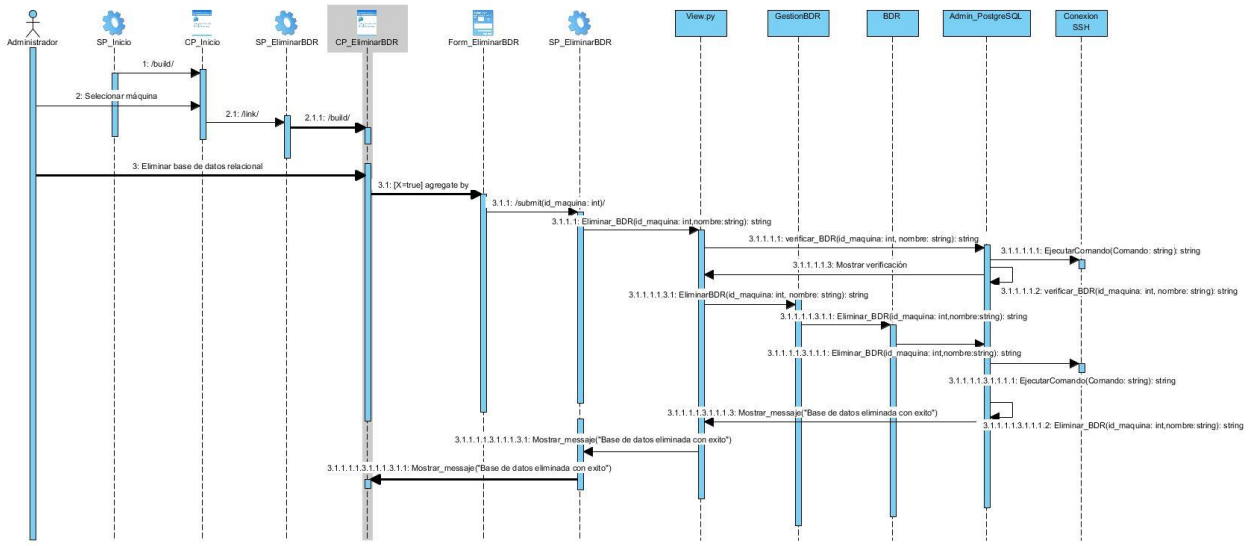


Diagrama de secuencia: HU 11 Eliminar base de datos relacional en Ubuntu

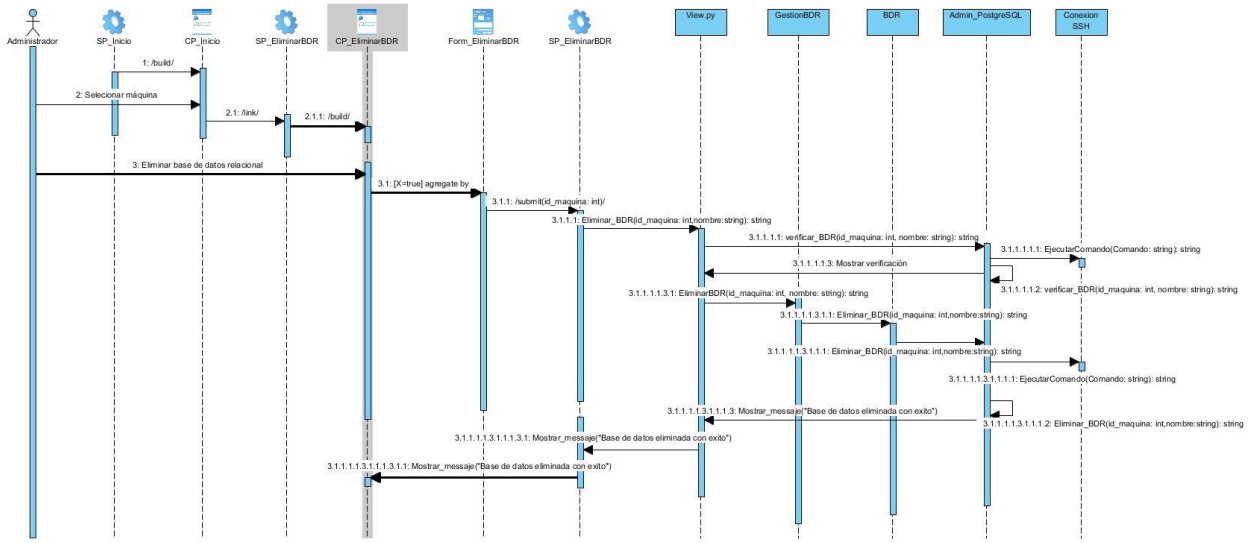


Diagrama de secuencia: HU 12 Eliminar base de datos relacional en Debian

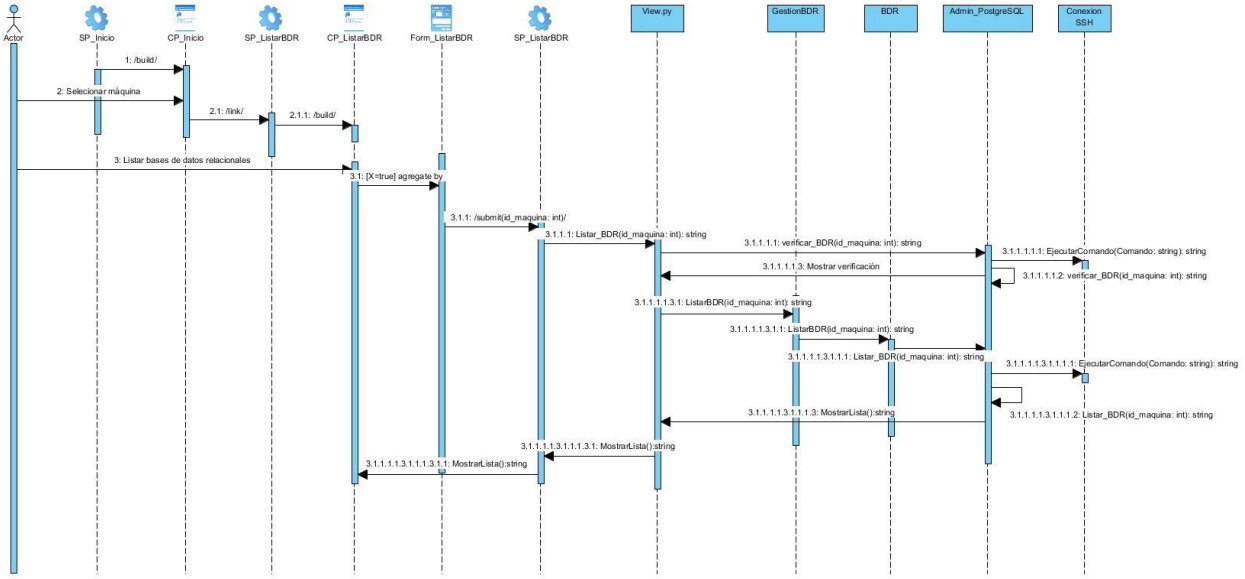


Diagrama de secuencia: HU 1 Listar bases de datos relacionales en CentOS 7

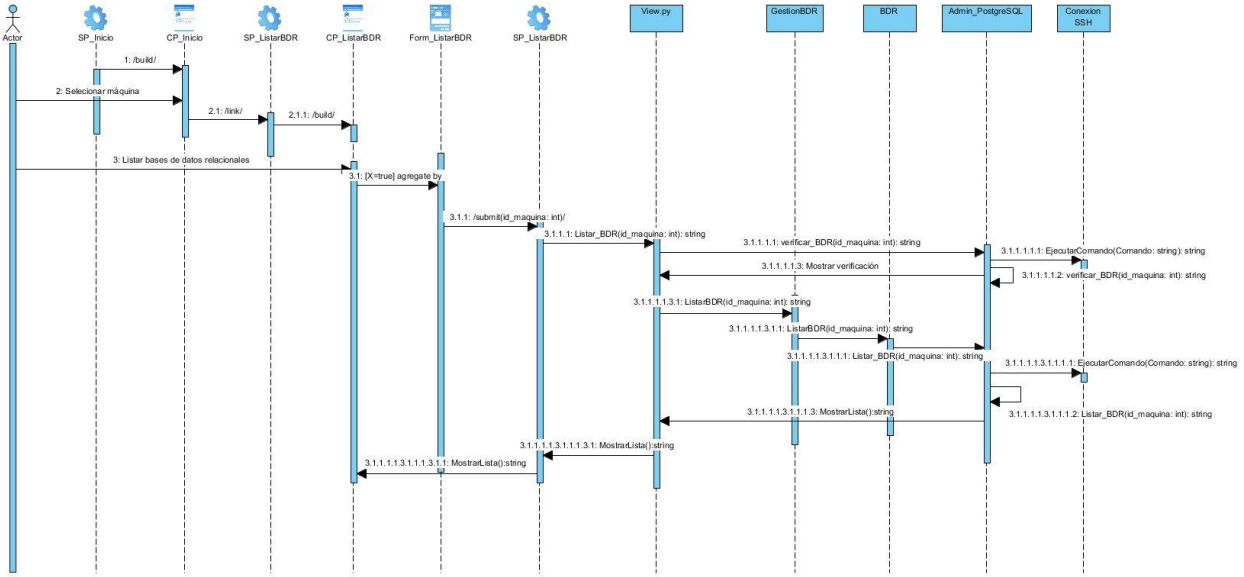


Diagrama de secuencia: HU 2 Listar bases de datos relacionales en Ubuntu

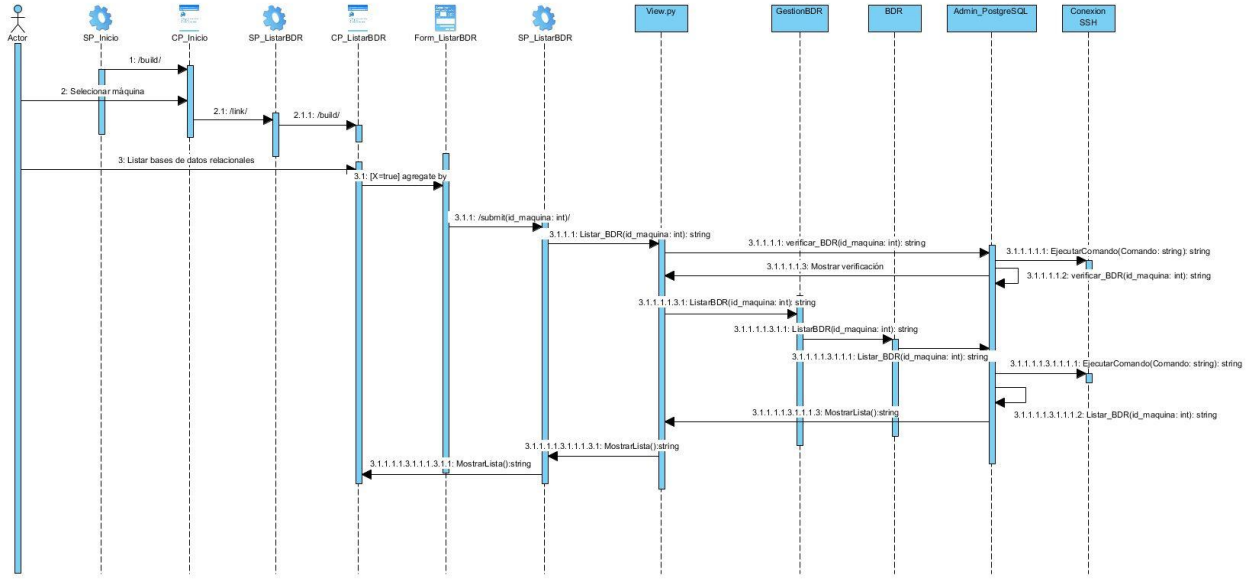


Diagrama de secuencia: HU 3 Listar bases de datos relacionales en Debian