Universidad de las Ciencias Informáticas

Implementación del componente réplica de base de datos para Akademos v2.0.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores:

Rosell Pupo Polanco

Yenier González Pérez

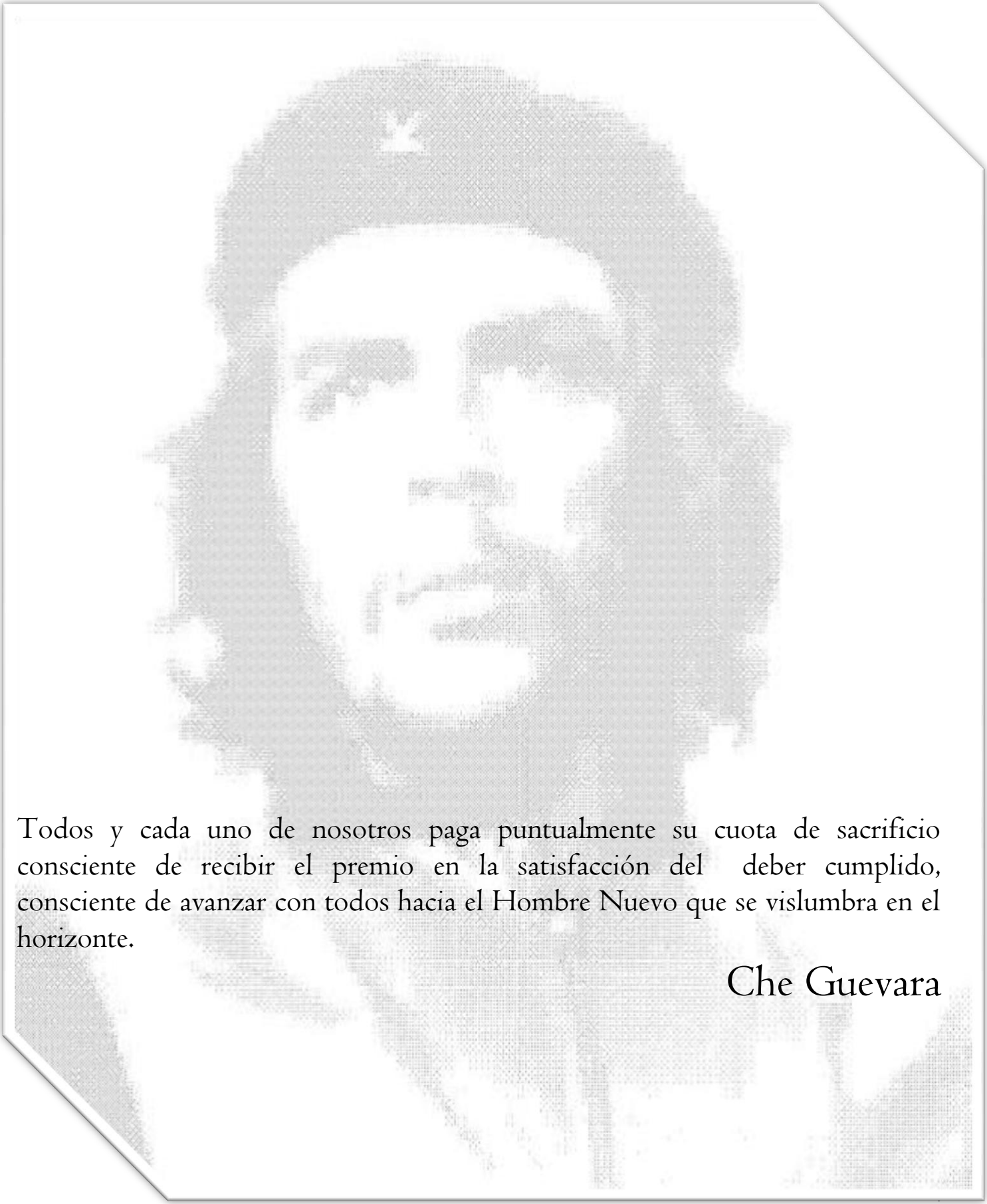
Tutores:

Ing. Joe Del Toro Domínguez

Ing. Dianly Santiler Álvarez

Ciudad de la Habana

Junio, 2009



Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, consciente de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Che Guevara

Agradecimientos

De Yenier:

…A nuestro eterno comandante por pensar en la creación de este gran proyecto, por sus ideas, por sus ejemplos, por darnos la oportunidad de ser participes de uno de sus más grandes sueños…

…A mis padres Marlenis y Víctor y mi padrastro Félix, a quienes les debo la vida y viviré eternamente agradecido por inculcarme siempre sus sabios consejos y experiencias…

…A mi hermano Yerit por darme la oportunidad de servirle de guía y ejemplo…

…A mi abuelita linda por toda la confianza que han depositado en mí…

…A mis tíos y tías por siempre estar presente cuando los necesitaba…

…A Alianne por demostrar ser mi amiga y ayudarme tanto en toda mi carrera y cocinarme cuando me quedaba sin almorzar por estar de fiesta…

…A Rosell, mi amigo y compañero de tesis, por haber compartido la autoría de la tesis conmigo...

…A todos mis amigos que vienen desde primer año soportándome y formando fiesta más conocido como la PMM que aunque quedamos poco somos insuperables…

…A mis amigos Cesar de Dios, Mariano Reingart por estar presente en las buenas y en las malas…

… A todos aquellos profesores que de una forma u otra me enseñaron y me ayudaron a llegar a ser un ingeniero…

…A las personas que de una forma u otras aportaron su granito de arena para que lograra hacer este sueño realidad y nunca dejaron de confiar en mí...

…A Norma por siempre brindarme su amor y apoyo cuando más lo necesitaba…

…A el piquete del doble que siempre me ayudaron tanto en la Tesis como para colarme en la cola del doble…

…A el piquete de chicas que tengo en el jabber que siempre dieron un si cuanto le escribía para alguna fiesta…

…A la UCI por hacerme realidad mi mayor sueño que es el de hacerme ingeniero…

De Rosell:

… A mi padre, por ser el ejemplo que siempre he seguido…

… A mi madre, por su amor y el sacrificio que ha dado para conseguir mis sueños…

… A los dos, por ser mis padres…

… A mis hermanas, por siempre estar pendientes de mí y cuidarme tanto…

… A mi sobrina y sobrino por hacerme despejar los fines de semana…

… A mi gran amiga Yadira, por su incondicional ayuda y apoyo… ahhh y por sus dulces…

A mi amiga Dunia, por todos los momentos lindos que hemos compartido…

… A ti Liliana, por tu sencillez, por tu amor y tu paciencia…

… A Yenier, por haber compartido este gran momento conmigo…

A Zuleira, por su amistad…

… A todos mis amigos y amigas del grupo…

… A todas las personas que me han brindado su ayuda…

… A la UCI, por recibirme en sus brazos…

Dedicatoria

De Yenier:

A mi mamá, por ser el mejor regalo que me ha dado la vida, mi mayor alegría, mis ojos, mi gran virtud, eres la mayor muestra de amor y bondad que dios ha guardado para mí. Tú lo que más amo.
A mi hermano Yerit por ser mi mano derecha en todo momento y por brindarme en todo momento su cariño y amor.
A la memoria de mi padre Víctor que me hubiese gustado mucho darle el placer de verme hecho ingeniero y donde quiera que esté sabe que en estos últimos tiempos he pensado mucho en él. A Dios por darme la fé.
A mi abuela Juana que ha hecho de mí un hombre de bien, que ha vivido para mí y daría con los ojos cerrados una vida de felicidad por evitarme un instante de sufrimiento. Siempre los llevo en mi corazón.

De Rosell:

Todo el esfuerzo y las horas de empeño para lograr realizar este trabajo quiero dedicarlos a las personas que han sido y son la razón de mi ser, y que adoro desde lo más profundo de mi corazón… a mi mamá Elvis y mi papá Rosell, a mis hermanas Beatriz y Vismarys y mis sobrinos Jean Carlos y María Carla…

A Kirenia y Lizandra por ser mis amigas en las buenas y en las malas…

A Maribel Pupo, Yailén Cabrera, Karel Romero, Yunier (Lucaz), Yurima, Lizandra, Leonor y todos los otros compañeros del 10 y 11 grado…

A Lizandra Rodríguez, por el inmenso cariño que nos une…

A mis amigas y amigos por todos los momentos que he compartido con cada uno de ellos.

Resumen

La Universidad de las Ciencias Informáticas (UCI), creada en el 2002 como programa de la Batalla de Ideas, debe llevar el control de todos los estudiantes y profesores residentes o no en la misma, además de estos se encarga de controlar la información de los estudiantes procedentes de las Facultades Regionales (FR) de Artemisa, de Ciego y de Manzanillo. Controlar todos estos datos y manipularlos es un proceso muy complicado que requiere de una buena gestión de la información entre las partes involucradas y una gran responsabilidad en la misma.

Este proceso se hace aún más complejo al no contar con un sistema que actualice la información procedente de las FR de forma automática, por lo que los objetivos que se persiguen en este trabajo son diseñar e implementar una aplicación que solucione este problema, permitiendo enviar información de las FR hacia la UCI y en otras ocasiones desde la UCI hacia la FR en dependencia de las necesidades de estos centros.

El presente trabajo propone solucionar este problema a través de la implementación de un sistema que posibilite la réplica de datos, esta puede ser empleada tanto en un entorno Maestro-Esclavo como en uno Maestro-Maestro, también debe permitir la configuración de reglas para solo replicar información que cumpla con determinadas características. Para ello se emplearán las herramientas más convenientes para la Universidad relacionadas con el almacenamiento de la información las últimas tendencias en el mundo de la Informática.

Palabras clave

Proceso, gestión, sistema, réplica, configuración, reglas.

Índice

Índice de Tablas

ÍNDICE DE FIGURAS

Introducción

Las innovaciones tecnológicas producidas en los últimos años han promovido un cambio en la forma de observar los sistemas de información y en general las aplicaciones informáticas.

Aún cuando es posible que un usuario común no perciba los desarrollos relevantes de nuevos productos, para las aplicaciones existe una demanda permanente por mayor funcionalidad, mayor número de servicios, más flexibilidad y mejor rendimiento. Así, al diseñar un nuevo sistema de información o al prolongar la vida de uno existente, se debe buscar siempre formas para enlazar las soluciones ofrecidas por la tecnología disponible a las necesidades de las aplicaciones de los usuarios.

Un área en la cual las soluciones están integrando tecnología con nuevas arquitecturas o formas de hacer las cosas es, sin lugar a dudas, el área de los sistemas distribuidos de información. Ellos se refieren al manejo de datos que se encuentran en muchos sitios interconectados a través de una red de comunicaciones. Un caso específico de estos sistemas distribuidos es lo que se conoce como bases de datos distribuidas.

La Universidad de las Ciencias Informáticas (UCI), es una de las instituciones preocupadas y ocupadas en el desarrollo de soluciones informáticas aplicadas a su entorno. Dada la necesidad de compartir y acceder a la información, la mayoría de los sistemas informáticos son sistemas distribuidos a lo largo del país por las Facultades Regionales (FR); ubicadas en Artemisa, Ciego de Ávila y Manzanillo respectivamente; interconectadas por una red, donde los nodos forman una topología árbol, donde la UCI desempeña el papel de nodo raíz o primario, como se muestra en la figura 1.

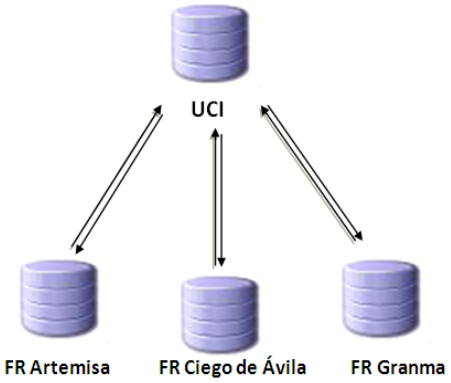


Figura 1 Red de comunicación.

La UCI debe llevar el control de los más de 10 000 estudiantes que posee además de los profesores y trabajadores de otras esferas. Como si esto fuera poco, lleva el control de los estudiantes y profesores en cada una de las FR que supervisa. La gestión de toda esta información es un proceso complejo que depende sobre todas las cosas de la actualidad que tenga la información que se consulta para lograr el éxito pues se puede gestionar información no actualizada y esto provocaría que se tenga que repetir el proceso una vez actualizada esta información. El problema es que no se cuenta con un mecanismo de réplica que permita mantener actualizados estos datos en todo momento.

En la actualidad los procesos de actualización de la información desde las FR hacia la UCI y viceversa que se emplean no satisfacen totalmente las necesidades de ninguna de estas instituciones. La forma de actualización de los datos actual es un proceso donde intervienen varias partes y requiere la gestión de varios aspectos como transporte y recursos humanos para lograr su objetivo.

Esto trae consigo que no en todas las ocasiones se cumpla con los plazos establecidos para las actualizaciones, y al ocurrir esto provoca demora en la gestión de la información y de así dificulta la toma eficiente y rápida de decisiones, por lo que es una necesidad inmediata de la universidad realizar una aplicación o utilizar una existente que facilite el proceso de actualización de los datos, en la base de datos que contiene la información de las FR en la UCI, cada vez que se necesite o en los plazos establecidos para ello.

Los aportes prácticos que se esperan con la realización de este trabajo es que se pueda solucionar el problema antes planteado, donde la información se podrá actualizar desde las FR hacia la UCI lo que ahorraría tiempo, esfuerzos y recursos a la universidad.

Con ese propósito el objetivo general del trabajo es diseñar e implementar una aplicación que realice la réplica de datos entre la UCI y las Facultades Regionales de manera bidireccional y de forma eficiente, y que pueda ser extendida a otros lugares. A partir de este se derivan los siguientes objetivos específicos:

1. Dominar conceptos como los de réplica de datos, sistema de replicación de datos.
2. Conocer sobre la aplicación de la réplica de datos en la universidad.
3. Recopilar información sobre las características que debe cumplir la aplicación.
4. Hacer el diseño del software para luego implementar el mismo.
5. Implementar una aplicación que solucione el problema que existe de actualización de la información.
6. Brindar una ayuda para el entendimiento de las personas que lo empleen.

El objeto de estudio es el proceso de replicación de datos. Y se limita el campo de acción a la réplica de datos soportada en Python y PostgreSQL. Para cumplir con los objetivos planteados se proponen las siguientes tareas de investigación:

1. Buscar información en medios bibliográficos sobre el proceso de réplica de datos y herramientas que la soportan para seleccionar una de ellas.
2. Entrevistar a personas con dominio sobre el tema para conocer el empleo de la réplica de datos en la UCI.
3. Enumerar los requerimientos funcionales y no funcionales para el diseño de la propuesta.
4. Desarrollar el sistema propuesto.
5. Confeccionar la ayuda del sistema.

El contenido del presente trabajo está dividido en 4 capítulos distribuidos de la manera siguiente: En el capítulo 1 se trata la fundamentación teórica del tema, incluye un estudio del estado del arte abordando

temas como las últimas tendencias, técnicas, metodologías y programas que se usan en la actualidad para desarrollar aplicaciones que soportan la réplica de datos. Todo esto a nivel internacional y nacional, enfocándonos en nuestra Universidad como el principal desarrollador de software de nuestro país en estos momentos.

El capítulo 2 describe el sistema que se propone para dar solución a la problemática actual que da origen al mismo. En primer lugar se describen los procesos que dan origen a esta situación y luego los procesos que serán automatizados. Se especifica el modelo de dominio y se recogen las características básicas del sistema expresadas en funcionalidades y en requisitos no funcionales como apariencia, rendimiento, confiabilidad, entre otras.

El capítulo 3 recoge el análisis y diseño de la solución propuesta. Esto incluye la definición del modelo del análisis o las clases del análisis. Además se muestran los diagramas de interacción que representan el flujo principal y alternativo de eventos en cada caso de uso del sistema. Se describen las principales clases del diseño, especificando sus atributos y operaciones.

El capítulo 4 está dedicado a la fase implementación y pruebas. Aquí se ilustra el diagrama de despliegue y de componentes como parte de la implementación. En el caso de la fase de pruebas se describen los casos de pruebas que se emplearon en la misma.

Capítulo 1. Fundamentación teórica

1.1. Introducción

En este capítulo se aborda el estado del arte de la réplica de datos en todos los niveles, ya sea a nivel internacional, nacional como local o de Universidad. Se incluyen las tendencias actuales, las técnicas más empleadas así como las tecnologías y metodologías empleados en esta tarea. Se abarca lo relacionado a los programas más capacitados y utilizados en el tema con el objetivo de acercarnos más a ellos y familiarizarnos con estos, para una futura selección.

1.2. Réplica de datos

Para comprender el objeto de estudio del trabajo se fundamentan algunos conceptos que nos acercarán al contenido de este trabajo.

¿Qué es una base de datos?

Una base de datos (BD) es el lugar donde se agrupan un conjunto de datos organizados que pertenecen a un mismo contexto. Los datos se almacenan sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos. [1]

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

De acuerdo a la ubicación física de las BD se pueden clasificar en estáticas, que son aquellas que no se componen de otras BD sino de los datos almacenados en ella misma; y las distribuidas (BDD), siendo estas una colección de múltiples instancias de BD, distribuidas a través de un variado número de sitios en una red. [2]

Un sitio es una copia de la estructura de la BD, la cual contiene todos los datos o parte de ellos y una aplicación para comunicarse con esta, puede ser:

- Una computadora con su BD local.
- Una BD local compartida en un grupo de trabajo (ej. un servidor de BD compartido entre 5 usuarios en el mismo departamento).
- Una gran BD cliente/servidor utilizada por miles de usuarios.

Una red es una configuración de dos o más sitios interconectados entre sí. Ahora estamos en condiciones de conocer qué es la réplica de datos.

La réplica de datos es un proceso que permite copiar y distribuir idénticamente tablas desde una BD hacia uno o múltiples sitios de una red. Este proceso asegura que los datos estén siempre disponibles en el lugar necesario para ser utilizados en el momento indicado.

Un sistema de replicación es aquel sistema que soporta la réplica de datos, y debe caracterizarse principalmente por:

- *La efectividad*: depende de la forma en la que los datos sean distribuidos y almacenados. A mayor efectividad, mayor será la disponibilidad de datos para ejecutar procesos paralelos.
- *Alta disponibilidad*: es la razón de tiempo prudente en la que un servicio puede ser accedido. En el mejor de los casos puede ser de un 100%, a pesar de los fallos que se puedan presentar en el servidor, ya que debe existir un servidor adicional que posea alguna técnica de replicación que lo pueda suplantar en caso de ser necesario.
- *Tolerancia a fallos*: garantiza un comportamiento correcto, donde efectivamente pueden existir un número finito de fallos y tipos de fallos.
- *Bien coordinado*: cada una de las partes que interviene en la réplica de datos llegan a un consenso para realizar las invocaciones de los servicios a los objetos, que al final de la transacción debe realizarse tal y como fue solicitada, para lo cual debe utilizar algún tipo de ordenamiento, con esto se evitan posibles conflictos durante el proceso. [3]

1.2.1 Entornos de réplica

La réplica de datos puede aplicarse de dos maneras diferentes de acuerdo a las necesidades propias de las personas o instituciones que la lleven a cabo, estas son:

Maestro-Esclavo: También es conocido como de solo lectura, porque permite a un solo sitio (maestro) realizar consultas de escritura sobre los demás, mientras estos solo pueden hacer consultas de lectura (esclavos).

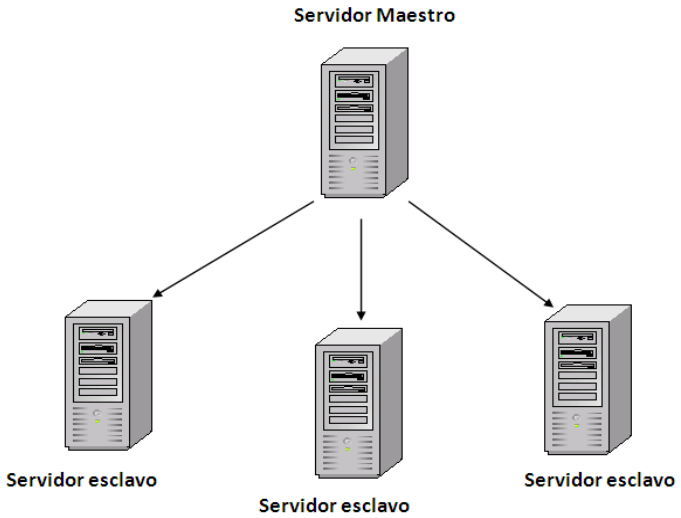


Figura 2 Entorno de Réplica Maestro – Esclavo

Maestro-Maestro*:* También llamado par-a-par o réplica de camino de n, porque permite múltiples sitios (maestros), actuando como pares iguales. Cada sitio es un sitio maestro, y se comunica con otros sitios maestros. Esta capacidad tiene también un severo impacto en el desempeño debido a la necesidad de sincronizar los cambios entre todas las partes que intervienen en la réplica.

Este tipo de entorno puede ser usado para mantener sitios recuperables ante posibles desastres o caídas, así como para proveer sistemas con alta disponibilidad y para balancear la carga de consultas a través de las distintas ubicaciones.

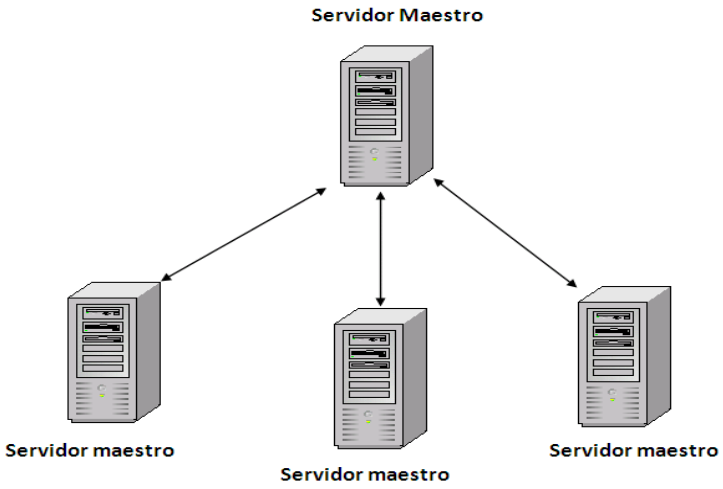


Figura 3 Entorno de Réplica Maestro – Maestro.

1.2.2 Técnicas de replicación

Las diferentes técnicas de réplica de datos son:

- **Confirmación en dos fases:** Esta tecnología es sincrónica, surge a mediados de los años 80. Permite la sincronización de datos distribuidos. Cada transacción solamente es aceptada si todos los sistemas implicados en la réplica están conectados y listos para recibirla, si al menos uno falla, todo el proceso es anulado.
- **Descarga y Recarga:** Esta técnica es asincrónica y consiste en hacer un volcado de los datos, copiar la salva para un dispositivo de almacenamiento para luego distribuir la salva por los demás servidores. Esta técnica presenta el inconveniente de que en la mayoría de las ocasiones se consultan datos que tienen semanas de estar desactualizados, además de que el proceso se realiza de forma manual.
- **Instantánea:** Consiste en hacer una instantánea de una tabla en un momento dado y esta "foto" es la que se replica en las demás bases de datos. Aunque mucho más avanzada que la técnica de Descarga y Recarga esta presenta el inconveniente de que las tablas esclavas son tablas de solo lectura. Se recomienda utilizar cuando la mayoría de los datos no cambian con frecuencia; se replican pequeñas cantidades de datos y cuando los sitios con frecuencia están desconectados y

es aceptable un periodo de latencia largo (la cantidad de tiempo que transcurre entre la actualización de los datos en un sitio y en otro).

- **A través de disparadores:** La función del disparador es ejecutar una acción cuando ocurre un evento en la base de datos, los eventos pueden ser de inserción, actualización o eliminación. Conjuntamente con las instantáneas, los disparadores son otro de los mecanismos asincrónicos que proporciona la base de datos como una manera de replicación de datos. Para mayor comodidad serán llamados Triggers, ya que es el término por el cual son conocidos.

1.2.3 Aplicación de la réplica de datos

Las instituciones utilizan la replicación en sus aplicaciones por varias razones, las cuáles pueden ser categorizadas de la siguiente forma:

1. **Distribuir datos a otras ubicaciones.**
2. **Consolidar datos desde otras ubicaciones.**
3. **Intercambiar datos bidireccionalmente con otras ubicaciones.**

1. La distribución de datos involucra el movimiento de todos o un subconjunto de datos de una o más ubicaciones. La distribución es utilizada para proveer datos a aplicaciones desarrolladas en plataformas similares o diferentes. Esto puede ser tan simple como mantener una copia de los datos de producción en otro sistema similar, así como requerir una transformación compleja para adaptarse a los requerimientos de una nueva aplicación. Los datos que se copien pueden necesitar ser filtrados o transformados para la nueva aplicación. La distribución también puede ser usada para proveer coexistencia de dos sistemas, en la instancia de migración de uno al otro.

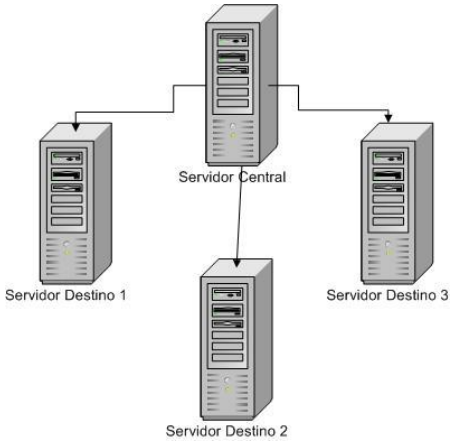


Figura 4 Replicación desde un servidor central hacia dos servidores destinos.

2. Una institución puede tener sus datos en distintos sistemas distribuidos. A modo de ejemplo, una empresa de ventas puede tener sus datos en cada local, una empresa de manufactura los puede tener en cada planta de procesamiento, así como una compañía aseguradora podría tenerlos en cada una de sus oficinas o llegar incluso a cada computador personal de sus corredores. La replicación puede copiar los cambios de cada sitio distribuido al sitio central, para analizar, hacer reportes, o para el procesamiento central de los datos. Otro tipo de replicación bidireccional, es aquella que no tiene designada un servidor maestro. Cada ubicación copia los cambios desde todos los otros sitios directamente. Esto habitualmente sucede en los entornos “multi-maestro” o “par-a-par”.

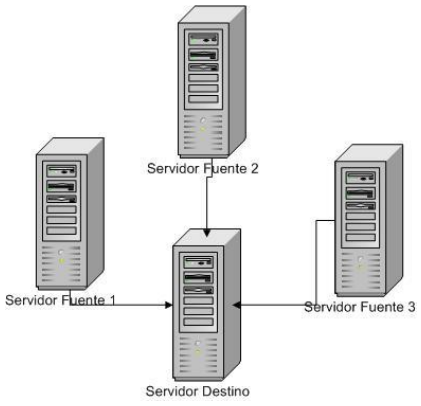


Figura 5 Replicación desde dos servidores fuentes, hacia un servidor destino.

3. Si los datos se pueden modificar en múltiples ubicaciones, entonces la replicación debe procesar los cambios realizados en cada uno de los sitios de forma coordinada. Uno de los servidores, es visto como el servidor maestro, quien se encarga de distribuir los cambios a todos los sitios. Los cambios realizados en los destinos fluyen hacia los otros sitios a través del servidor maestro.

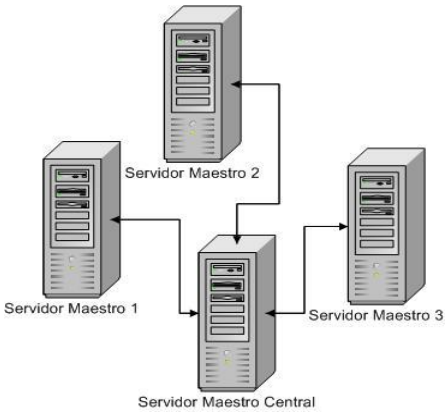


Figura 6 Réplica bidireccional, todos los servidores son maestros

1.2.4 Ambientes de replicación

Un ambiente de replicación es una configuración de dos o más sitios mediante un escenario par a par. Cada sitio es un par que contiene un motor de replicación y una BD simple o compartida. El motor de replicación puede residir en la misma computadora que la BD asociada o en una computadora separada. En cada caso, la BD debe ser accesible por el cliente del motor de replicación (ej. ODBC).

Cada sitio almacena solo los datos que requieren los usuarios locales. En segundo plano, el motor de replicación gestiona los cambios realizados a la BD sincronizando las actualizaciones de los datos con otros sitios activos en la red.

Una red de replicación puede ser de los siguientes tipos:

- ***Homogénea***: Se replican datos entre BD con gestores y plataformas del mismo tipo (ej. PostgreSQL + Windows ←→ PostgreSQL + Windows).
- ***Homogénea con diferentes plataformas***: Se replican datos entre BD con gestores del mismo tipo y plataformas diferentes (ej. PostgreSQL + Windows ←→ PostgreSQL + Linux).

- ***Heterogénea:*** Se replican los datos entre bases de datos en gestores diferentes no importa si el sistema operativo sea el mismo o no (ej. PostgreSQL + Windows ←→ Oracle + Linux).

Una red de replicación requiere una estructura básica TCP/IP que posibilite una comunicación efectiva y eficiente entre todos los sitios. La red de replicación por sí misma constituye estructuralmente una red virtual que se coloca por encima de la red física.

1.2.5 Conflictos de la replicación de datos

Existen varios tipos de conflictos a tener en cuenta, al tratar la replicación de datos. Los conflictos pueden ocurrir cuando estamos trabajando en un ambiente de replicación que permite actualizaciones concurrentes sobre los mismos datos en múltiples sitios.

Estos conflictos son:

- **Conflicto de actualización:** Este conflicto ocurre cuando dos transacciones originadas desde distintos sitios actualizan el mismo registro, en forma cercana en el tiempo.
- **Conflicto de unicidad:** Sucede cuando la replicación de un registro intenta violar una restricción de integridad, ya sea por llave primaria o única (*Primary Key* o *Unique).* Por ejemplo considere lo que sucede cuando dos transacciones originadas de dos sitios diferentes, cada una inserta un registro, en su respectiva tabla, con el mismo valor de clave primaria.
- **Conflicto de supresión:** Un conflicto de supresión ocurre cuando dos transacciones originadas de sitios diferentes, una de ellas intenta borrar un registro, y la otra actualizar o borrar el mismo registro, ya que en este caso el registro no existe, tanto para ser actualizado como para ser eliminado.
- **Conflicto de orden:** Los conflictos de orden pueden ocurrir en ambientes de replicación con tres o más sitios maestros. Si la propagación al sitio maestro X, está bloqueada por alguna razón, entonces la replicación de modificaciones en datos puede seguir siendo propagada a través de los otros sitios maestros; al finalizar la propagación, estas modificaciones debieron ser propagadas al sitio X en un orden diferente a como ocurrieron en los otros sitios maestros, pudiendo producirse un conflicto.

1.2.6 Resolución de conflictos

La manera que se pueden resolver cada uno de estos conflictos se explica a continuación. Para evitar conflictos de:

Actualización

Para este caso existen tres formas aceptadas de resolverse:

- **Prioridad:** Cada servidor obtiene una prioridad única, y el servidor de mayor prioridad gana respecto a los demás servidores.
- **Timestamp**: La más nueva o la más antigua de las modificaciones es la considerada correcta, y por defecto, si no se eligió ninguno de los criterios gana la más nueva.
- **Particionamiento de datos:** Se garantiza que cada registro sea manipulado por un único servidor, lo que simplifica la arquitectura.

Unicidad

También existen tres formas aceptadas para resolverse:

- Para cada servidor brindar un rango distinto de números para los generadores de claves.
- Agregar el identificador del servidor a la clave primaria.
- Replicar en tablas separadas, y acceder a los datos a través de una vista formada por la unión de ellas. Para resolver el conflicto de potenciales claves duplicadas en la unión se usará una p-seudo columna que representa la base de datos fuente.

Supresión

Para evitar este tipo de conflictos, una posible solución es que los sitios marquen lógicamente los registros a ser borrados y que periódicamente el sitio maestro corra un proceso que realice el “delete” físico de los datos, es decir que desde los sitios replicados no se puede ejecutar una sentencia “delete”.

Orden

Este tipo de conflicto suele resolverse asignándole distintas prioridades a los sitios maestros, de forma que se ordenen las transacciones de acuerdo a estas prioridades.

1.3 Estado del Arte

En el ámbito internacional las soluciones que se le han dado a la réplica de datos son principalmente soluciones que se apoyan en el funcionamiento de herramientas como PgCluster, Slony, Pgpool y PgReplicator. Estas herramientas son sistemas basados en ficheros de configuración que se crean manualmente por la persona encargada de la réplica. Algunas similitudes entre ellas son que están implementador en el lenguaje C, son un poco complejas al tener muchas líneas de código, con juegos de comandos propios para su administración.

Dentro de las soluciones propuestas para la réplica de datos en nuestro país, la mayor parte de estas soluciones has sido propuesta por la UCI por ser el más productor de software del país.

Las más interesantes de estas soluciones son:

- Réplica bidireccional basada en control de cambios.
- Sistema de réplica para bases de datos distribuidas en PostgreSQL.
- Solución para la réplica de datos del Proyecto Prisiones.

Ahora fundamentamos un poco cada una de estas soluciones:

- **Réplica bidireccional basada en control de cambios**

Esta solución fue propuesta por el proyecto Identidad de la facultad 1, se caracteriza fundamentalmente por ser bidireccional y basado en control de cambios y no en colas lo que evita las transmisiones redundantes. Soporta ambientes heterogéneos, no importa el gestor de base de datos. Soporta particionamiento horizontal de los datos y garantiza la integridad de los datos pues provee mecanismos de detección y resolución de conflictos, está implementado sobre la plataforma .NET y posee una topología flexible y escalable. No soporta el entorno de replicación maestro-esclavo.

- **Sistema de réplica para bases de datos distribuidas en PostgreSQL**

Es una aplicación de escritorio programada en Python y como lenguaje para la interfaz grafica la librería GTK y la librería GLADE. Realiza la réplica de datos en un solo entorno de réplica que es maestro-

esclavo, y solo está implementada para el gestor de base de datos PostgreSQL, como herramienta de replicación utilizan Slony I. Sus principales funcionalidades son:

- Configurar la réplica de espejo.
- Configurar la réplica fragmentada.
- Reporte de réplica.
- Realizar mantenimiento.

- **Solución para la réplica de datos del Proyecto Prisiones**

La solución propuesta por el Proyecto Prisiones de la facultad 4 se caracteriza en que soporta la replicación de transacciones a través de Hibernate, JDBC y la replicación de acciones a través de disparadores, se integra con el SGBD Oracle en estos momentos, aunque se encuentran en fase de desarrollo el soporte para PostgreSQL, MySQL y Microsoft SQL Server.

Utiliza los protocolos TCP/IP, FTP y HTTP para la transferencia de datos de replicación, los archivos de gran tamaño son enviados por FTP permitiendo resumir la transmisión en caso de interrupciones en la red. Los conflictos se detectan y pueden ser solucionados interactivamente o automáticamente mediante la definición de reglas para la resolución de estos (este módulo de resolución de conflictos está actualmente en desarrollo). Presenta interfaz visual web, por lo que se puede administrar y configurar remotamente solo con emplear un navegador web.

1.4 Tendencias y tecnología actuales

El creciente desarrollo de la informática a nivel mundial ha sido causa de competencias entre productores de software por el poder y dinero, en este sentido, esto ha traído como consecuencia la aparición de nuevas tecnologías cada vez más robustas, herramientas cada vez más útiles y lenguajes de programación más potentes.

Analizaremos las últimas tendencias en cada una de estos aspectos con el objetivo de seleccionar la que utilizaremos en el desarrollo de nuestra propuesta.

1.4.1 Lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar un programa de forma colaborativa.

A continuación caracterizamos algunos de los lenguajes más utilizados en los últimos años según encuestas hechas a usuarios en Internet.

1.4.1.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por *Sun Microsystems* a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un código representado por la unidad básica de almacenamiento de información (*bytecode*), aunque la compilación en código máquina nativo también es posible. En tiempo de ejecución, este código es normalmente interpretado o compilado a código nativo, aunque también es posible la ejecución directa por hardware por un procesador Java.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por *Sun Microsystems* en 1995. Desde entonces, *Sun* ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través de la Comunidad de Java (*Java Community Process*), otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre. [6]

1.4.1.2 Pre-procesador de hipertexto

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*), pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas QT o GTK+.

Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por El Grupo PHP (*The PHP Group*), y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la *PHP License*, la Fundación de Software Libre (*Free Software Foundation*) considera esta licencia como software libre.

Entre las ventajas de este lenguaje se encuentran que es un lenguaje multiplataforma, la capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destacándose su conectividad con MySQL, su capacidad de expandir su potencial utilizando la enorme cantidad de módulos (extensiones), posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda, es libre, por lo que se presenta como una alternativa de fácil acceso para todos. Además permite las técnicas de programación orientada a objetos, no requiere definición de tipos de variables y tiene manejo de excepciones a partir PHP5.

Como aspectos negativos decir que si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aún estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. [7]

1.4.1.3 C++

Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos como la programación estructurada y la programación orientada a objetos. Por esto se suele decir que el C++ es un lenguaje multiparadigma.
Una particularidad del C++ es la posibilidad de redefinir los operadores o lo que se conoce como sobrecarga de operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales. Permite trabajar tanto a alto como a bajo nivel. [8]

1.4.1.4 Python

Python es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. En la actualidad Python se desarrolla como un proyecto de código abierto, administrado por la fundación de software de Python. El nombre del lenguaje proviene de la afición de su creador original, Guido van Rossum, por los humoristas británicos Monty y Python. El principal objetivo que persigue este lenguaje es la facilidad, tanto de lectura, como de diseño.

Permite dividir el programa en módulos reutilizables desde otros programas Python. Viene con una gran colección de módulos estándares que se pueden utilizar como base de los programas o como ejemplos para empezar a aprenderlo. También hay módulos incluidos que proporcionan entrada y salida de ficheros, llamadas al sistema, puntos terminales de conexión (sockets) y hasta interfaces gráficas de usuario (*GUI*) como TK, GTK, QT, PythonCard, entre otros. Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

Es soportado en las siguientes plataformas: Unix, Windows, OS/2, Mac, además de la capacidad para integrarse con otras plataformas de desarrollo como Java, para lo cual usa Jython, una implementación de Python para la máquina virtual de Java. La implementación de Python es bajo la licencia de código abierto que permite el uso libre y distribución para un uso comercial. La licencia de Python es administrada por la Fundación Python de Software.[9]

1.4.1.5 Wxpython

Wxpython nació cuando Robin Dunn necesitaba una GUI que funcionara en sistemas HP-UX y también en Windows 3.1 en unas pocas semanas. Mientras evaluaba las soluciones comerciales, se encontraron uniones (*bindings*) de Python para las herramientas wxWidgets. Por esto, aprendió Python en un corto tiempo, y se convirtió en uno de los principales desarrolladores de Wxpython, junto a Harri Pasanen.

Las primeras versiones del envoltorio (*wrapper*) fueron creadas a mano. Sin embargo, el código base no tardó en tornarse muy difícil de mantener en sincronización las versiones de wxWidgets. Pero versiones posteriores fueron creadas con SWIG, reduciendo enormemente la cantidad de trabajo necesario para actualizar el envoltorio. La primera versión moderna fue anunciada en 1998.

Está implementado como un módulo de extensión de Python que envuelve a la popular biblioteca multiplataforma wxWidgets, que está escrito en C++. Como Python y wxWidgets, Wxpython es código abierto lo que significa que es libre para cualquier persona a utilizar y el código fuente está disponible para cualquier persona para ver y modificar, o cualquier persona puede contribuir correcciones o mejoras al proyecto.

En la actualidad está soportado en las plataformas de 32 bits de Microsoft Windows, Unix o la mayoría de sistemas tipo UNIX, y Macintosh OS X. Dado que el idioma es Python, los programas Wxpython son sencillos, fáciles de escribir y fácil de entender.[10]

1.4.1.6 Pl/Python

PlPython es un lenguaje Procedural que permite escribir funciones Python para la base de datos relacional PostgreSQL. Como se puede acceder a todas las funciones de Python, no debe usarse para usuarios no confiados, por eso se lo llama plpythonu donde la u significa desconfianza (*untrusted*).

El cuerpo de una función plpythonu es simplemente un script de Python. Cuando la función es llamada, sus argumentos son pasados como elementos de una lista de argumentos (*args)*; estos son pasados como variables ordinarias. El resultado es devuelto de la manera usual, con un retorno (*return)* o una entrega (*yield)* en el caso que devuelvan un conjunto de resultados. Los valores NULL de PostgreSQL equivalen a *None* en Python.

Está disponible el diccionario SD para almacenar datos entre cada llamada a función, y el diccionario global GD para usarse desde todas las funciones.

1.4.2 Gestores de bases de datos

Los sistemas de gestión de base de datos (*Database management system*, *SGBD*) son un tipo de programa muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los SGBD es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante.

Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos. Proveen mecanismos para garantizar la recuperación de la base de datos hasta un estado consistente conocido en forma automática. Proporcionan una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder, es también frecuente que dichos accesos se realicen de forma simultánea.

1.4.2.1 MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desde enero de 2008 es una subsidiaria de (*Sun Microsystems*) y ésta a su vez de Oracle Corporación desde abril de 2009.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y la patente código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee la patente de la mayor parte del código.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.[11]

1.4.2.2 SQL Server

Microsoft SQL Server es un SGBD relacional basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son *Oracle*, *Sybase ASE*, *PostgreSQL*, *Interbase*, *Firebird* o *MySQL*.

Las características principales de Microsoft SQL Server son*:*

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida(*MSDE*), con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita. En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD. Para el desarrollo de aplicaciones más complejas (tres o más capas), *Microsoft SQL Server* incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows.[12]

1.4.2.3 Oracle

Oracle es un SGBD relacional desarrollado por Corporación Oracle (*Oracle Corporación)*. Surge a finales de los 70 bajo el nombre de *Relational Software* a partir de un estudio sobre SGBD de George Koch. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

Ha sido criticada por algunos especialistas la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los

usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y otros con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

La única edición gratuita es la *Express Edition*, que es compatible con las demás ediciones de Oracle base datos 10gR2 y Oracle base datos 11g.[13]

1.4.2.4 Firebird

Firebird es un SGBD relacional de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por *Borland* en el 2000. Su código fue reescrito de C a C++. El proyecto se ha desarrollado activamente y el 18 de abril de 2008 fue liberada la versión 2.1.

Sus principales características son:

- Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows.
- Ejecutable pequeño, con requerimientos de hardware bajos.
- Arquitectura Cliente/Servidor sobre protocolo TCP/IP y otros (*embedded*).
- Soporte de transacciones ACID y claves foráneas.
- Es medianamente escalable.
- Buena seguridad basada en usuarios (*roles*).
- Bases de datos de sólo lectura, para aplicaciones que corran desde dispositivos sin capacidad de escritura, como CD-ROM.
- Existencia de controladores ODBC, OLEDB, JDBC, PHP, Perl, .net, etc.
- Pleno soporte del estándar SQL-92, tanto de sintaxis como de tipos de datos.
- Completo lenguaje para la escritura de disparadores y procedimientos almacenados denominado PSQL. [14]

1.4.2.5 PostgreSQL

PostgreSQL es un SGBD objeto-relacional (*Object Relational Database Manager System*), basado en Postgres versión 4.2, desarrollado en la Universidad de California, en el Departamento de Ciencias de la Computación de Berkeley.

Postgres es pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas de bases de datos comerciales mucho más tarde. PostgreSQL es un descendiente del "código abierto", soporta gran parte del SQL estándar y muchas modernas funcionalidades como:

- Consultas complejas.
- Llaves foráneas.
- Disparadores (en inglés, Triggers).
- Vistas.
- Integridad transaccional.
- Control de versionado concurrente (MVCC en sus siglas en ingles). Estrategia de almacenamiento que permite trabajar con grandes volúmenes de datos.

Cumple completamente con las características atomicidad, consistencia, aislamiento y durabilidad (*Atomicity, Consistency, Isolation and Durability*: *ACID*) para realizar transacciones seguras, es multiplataforma, está disponible para 34 plataformas en su última versión estable. Posee interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY, además de traer soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos (*Secure Sockets Layer: SSL*).

Además, PostgreSQL puede ser personalizado por el usuario en muchas formas, según sus necesidades, por ejemplo, adicionando entre otros, un nuevo:

- Tipo de datos.
- Funciones.
- Operadores.
- Funciones agregadas.
- Lenguajes procedurales.

Y debido a la liberación de la licencia, PostgreSQL puede ser utilizado, modificado y distribuido por cualquiera gratuitamente, para cualquier propósito ya sea con fines privados, comerciales o académicos.[15]

1.4.3 Interfaz gráfica de usuario

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

La interfaz gráfica de usuario (*Graphical User Interface*: *GUI*) es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico.

1.4.3.1 GTK

GTK es una biblioteca para crear GUI. Ofrece un juego completo de elementos para construir una interfaz de usuario a la medida. Se llama GIMP Toolkit porque fue escrito para el desarrollo del Programa de Manipulación General de Imagen (*General Image Manipulation Program*), pero ahora GTK se utiliza en un gran número de proyectos de programación, incluyendo el proyecto de entorno de escritorio multiplataforma para Linux GNOME (*GNU Network Object Model Environment*).

GTK es software libre al estar bajo la Licencia Pública Menor de GNU (LGPL); y parte importante del proyecto GNU. Cabe mencionar que Qt es una alternativa a GTK que también es muy utilizada (en el entorno KDE, por ejemplo).

GTK es basado en tres librerías desarrolladas por el equipo de GTK:

- *Glib* es la biblioteca de bajo nivel que constituye la base principal de GTK y GNOME. Suministra el manejo de estructura de datos para C, cubierta de portabilidad, e interfaces para funcionalidades de tiempo de ejecución, bucle de evento, hilos, carga dinámica, y un sistema de objeto.

- *Pango* es una biblioteca para el diseño y la versión del texto, con un énfasis sobre la internacionalización. Constituye el punto principal de texto y tipo de caracteres respondiendo para GTK +-2.0.
- La biblioteca *ATK* suministra un juego de interfaces para la accesibilidad. Soportando las interfaces de ATK, una aplicación o juego de herramientas pueden ser usados con tales herramientas como lectores de pantalla y dispositivos de entrada alternativos, entre otros.

GTK ha sido diseñado desde el principio para respaldar un varios leguajes, no solamente C/C++. Usando GTK desde lenguajes como Perl y Python, especialmente en combinación con Glade, constructor de Interfaz Gráfica de Usuarios; brinda un método eficaz del desarrollo rápido de aplicaciones.[16]

1.4.3.2 PythonCard

PythonCard es un conjunto de herramientas para la construcción *GUI* para crear aplicaciones de escritorio multiplataforma en Windows, Mac OS X, y Linux, usando Python. La motivación de PythonCard es *"Las cosas simples deberían ser simples, y las complejas deberían ser posibles"* [4]. PythonCard es para desarrollar aplicaciones gráficas de manera rápida y fácil con un mínimo esfuerzo y código. Es parecido a Visual Basic Clásico, con respecto a la creación visual de interfaces gráficas y programación por eventos.

PythonCard usa Wxpython. Si estás familiarizado con Wxpython, se puede ver a PythonCard como una manera simple de hacer programas Wxpython con un gran conjunto de ejemplos y herramientas que se pueden copiar y mejorar para crear aplicaciones multiplataforma.

PythonCard es un proyecto de código abierto y está siendo desarrollado bajo los términos de una licencia estilo BSD. Esto básicamente significa que está libre descargar y utilizar los ejecutables, código fuente, páginas web o cualquier otro artículo producido por el proyecto y utilizarla como lo desea, siempre y cuando reconozcan la fuente de ese tema y reproducir la licencia asociado a él.[17]

1.4.4 Herramientas para la réplica de datos

1.4.4.1 Pgpool

Pgpool es un programa intermediario (*middleware*) que trabaja entre PostgreSQL y servidores de una base de datos PostgreSQL cliente. Tiene las siguientes características:

- Trabaja entre los servidores de datos y el cliente para reducir el consumo (*overhead*) de establecimiento de la conexión.
- Trabaja hasta con 2 servidores, si el principal cae automáticamente trabaja con el otro.
- Puesta en común de conexión, guarda conexiones con el servidor PostgreSQL, y la reutilización siempre que sea una nueva relación con las mismas propiedades (es decir, nombre de usuario, bases de datos, versión del protocolo). Reduce de conexión y mejora el rendimiento general del sistema.
- Permite la replicación ya que puede gestionar múltiples servidores PostgreSQL. El uso de la función de replicación en tiempo real permite crear una copia de seguridad en 2 o más discos físicos, por lo que el servicio pueda continuar sin parar los servidores en caso de un fallo de disco.
- Limita conexiones superiores: hay un límite en el número máximo de conexiones simultáneas con PostgreSQL, y las conexiones se rechazan después de esta cantidad de conexiones.

1.4.4.2 Slony I

Slony es el plural de elefante en el idioma ruso. La mascota para Slony: elefante, es una muy buena variación del usual elefante mascota de Postgres. Es un sistema de replicación “maestro a múltiples esclavos”, soporta la réplica en cascada y permite a un esclavo ser a su vez maestro para otro servidor.

Incluye los tipos de funcionalidades necesarias para replicar bases de datos grandes a un número razonablemente limitado de sistemas esclavos. En este contexto razonable es un número no superior a 10 servidores. Si el número de servidores crece más allá de 10, el costo de las comunicaciones aumentará prohibitivamente, y las ventajas incrementales de tener servidores múltiples fallarán en ese punto.

Además permite indicar qué cambios replicar de un servidor a otro. Slony-I implementa la réplica asincrónica, usando disparadores para determinar las actualizaciones de las tablas, donde un solo origen (maestro) se puede replegar a los suscriptores múltiples (esclavos) incluyendo suscriptores conectados en

cascada. Slony I realiza una réplica de espejos, exactamente igual al origen de datos, no es posible actualizar los datos a medida que se produce algún cambio en ellos.[18]

1.4.4.3 PgCluster

Es un sistema de replicación sincrónico multi-maestro para PostgreSQL. Consiste en tres tipos de servidores, un servidor para balance de carga, un clúster de base de datos y un servidor de réplica.

Tiene dos funciones principales:

- **Compartir carga**
  - La carga de la sesión de las demandas es distribuida. Es efectivo en aplicaciones Web donde existe gran demanda por el número de peticiones.
- **Alta disponibilidad**
  - Cuando ocurre un fallo en el Clúster DB, el servidor de balance de carga y el de replicación separan el fallo del sistema, y continúa el servicio que usa el DB restante.
  - El Clúster DB cuando es reparado puede restaurarse dinámicamente a un sistema, sin detener el servicio.
  - Los datos son copiados automáticamente a la DB restaurada o añadidos desde otra DB.[19]

1.4.4.4 Pyreplica

Pyreplica es una herramienta simple que permite la replicación asincrónica de datos para PostgreSQL, utilizando disparadores implementados en pl/python, señales, secuencias y un script cliente influenciado por Slony pero mucho más simple y fácil. Soporta además la replicación en un entorno maestro-maestro, donde se replican datos en ambas direcciones.

Está programado en Python por lo que es simple y flexible, permitiendo una fácil instalación, no se necesita aprender nuevos comandos para su administración, es muy fácil de adaptar manualmente, es eficiente pues tiene un bajo impacto en el uso de memoria y la red al no utilizar transmisión secuencial (*polling*) y es multiplataforma. [20]

Le brinda al usuario diversas funcionalidades como son:

- Permite réplica maestro-esclavo y multi-maestro limitado.
- Detección de conflictos.
- Notificaciones vía Email.
- Monitoreo de las conexiones (*KeepAlive*).
- Conexiones directas a las etapas finales (*backends*).
- Sin protocolos especiales (SQL textual).
- Si herramientas externas.
- Protegido con transacciones en dos fases.

1.4.5 Herramientas de desarrollo

1.4.5.1 ArgoUML

Herramienta que contiene funciones avanzadas en las etapas de diseño y modelación de software. Está desarrollado sobre licencia comercial y su precio oscila entre 1000 y 4500 USD. Sus principales características son que tiene muy buenas ayudas, soporta todas las especificaciones UML y está integrado con la WEB [21]

1.4.5.2 Poseidon

Es una herramienta para modelar cualquier clase de sistema, tenga o no que ver con programación, presenta licencia Community Edition y Professional Edition. Sus principales características son que soporta diagramas UML, presenta opciones avanzadas de impresión, soporta gráficos en la mayoría de los formatos, se puede generar código para Java y exportar elementos en HTML, es fácil de instalar y actualizar, y es compatible entre ediciones.

1.4.5.3 MagicDraw UML

Herramienta de modelaje con completas características UML. Ha sido implementada totalmente en JAVA. Tiene licencia MagicDraw Reader (gratis), MagicDraw Community Edition (gratis para desarrollo de proyectos no comerciales) y The Professional Edition (comercial).

1.4.5.4 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor

coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

1.4.6 Otras herramientas

1.4.6.1 Psycopg2

Psycopg2 es un conductor compatible con PostgreSQL y que está bajo desarrollo activo. Está diseñado para aplicaciones con subprocesos múltiples y administra su propia conexión a la piscina.

Otra característica interesante del conductor es que si se utiliza el tipo de datos PostgreSQL matriz, psycopg2 convertirá automáticamente un resultado que contenga ese tipo de datos a una lista de Python. [22]

1.4.7 Tecnologías

1.4.7.1 Software libre

El movimiento Código Abierto apareció en 1998 con un grupo de personas, entre los que cabe destacar a Eric S. Raymond y Bruce Perens, que formaron la Iniciativa de Código Abierto (*Open Source Initiative*, OSI). Buscaban darle mayor relevancia a los beneficios prácticos de compartir el código fuente e interesar a las principales casas de software y otras empresas de la industria de la alta tecnología en el concepto. El movimiento del software libre hace especial énfasis en los aspectos morales o éticos del software, viendo la excelencia técnica como un producto secundario deseable de su estándar ético. Este movimiento ve la excelencia técnica como el objetivo prioritario, siendo la compartición del código fuente un medio para dicho fin.

Software libre es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. Análogamente, el software gratuito (*Freeware*) incluye en algunas ocasiones el código fuente; sin embargo, este tipo de software no es libre

en el mismo sentido que el software libre, al menos que se garanticen los derechos de modificación y redistribución de dichas versiones modificadas del programa.

De acuerdo con tal definición, el software es "libre" si garantiza:

- la libertad para ejecutar el programa con cualquier propósito (llamada "libertad 0").
- la libertad para estudiar y modificar el programa ("libertad 1").
- la libertad de copiar el programa de manera que puedas ayudar a tu vecino ("libertad 2").
- la libertad de mejorar el programa, y hacer públicas tus mejoras, de forma que se beneficie toda la comunidad ("libertad 3").

Es importante señalar que las libertades 1 y 3 obligan a que se tenga acceso al código fuente.

En 1984, Richard Stallman comenzó a trabajar en el proyecto GNU (es un acrónimo recursivo para "Gnu No es Unix"), fundando la Fundación de Software Libre (*Free Software Foundation, FSF*) un año más tarde. Stallman introdujo una definición para software libre (en inglés, free software), el cual desarrolló para dar a los usuarios libertad y para restringir las posibilidades de apropiación del software.

1.4.8 Metodologías

La mayoría de las tecnologías usadas para el diseño de aplicaciones utilizan el lenguaje UML como esencia en el modelado de las relaciones entre los objetos del diseño.

Lenguaje unificado de modelado

son las siglas de Lenguaje Unificado de Modelado (*Unified Modeling Language*), notación esquemática en su mayor parte con que se construyen sistemas por medio de conceptos orientados a objetos.

El UML es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML permite una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

UML permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar *de facto* de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Tiene como objetivo brindar un material de apoyo que le permita al lector poder definir diagramas propios como también entender diagramas ya existentes.

Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. [23]

1.4.8.1 Proceso Unificado del software (RUP)

Para desarrollar un software se necesita una forma ordenada de trabajo. Un proceso que integre las múltiples facetas del desarrollo. Se necesita un método común, un proceso que:

- Proporcione una guía para ordenar las actividades de un equipo.
- Dirija las tareas de cada desarrollador por separado y del equipo como un todo.
- Especifique los artefactos que deben desarrollarse.
- Ofrezca criterios para el control y la medición de los productos y actividades de proyectos.

El Proceso Unificado es un proceso de desarrollo de Software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el Proceso Unificado es más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto. El Proceso Unificado está basado en componentes, lo cual quiere decir que el sistema de software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

El Proceso Unificado utiliza el UML para preparar todos los esquemas de un sistema software. De hecho, UML, es una parte esencial del Proceso Unificado – sus desarrollos fueron paralelos.

No obstante, los verdaderos aspectos definitorios del Proceso Unificado se resumen en tres fases claves: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. Esto es lo que hace único al proceso unificado.[24]

1.4.8.2 Programación extrema (XP)

XP (*eXtreme Programing*) nace como nueva disciplina de desarrollo de software hace aproximadamente unos seis años, y ha causado un gran revuelo entre el colectivo de programadores del mundo. Kent Beck, su autor, es un programador que ha trabajado en múltiples empresas y que actualmente lo hace como programador en la conocida empresa automovilística DaimlerChrysler. Con sus teorías ha conseguido el respaldo de gran parte de la industria del software y el rechazo de otra parte. La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica.

Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.

El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.[25]

1.4.8.3 Scrum

Scrum es un proceso de desarrollo de software iterativo y creciente utilizado comúnmente en entornos basado en el desarrollo ágil de software. Aunque Scrum estaba enfocado a la gestión de procesos de desarrollo de software, puede ser utilizado en equipos de mantenimiento de software, o en una aproximación de gestión de programas: Scrum de Scrums.

Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. Los roles principales en Scrum son el maestro de Scrum (*ScrumMaster)*, que mantiene los procesos y trabaja de

forma similar al director de proyecto, el *ProductOwner*, que representa a los clientes externos o internos (*stakeholders*), y el equipo (*Team*) que incluye a los desarrolladores.

Existen varias implementaciones de sistemas para gestionar el proceso de Scrum, que van desde notas amarillas y pizarras hasta paquetes de software. Una de las mayores ventajas de Scrum es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.[26]

1.5 Conclusiones

En este capítulo se han introducido conceptos indispensables para la comprensión del proceso de distribución de la información con el uso de las tecnologías actuales.

Se analizaron las distintas soluciones aportadas por la Universidad a la réplica de datos y se llega a la conclusión de que aunque estas resuelven el problema de intercambiar información tienen como desventajas que están implementadas en C lo que trae consigo la compilación y los errores de ejecución.; tienen decenas de ficheros y miles de líneas de código por lo que resultan complejos, con juegos de comandos propios para su administración; algunos no funcionan en Windows o resulta difícil su instalación y son inestables en condiciones extremas. Todas estas razones son basadas en experiencias de personas con dominio del tema.

Se propone realizar una nueva solución donde se utilice la herramienta Pyreplica por su simplicidad y capacidad para realizar la réplica en ambos entornos, además de su flexibilidad y sencillez al estar desarrollado en Python. Su facilidad para instalarlo, administrarlo y adaptarlo son otros aspectos a tener en cuenta también para su selección. Resulta eficiente al causar bajo impacto en la utilización de la memoria y la red.

Por la facilidad para conectar el Pyreplica mediante psycopg2 con PostgreSQL, este es el SGBD que emplearemos, además de ser relacional orientada a objetos y libre, publicado bajo la licencia BSD y de su capacidad para desempeñarse eficientemente en múltiples plataformas. Como lenguaje de programación seleccionamos Python porque en primer lugar el Pyreplica está implementado en este lenguaje, y en segundo lugar porque es multiplataforma y multiparadigma permitiendo varios estilos de programación como el orientado a objeto, el imperativo y el funcional, además es un lenguaje interpretado mediante script lo que posibilita que consume poco recursos y un desarrollo rápido de las aplicaciones.

Por estar soportado en este mismo lenguaje para el desarrollo de la GUI seleccionamos el PythonCard, además es de su facilidad de diseño y su sencillez con el empleo de Wxpython.

La metodología propuesta para el desarrollo de la aplicación es RUP, por su capacidad para guiar a los trabajadores a desarrollar un programa, por ser iterativo y tener como objetivo la entrega gradual de soluciones. Para el modelado de la aplicación se utilizó Visual Paradigm como software de desarrollo por ser libre y por sus nuevas características como soporte de un modelo colaborativo con CVS y Subversion, por sus capacidades en la generación de código, de objetos y de bases de datos.
Además de esto, estos programas se establecen en la arquitectura del proyecto.

Capítulo 2. Características del sistema.

2.1 Introducción

En el presente capítulo se hace la descripción de la propuesta de solución al problema actual de actualización de información desde las FR, es importarte conocer cómo funcionan estos procesos, ya que ayudan a tener una idea más clara del funcionamiento de la Universidad para elaborar una aplicación que se ajuste mejor a las necesidades de la misma.

Además, se enumeran los requisitos funcionales y no funcionales que debe cumplir el sistema que se propone, y se identifican mediante un Diagrama de Casos de Uso, las relaciones de los actores con las funcionalidades que brinda el sistema.

2.2 Descripción del objeto de estudio

2.3 Situación problémica

Desde su comienzo la UCI ha sido una de las instituciones en nuestro país preocupada y dedicada al desarrollo de software, con el surgimiento de las FR de Artemisa, Ciego de Ávila y Manzanillo el incremento de la gestión de la información se ha hecho notar y se ha convertido en prioridad viabilizar el proceso de actualización de la información de los estudiantes que cursan sus estudios en estos centros hacia la Universidad.

Para nadie es desconocida la gran cantidad de estudiantes para la cual la UCI debe llevar el control, incluyendo expedientes y resultados docentes. Además maneja lo relacionado a los ingenieros que han sido graduados, esté o no prestando servicios en la universidad.

La gestión de toda esta información es un proceso nada sencillo que depende sobre todas las cosas de la actualidad que tenga la información que se manipule para lograr el éxito pues se puede gestionar información no actualizada lo que provocaría que se tenga que repetir el proceso de gestión una vez

actualizada la información, por no contar con mecanismos que se encarguen de esto y causaría retrasos en la toma de decisiones.

Este proceso de intercambio de información se lleva a cabo de una manera ineficiente, pues no se cuenta con mecanismos o sistemas capaces de efectuarlo con la mayor efectividad y aceptación posible.

La actualización de la información se lleva a cabo de la siguiente forma:

- La secretaria docente general solicita al Departamento de Informatización (DI) actualizar los datos relacionados con los estudiantes de las FR.
- El DI designa a un especialista para que viaje a una facultad regional y traiga los datos.

Figura 7 Ubicación geográfica de las Facultades Regionales.

- Esto requiere la gestión de un transporte para trasladar al compañero designado hacia la FR y regresarlo a la universidad, este proceso puede demorar de 10 a 15 días.
- Una vez resuelto el transporte el especialista viaja a la FR y realiza un respaldo de toda la base de datos y se trae para la universidad en algún dispositivo de almacenamiento.
- Al llegar a la universidad el especialista se encarga de actualizar la información en la base de datos central mediante una aplicación desktop implementada con software propietario (C Sharp .NET) y esto se hace así:

- Para los estudiantes de nuevo ingreso se emplea la aplicación desktop antes mencionada que se encarga de insertar la información del estudiante en la base de datos.
- Para los estudiantes continuantes este proceso se realiza manualmente a través de sentencias y consultas SQL.

Después de actualizada la información cualquier cambio que se realice en una facultad regional la secretaria del centro debe informar a la secretaria general de la UCI para que esta realice los cambios en la base de datos de aquí.
Todo este largo proceso trae como consecuencias que la mayor parte del tiempo se consulte información desactualizada, repercutiendo además en la demora del tiempo desde que se solicita la actualización por parte de la secretaria hasta que se concrete la misma por parte del DI, afectando la toma eficiente y rápida de decisiones.

2.4 Información que se maneja

A la universidad le urge la necesidad de tener el control de cada estudiante y profesores, por lo que con este componente se realiza un flujo de información muy amplio ya que desde las FR se replican hacia la UCI la información que identifica a cada estudiante y profesores para desde ese modo la universidad llevar un control estricto de cada FR.

2.5 Propuesta de Solución

En el capítulo 1 se concluyó con que la herramienta que se va a utilizar para realizar la réplica de datos es el Pyreplica. Esta herramienta al igual que las demás son sistemas que se configuran a través de ficheros creados manualmente, por lo que se propone la realización de una aplicación Desktop que permita la configuración de la herramienta en cualquier entorno, posibilitando además insertar y eliminar reglas de replicación, realizar copias de respaldo de las bases de datos involucradas en la réplica así como restaurar una a partir de una salva previamente realizada, generar reportes de réplicas en un intervalo determinado y monitorear la réplica en tiempo real. La misma se desarrollará con el empleo de las herramientas seleccionadas: Python, PythonCard, Psycopg2, PostgreSQL.

El funcionamiento básico del Pyreplica es el siguiente:

Consiste en un disparador plpythonu de registro (*py_log_replica*) y un script demonio esclavo (pyreplica.py y daemon.py).

El disparador almacena un registro de replicación (instrucciones de manipulación de datos INSERT, UPDATE, DELETE sobre las tablas afectadas en la tabla *replica_log*) y señala con un mensaje NOTIFY a las réplicas.

```
if event == 'INSERT':
  sql = 'INSERT INTO "%s" (%s) VALUES (%s)' % (
          relname,
          ', '.join(['"%s"' % k for k in new.keys()]),
          ', '.join([mogrify(v) for v in new.values()]),
        )
elif event == 'UPDATE':
  modified = [(k,v) for (k,v) in new.items() if old[k]<>v]
  if modified: # only if there are modified fields
    if warn_conflicts:
      # Insert a query to warn if there is a data conflict
      sql = 'SELECT %s,%s,%s FROM "%s" WHERE %s AND (%s) ' % (
          ', '.join(['%s AS "old_%s"' % (k,k) for k,v in modified]),
          ', '.join(['%s AS "new_%s"' % (mogrify(v),k) for k,v in modified]),
          ', '.join(['%s AS "expected_%s"' % (mogrify(old[k]),k) for k,v in modified]),
          relname,
          ' AND '.join(['"%s" = %s' % (k,mogrify(v)) for k,v in old.items() if k in primary_keys]),
          ' OR '.join(['"%s"<>%s' % (k,mogrify(v)) for k,v in old.items() if k in [km for (km,vm) in modified]]),
      )
      plpy.execute(plan, [ sql ], 0)
    sql = 'UPDATE "%s" SET %s WHERE %s' % (
          relname,
          ', '.join(['"%s"=%s' % (k,mogrify(v)) for k,v in modified]),
          ' AND '.join(['"%s"=%s' % (k,mogrify(v)) for k,v in old.items() if k in primary_keys]),
        )
  else:
    sql = ""
elif event == 'DELETE':
  sql = 'DELETE FROM "%s" WHERE %s' % (
          relname,
          ' AND '.join(['"%s"=%s' % (k,mogrify(v)) for k,v in old.items() if k in primary_keys]),
        )
if sql:
  # store trigger log sql
  plpy.execute(plan, [ sql ], 0)
  r = plpy.execute('NOTIFY "replicas"', 0)
```

Figura 8 Disparador *py_log_replica.*

El script cliente se conecta a ambas bases de datos (maestra y esclava), escucha las señales NOTIFY en la base maestra, y re-ejecuta el registro de réplica en la base esclava cuando llega dicha señal. Utiliza secuencias y transacciones para evitar pérdida de datos.

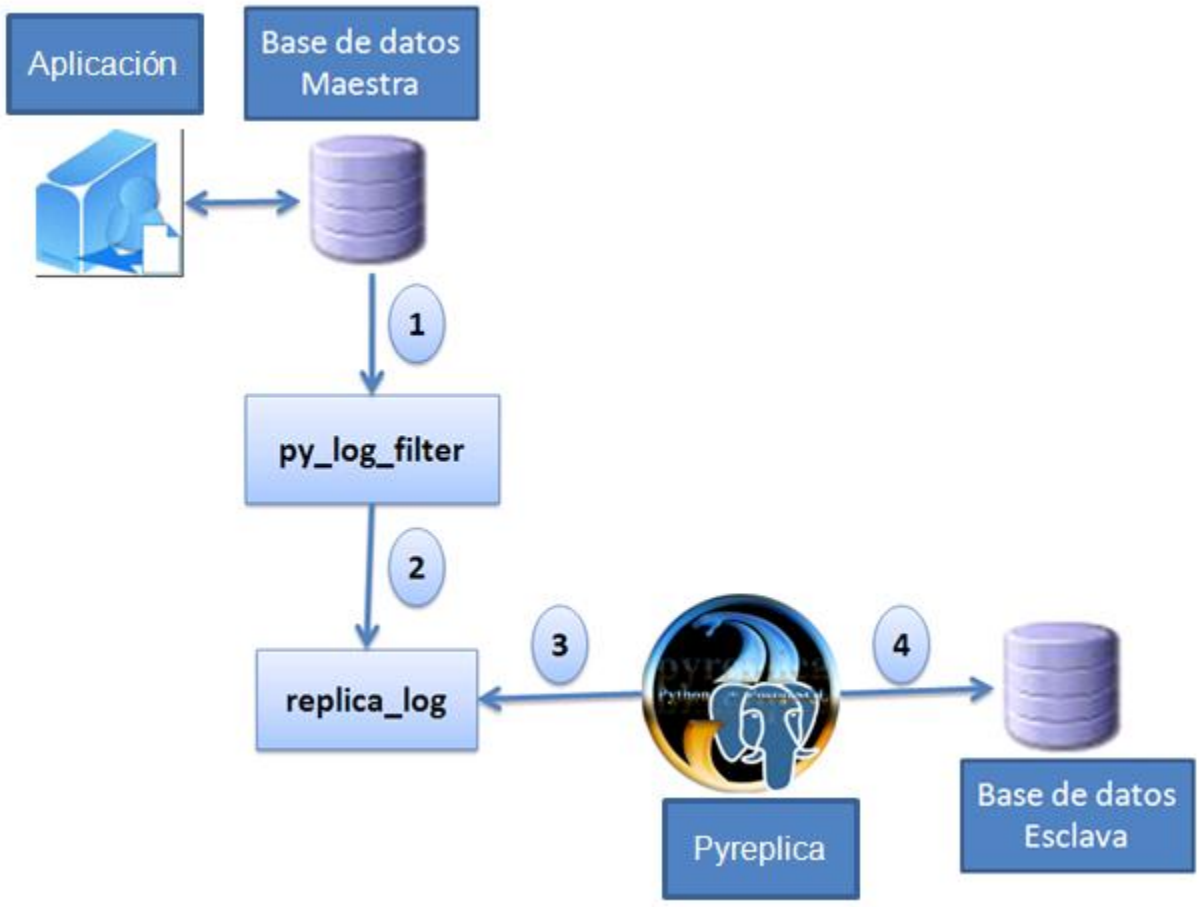


Figura 9 Automatización de la réplica maestro- esclavo

Flujo 1: El disparador detecta los cambios y verifica los nuevos valores con las reglas almacenadas en la tabla py_log_filter, si se cumplen las reglas almacena los cambios usando la clave primaria de la base de datos en la tabla replica_log. Por lo cual, si la tabla no tiene clave primaria, no puede ser replicada.

Flujo 2, 3 y 4: Como usa señales NOTIFY, la replicación es prácticamente instantánea y más eficiente (sin *polling*). Si el cliente esta caído y las señales NOTIFY se pierden, cuando el cliente se conecta nuevamente, automáticamente re-ejecuta los datos replicados perdidos.

El script demonio de replicación usa transacciones de cometer (*commit*) en dos fases (*TPC*), para asegurarse que ambos servidores se actualizaron correctamente.

Puede configurarse para enviar notificaciones vía correo electrónico (cuando el demonio de replicación inicia, se detiene o tiene algún error o advertencia)

Mediciones de rendimiento simple muestran que el disparador solo es un 50% más lento que uno basado en C (como en Slony), con los beneficios de que puede ser fácilmente instalado, portado, mantenido y adaptado.

Las relaciones en el entorno maestro-maestro son similares a este proceso solo que se va a obtener de la tabla replica_log la información que haya sido modificada por los usuarios distintos a los del maestro a donde se va a replicar la información para evitar conflictos en la replicación.

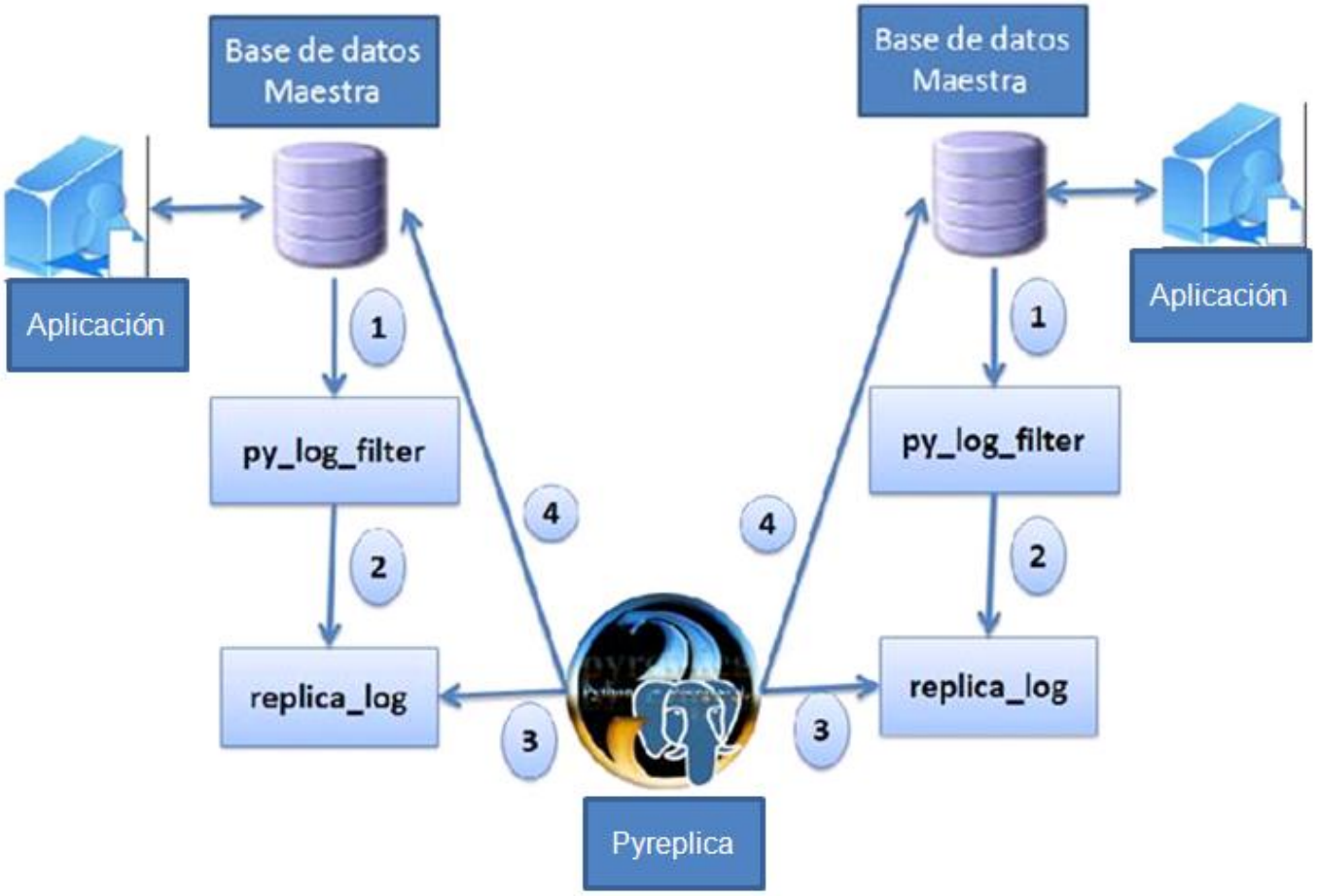


Figura 10 Automatización de la réplica maestro-maestro.

2.6 Modelo de Dominio

Teniendo en cuenta las descripciones de los procesos en el comienzo del capítulo, se determina que el negocio que se está estudiando, tiene un bajo nivel de estructuración por lo que se propone dar un enfoque a todo este proceso de actualización de la información mediante un modelo del dominio.

¿Qué es un modelo del dominio?

El modelo del dominio es un artefacto construido en la fase de concepción durante la tarea construcción del modelo del dominio, y representa conceptos de la realidad física. Se utilizan para capturar y expresar el entendimiento ganado área bajo análisis antes de diseñar un sistema. Es utilizado por el analista para comprender el sector, ya sea industrial o de negocio, al que el sistema va a servir.

Conceptos del Modelo del Dominio

Concepto	Descripción
Entidad	Representa una institución, la UCI, o cualquier FR.
Sistema	Es la unión del gestor de base de datos PostgreSQL con la información que este gestiona.
Administrador	Persona encargada del mantenimiento y gestión de la información en una entidad.

Tabla 2. 1 Conceptos del Modelo de Dominio.

Los conceptos de más relevancia durante el desarrollo de la aplicación, se relacionan entre sí y dan paso al dominio siguiente, donde se muestra la información del sistema y sus relaciones.

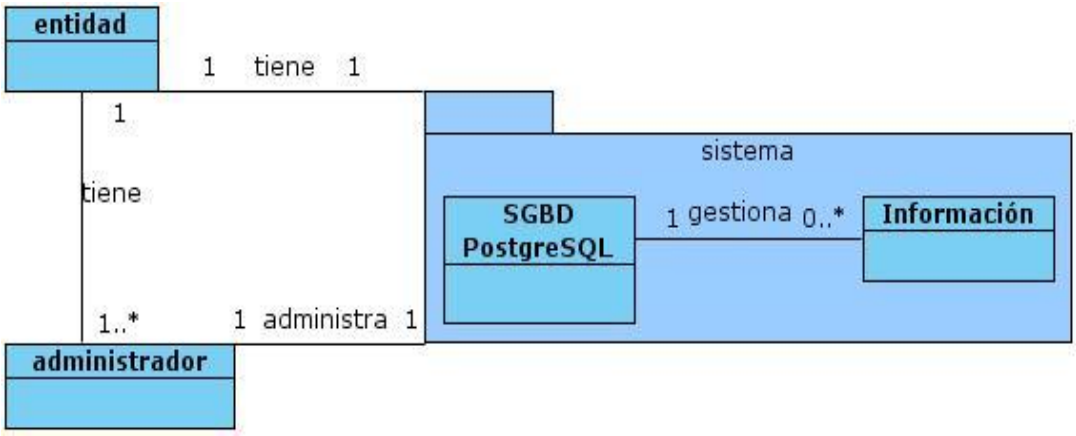


Figura 11 Modelo del dominio.

2.6.1 Descripción del modelo de dominio

Cuando nos referimos a entidad se trata de la UCI y las FR, cada una tiene un sistema compuesto por el gestor de bases de datos PostgreSQL que gestiona toda la información que se maneja en la entidad, este sistema es administrador por un administrador, donde este puede administrar varios sistemas.

2.7 Especificación de los requisitos de software

La aplicación debe cumplir una serie de aspectos desde el punto de vista funcional que son los que le dan a la misma el nivel de aceptación en sus clientes finales. Además se regirá por otros aspectos desde el punto de vista no funcional como rendimiento, usabilidad y confiabilidad.

2.7.1 Requisitos funcionales

Las funcionalidades que debe brindar la aplicación se encuentran enumeradas a continuación en orden descendente de acuerdo a la jerarquía que existe entre estas.

1. Configurar réplica maestro-esclavo

1.1. Seleccionar tipo de réplica Maestro-Esclavo.
1.2. Configurar DSN Maestro.
1.3. Configurar DSN Esclavo.
1.4. Configurar n DSN Esclavos.
1.5. Salvar configuración.
1.6. Generar archivos de configuración.

2. Configurar réplica maestro-maestro

2.1. Seleccionar tipo de réplica Maestro-Maestro.
2.2. Configurar DSN Maestro 1.
2.3. Configurar DSN Maestro 2.
2.4. Configurar n DSN Maestros.
2.5. Salvar configuración.
2.6. Generar archivos de configuración.

3. Insertar regla de replicación

3.1. Seleccionar DSN al que se le va aplicar la regla.
3.2. Seleccionar tabla.
3.3. Adicionar condición:

3.3.1. Seleccionar campo.
3.3.2. Seleccionar operador.
3.3.3. Entrar valor.
3.3.4. Agregar condición.
3.4. Repetir 3.3 n veces.
3.5. Seleccionar eventos.
3.6. Salvar regla.

4. Guardar Backup

4.1. Seleccionar DSN.
4.2. Seleccionar destino del Backup.
4.3. Entrar nombre del Backup.
4.4. Generar fichero de Backup.

5. Cargar Backup

5.1. Seleccionar DSN.
5.2. Selecciono fichero Backup creado previamente.
5.3. Restaurar DSN.

6. Generar reporte de réplica

6.1. Seleccionar fecha de inicio.
6.2. Seleccionar fecha de fin.
6.3. Seleccionar destino para el fichero de reporte.
6.4. Nombrar el fichero de reporte.
6.5. Obtener datos de réplica entre fecha de inicio y fecha de fin.
6.6. Guardar información en el fichero de reporte.

2.7.2 Requisitos no funcionales

Los requisitos no funcionales que debe cumplir la aplicación son los siguientes, en cuanto a:

Interfaz externa: Diseño sencillo e intuitivo que permita al administrador usar cómodamente la aplicación. La interfaz principal con fondo que introduzca el tema de la réplica de datos y las demás sencillas y sin colores fuertes.

Usabilidad: La aplicación debe estar concebida para un número limitado de usuarios, sólo podrá ser utilizado por aquellas personas que estén autorizadas a configurar la réplica entre los servidores de la UCI y FR.

Rendimiento: La aplicación deberá tener un nivel de respuesta aceptable, tanto para los accesos a la bases de datos, como para el proceso de réplica de datos.

Portabilidad: La aplicación es debe estar soportada en Windows y en Linux.

Seguridad: La aplicación está concebida para administradores o personas autorizadas.

Software: Debe estar programado en el lenguaje Python, para le interfaz de usuario se debe utilizar PythonCard, y el gestor de base de datos debe ser PostgreSQL versión 8.3 o superior.

Confiabilidad: Debe ser confiable en un alto grado, dado su fin de gestionar la información.

Ayuda: Debe tener un pequeño manual que permita a las personas ajenas a ella poder interactuar de la manera correcta.

Administración: Debe estar bien organizada para facilitar su administración. Alguna modificación en la programación debe hacerse en el paquete correspondiente sin afectar a los demás.

2.8 Definición de los casos de uso

2.8.1 Definición de los actores

Descripción de los actores del sistema:	
Actor	**Justificación**
Administrador	Persona especializada en el trabajo con la aplicación, es el que va a configurar la réplica y todas las otras funciones referentes a la misma.

Tabla 2. 2 Descripción de los actores del sistema.

2.8.2 Definición de los casos de uso

Caso de uso	Nombre
CU-1	Configurar Réplica
CU-2	Gestionar Reglas
CU-3	Manejar Backup
CU-4	Monitorear Réplica
CU-5	Generar Reporte

Tabla 2. 3 Definición de los casos de uso.

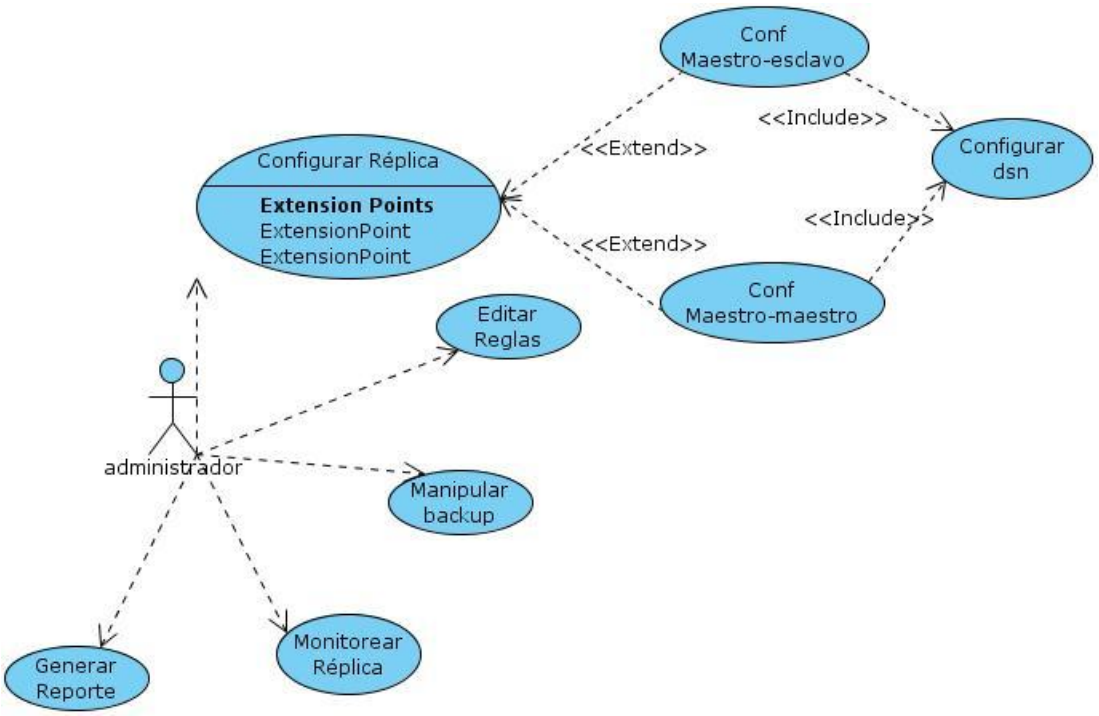


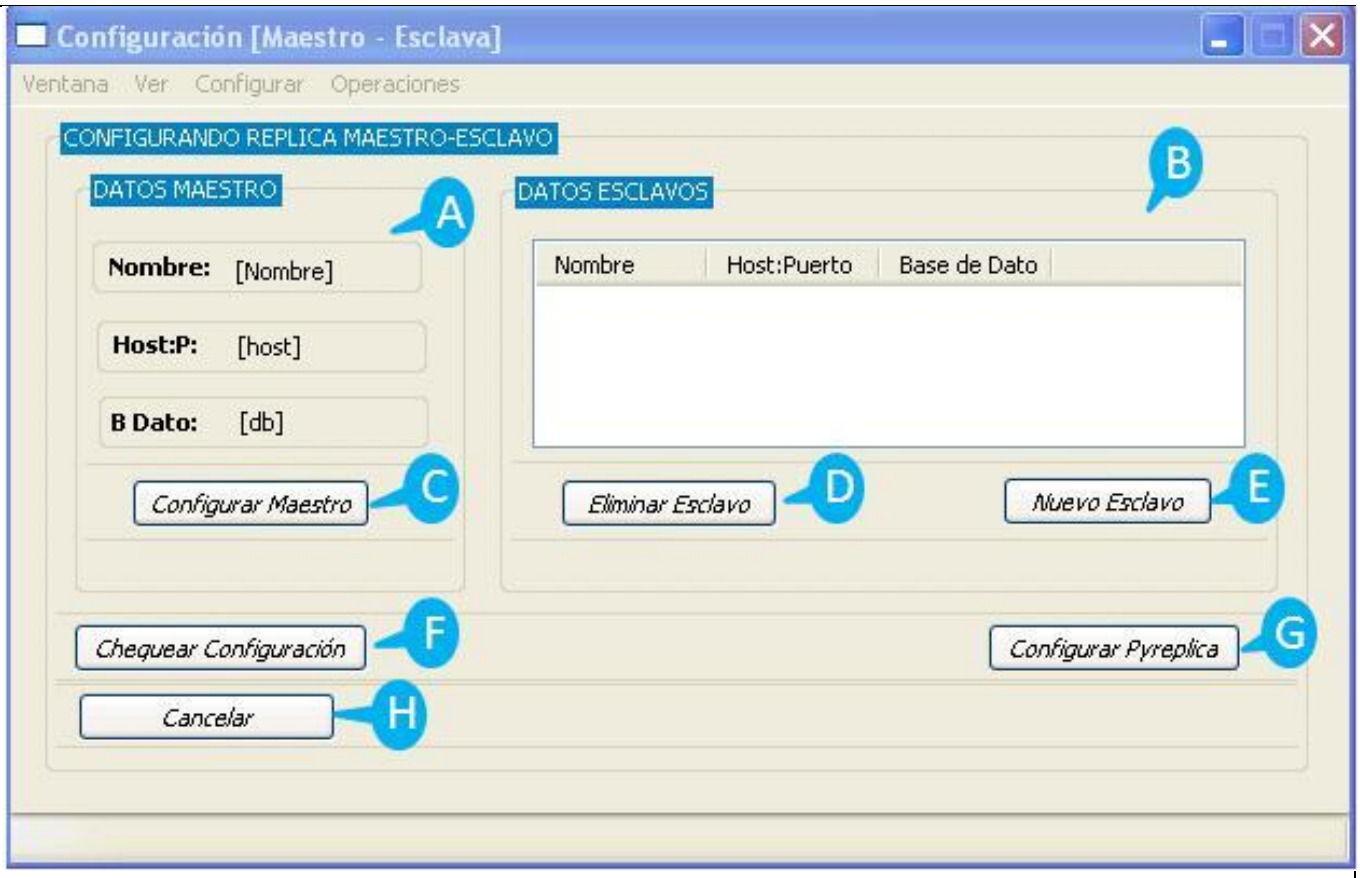
Figura 12 Diagrama de casos de uso del sistema

2.8.3 Casos de uso expandidos

2.8.3.1 Caso de Uso: Configurar Réplica

Caso de Uso	
CU - 1	Configurar Réplica
Propósito	Que la réplica quede configurada y lista para comenzar.
Actor	Administrador
Resumen	El caso de uso se inicia cuando el administrador da clic sobre la imagen 'configurar_replica' en la interfaz "principal" o selecciona la opción 'Configurar réplica' en el menú "Operaciones" de esta misma ventana.
Referencias	R1, R2

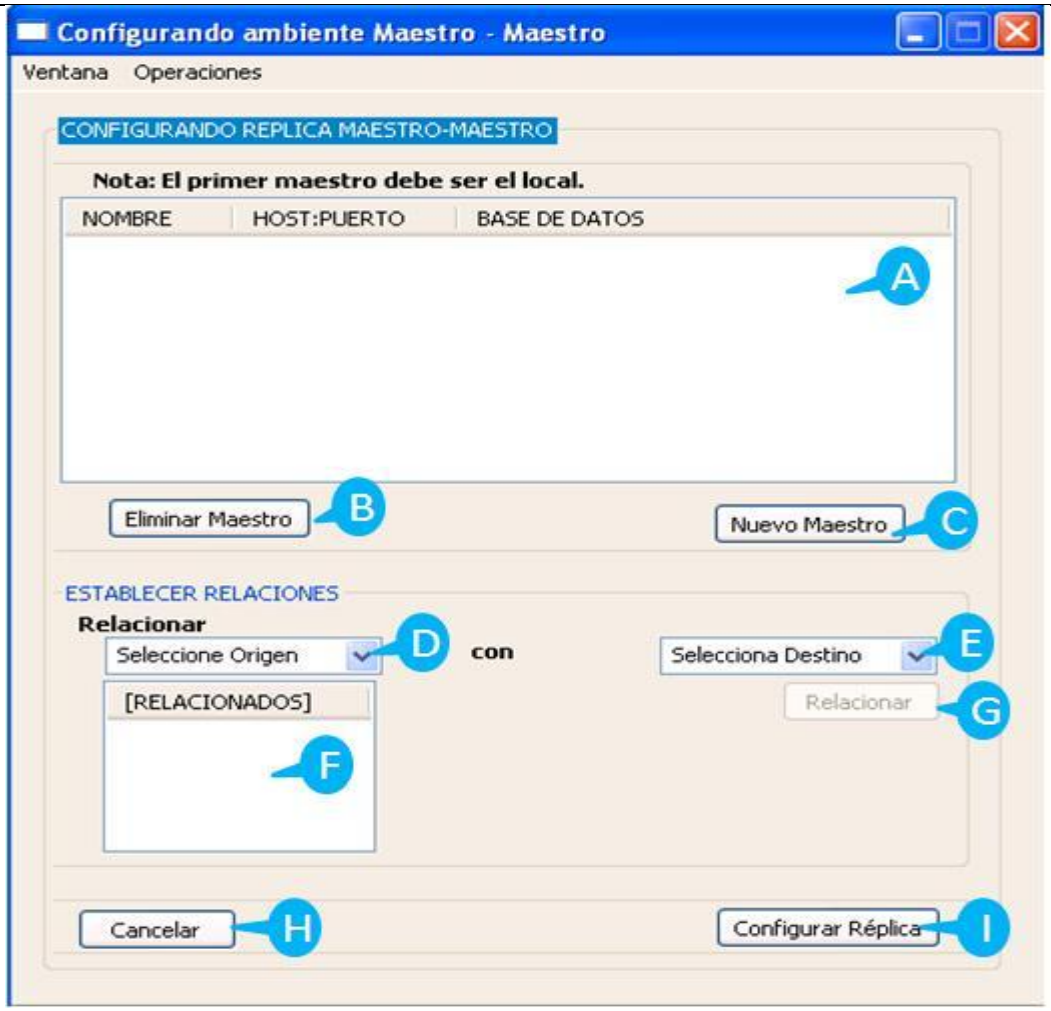
<table>
<tr><td>Pre-condiciones</td><td colspan="2">El DSN local debe haber sido configurado.</td></tr>
<tr><td colspan="3">Interfaz
 
</td></tr>
<tr><td>Descripción</td><td colspan="2">A: Radiogroup donde se selecciona el tipo de réplica a configurar.
B: Button para pasar a configurar la réplica seleccionada.
C: Button para regresar a la interfaz principal sin hacer nada.</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1. El administrador selecciona una opción en A.
2. Da clic en B.</td><td>3. Si la opción que seleccionó el administrador es ‘Maestro - Esclavo’ ver sección “Configurar Maestro-Esclavo”, si seleccionó la opción ‘Maestro - Maestro’ ver sección “Configurar Réplica Maestro-Maestro”.</td></tr>
<tr><td colspan="3">Flujos alternativos</td></tr>
<tr><td colspan="3">Sección “Configurar Réplica Maestro-Esclavo”</td></tr>
<tr><td colspan="3">Interfaz</td></tr>
</table>



Descripción	**A**: GroupBox donde se muestra la información del DSN Maestro. **B**: GroupBox donde se muestra la información de los DSN Esclavos. **C**: Button para Configurar el DSN Maestro. **D**: Button para eliminar un DSN Esclavo. **E**: Button para adicionar un DSN Esclavo. **F**: Button para verificar si hay alguna configuración guardada. **G**: Button para Configurar la réplica. **H**: Button para regresar a la interfaz 'Tipo de Réplica' sin hacer nada.

Acción del actor	**Respuesta del sistema**
	1. Si no se ha configurado el DSN local el sistema muestra la ventana 'Configurar DSN' y el administrador sigue los pasos de la sección "Configurar DSN".
1 El administrador da clic en **C** para configurar el	2 El sistema muestra la interfaz para configurar un

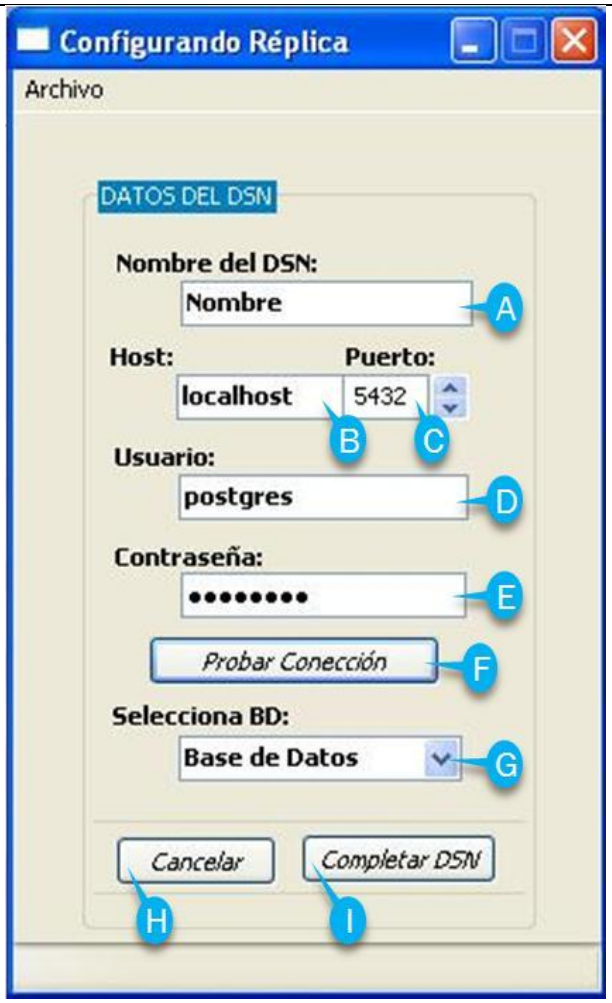
	DSN Maestro.		DSN.
3 4	El administrador sigue los pasos de la sección “Configurar DSN”. El actor da clic en **E** para adicionar un DSN Esclavo a la configuración.	5	El sistema muestra la interfaz para configurar un DSN.
6 7	El administrador sigue los pasos de la sección “Configurar DSN”. El actor da clic en **G**.	8	El sistema pide al administrador que rectifique los datos antes de guardarlos antes de salvar la configuración
9	El administrador da clic en el botón ‘**OK**’ del mensaje.	10	El sistema guarda la configuración en la base de datos postgres y avisa al administrador de que se guardaron.
11	El administrador da clic en el **I**.	12	El sistema prepara al maestro de la réplica y este queda listo para la misma y termina el caso de uso.
Sección “Configurar Réplica Maestro-Maestro”			
Interfaz			



Descripción	**A**: MultiColumnList para mostrar la información de los maestros configurados. **B**: Button para eliminar un maestro de la configuración. **C**: Button para adicionar un nuevo maestro a la configuración. **D**: Combobox para seleccionar el maestro origen en una relación. **E**: Combobox para seleccionar el maestro destino en una relación. **F:** MultiColumnList para mostrar los maestros relacionados a un maestro. **G**: Button para establecer una relación entre los maestros seleccionados en D y E. **H**: Button para salir sin hacer nada. **I**: Button para configurar la réplica maestro-maestro.

Acción del actor	**Respuesta del sistema**
	1 Si no se ha configurado el DSN local el sistema

	muestra la ventana ‘Configurar DSN’ y el administrador sigue los pasos de la sección “Configurar DSN”.
2. El administrador da clic en **C** para adicionar un maestro cada vez que lo desee.	3. El sistema muestra la interfaz configurar_dsn.
4. El administrador sigue los pasos de la sección ‘Configurar DSN’. 5. Seleccionar la opción “Mostrar“ en el menú Operaciones.	6. El sistema muestra la información de los maestros en **A**. 7. Edita **D** con los nombres de los maestros.
8. Selecciona un maestro en **D**.	9. Edita **E** con el nombre de los maestros restantes.
10. Da clic en **G**.	11. El sistema adiciona al maestro seleccionado en **E** a la lista de maestros del seleccionado en **D** y lo muestra en **F**.
12. El administrador realiza los pasos desde 7 hasta 9 **n** veces para terminar la configuración. 13. Da clic en **H**.	14. El sistema guarda la configuración y crea los ficheros de configuración para cada relación maestro-maestro y termina el caso de uso.
Sección “Configurar DSN”	
Interfaz	

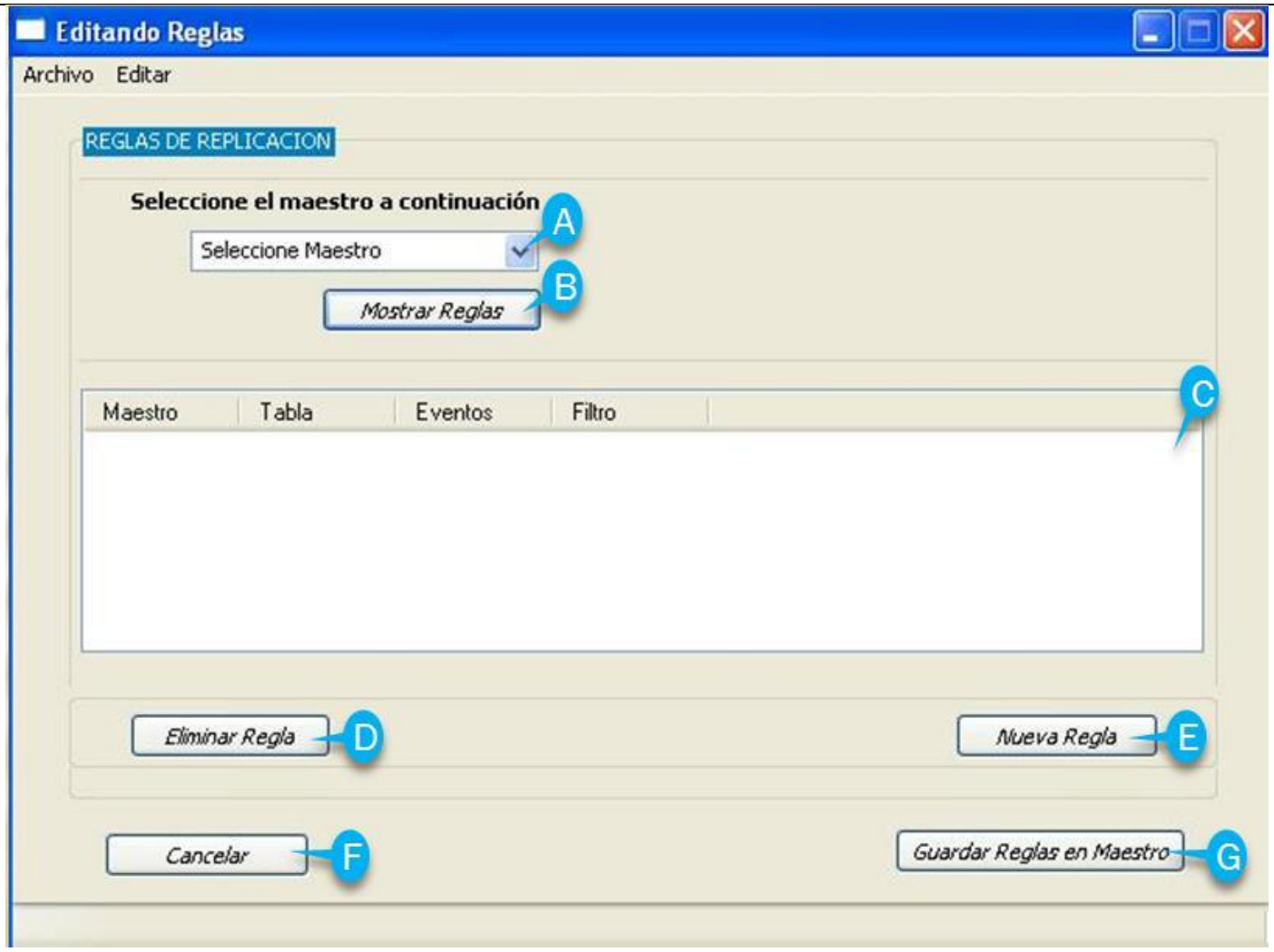
<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Descripción</td><td>A: TextField para entrar el nombre del DSN.
B: TextField para entrar la dirección IP del DSN.
C: Spinner para especificar el puerto para la conexión.
D: TextField para entrar el usuario para la conexión.
E: PasswordField para entrar la contraseña para la conexión.
F: Button para verificar si los datos son válidos.
G: Combobox para Seleccionar la base de datos a la que se quiere conectar.
H: Button para regresar a la interfaz padre sin hacer nada.
I: Button para completar los datos del DSN y proponer guardarlo.</td></tr>
<tr><td>Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td>2 El administrador entra la siguiente</td><td>4 El sistema se conecta al servidor Postgres</td></tr>
</table>

información del DSN: nombre, dirección IP, puerto, usuario y contraseña, en **A**, **B**, **C**, **D**, **E**, respectivamente. 3 El administrador da clic en F.	especificado. 5 Obtiene los nombres de las Bases de datos existentes. 6 Las lista en **G**.
7 El administrador selecciona la base de datos en **G**. 8 Da clic en **I**.	9 El sistema verifica que no se han producido cambios en los datos, adiciona la base de datos seleccionada a la configuración del DSN y pide al actor que verifique los datos antes de guardarlo.
10 El administrador da clic en el botón '**OK**' del mensaje mostrado.	11 El sistema guarda el DSN configurado y cierra la interfaz terminando así el caso de uso.
Post-condiciones	Queda configurada la réplica y lista para comenzar con el proceso de replicación.

Tabla 2. 4 Descripción del CU Configurar réplica.

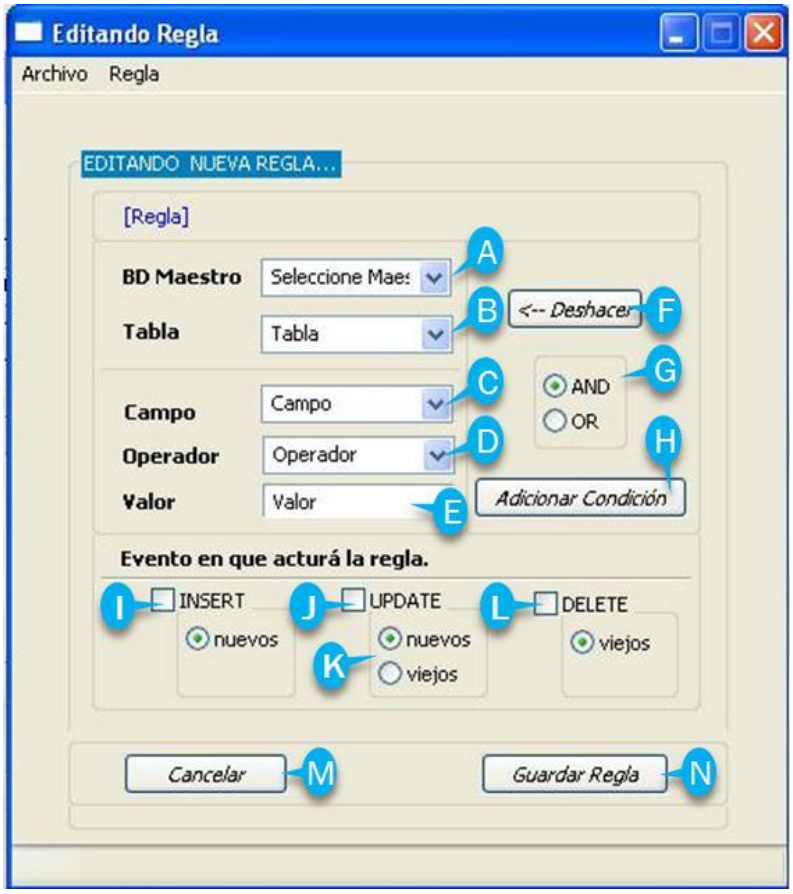
2.8.3.2 Caso de Uso: Gestionar Reglas

Caso de Uso	
CU - 2	Gestionar Reglas
Propósito	Que las reglas para la replicación queden configuradas.
Actor	Administrador
Resumen	El caso de uso se inicia cuando el administrador da clic sobre la imagen 'editar_reglas' o selecciona la opción 'Editar Reglas' en el menú "Operaciones" de la interfaz principal.
Referencias	R3
Pre-condiciones	El DSN local debe haber sido configurado.
Interfaz	



Descripción	**A**: Combobox para seleccionar el DSN maestro donde se van a editar las reglas. **B**: Button para mostrar las reglas que existen en el DSN Maestro seleccionado en A. **C**: MultiColumnList donde se van a mostrar las reglas. **D**: Button para pasar a eliminar una regla. **E**: Button para pasar a editar una nueva regla. **F**: Button para salir a la interfaz principal sin hacer nada. **G**: Button para guardar todas las reglas en el DSN Maestro seleccionad en A.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona en **A** él DSN Maestro donde se van a editar las reglas.	3. El sistema muestra todas las reglas que hay configuradas en el maestro seleccionado.

2. Da clic en B para mostrar las reglas que existen en ese momento en el maestro seleccionado.	
4. El administrador da clic en **D** o en **E**.	5. Si dio clic en **D** ir a sección “Eliminar Regla” si no ir a sección “Insertar Regla”.
Flujos alternativos	
Sección “Insertar Regla”	
Interfaz	

Descripción	**A**: Combobox para seleccionar el DSN Maestro al que pertenecerá la nueva regla. **B:** Combobox para seleccionar la tabla para la cual se creará la regla. **C**: Combobox para seleccionar el campo de la tabla seleccionada en B. **D**: Combobox para seleccionar el operador. **E:** TextField para especificar el valor. **F:** Button para deshacer la última operación que realicé sobre la regla.

	G: Radiogroup para seleccionar la forma de unión entre la última condición adicionada y la nueva. **H**: Button para adicionar una nueva condición. **I**: Checkbox para especificar si la nueva regla se va a aplicar al evento INSERT. **J**: Checkbox para especificar si la nueva regla se va a aplicar el evento UPDATE. **K**: Radiogroup para seleccionar a que valores se le aplica el filtro cuando ocurra un UPDATE. **L**: Checkbox para especificar si la nueva regla se va a aplicar el evento DELETE. **M**: Button para salir de la ventana sin hacer nada. **N**: Button para guardar la nueva regla.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona el DSN Maestro en **A**.	2. El sistema edita **B** con los nombres de las tablas que existen en el DSN Maestro seleccionado.
3. El administrador selecciona la tabla en **B**.	4. El sistema edita **C** con los nombres de los campos de la tabla seleccionada en B.
5. El administrador selecciona un campo en **C**.	6. El sistema edita a **D** con los operadores para especificar una condición.
7. El administrador selecciona un operador en **D**. 8. Entra un valor en **E** para la condición. 9. Da clic en **H** para adicionar la condición a la regla.	10. El sistema adiciona la condición a la regla e informa al administrador que se realizó la operación satisfactoriamente y muestra la **O**.
11. El administrador marca **I** (y\o) **J** (y\o) **L** para seleccionar los eventos en los que actuará la regla. 12. Da clic en **N** para guardar la regla.	13. El sistema adiciona la regla. 14. Informa al administrador y pregunta si va a editar otra regla.
15. El administrador da clic en el botón '**Cancel**' del mensaje.	16. Se cierra la interfaz "Nueva Regla" culminando así el caso de uso.
Sección "Eliminar Regla"	
Acción del actor	**Respuesta del sistema**

Tabla 2. 5 Descripción del CU Gestionar Reglas

2.8.3.3 Manejar Backup

Caso de Uso	
CU - 3	Manejar Backup
Propósito	Hacer un respaldo de la base de datos o restaurar una a partir de un respaldo previamente creado.
Actor	Administrador
Resumen	Se inicia cuando el administrador da clic en la imagen 'manipular_backup' o en la opción "Manipular Backup" en la interfaz principal.
Referencias	R4, R5
Pre-condiciones	• El DSN local debe estar configurado. • La configuración de cualquier tipo de replica debe estar completa.
Interfaz	

Descripción	**A**: Radiogroup para seleccionar la operación a realizar. **B**: Combobox para seleccionar el DSN a respaldar o a restaurar. **C**: Button para especificar la ubicación del fichero .Backup. **D**: TextArea donde se muestra la salida del pg_dump al realizar la operación. **E**: Button para regresar a la interfaz principal sin hacer nada. **F**: Button para realizar la operación seleccionada en A.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona una opción en **A**.	2. Si selecciona la opción “Guardar Backup” ir a sección “Guardar Backup”, y si selecciona “Cargar Backup” ir a sección “Cargar Backup”.
Flujos alternativos	

Sección “Guardar Backup”	
Acción del actor	**Respuesta del sistema**
	1. El sistema cambia los textos guías de la interfaz especificando que selecciona la base de datos a guardar y el destino del fichero de respaldo. Muestra en **F** ‘Guardar Backup’.
2. El administrador selecciona en **B** el DSN al que le hará el respaldo. 3. Selecciona la ubicación donde va a guardar el fichero. 4. Nombre el fichero a guardar. 5. Da clic en **F**.	6. El sistema pasa los parámetros al pg_dump del Postgres y se realiza la operación, al finalizar se muestra la salida en **D**. 7. Informa al administrador que se ha realizado el respaldo correctamente y culmina el caso de uso.
Sección “Cargar Backup”	
Acción del actor	**Respuesta del sistema**
	1. El sistema cambia los textos guías de la interfaz especificando que selecciona la base de datos a restaurar y la ubicación del fichero de respaldo a cargar. Muestra en **F** ‘Cargar Backup’.
2. El administrador selecciona en **B** el DSN al que va a restaurar. 3. Selecciona el fichero de respaldo que va a cargar. 4. Da clic en **F**.	5. El sistema llama al pg_dump con los parámetros especificados y se realiza la operación mostrando la salida en **D** al finalizar la misma. Se informa al administrador que se ha realizado la restauración y termina el caso de uso.
Post-condiciones	Se ha hecho un respaldo de una base de datos o se ha restaurado una a partir de un fichero de respaldo.

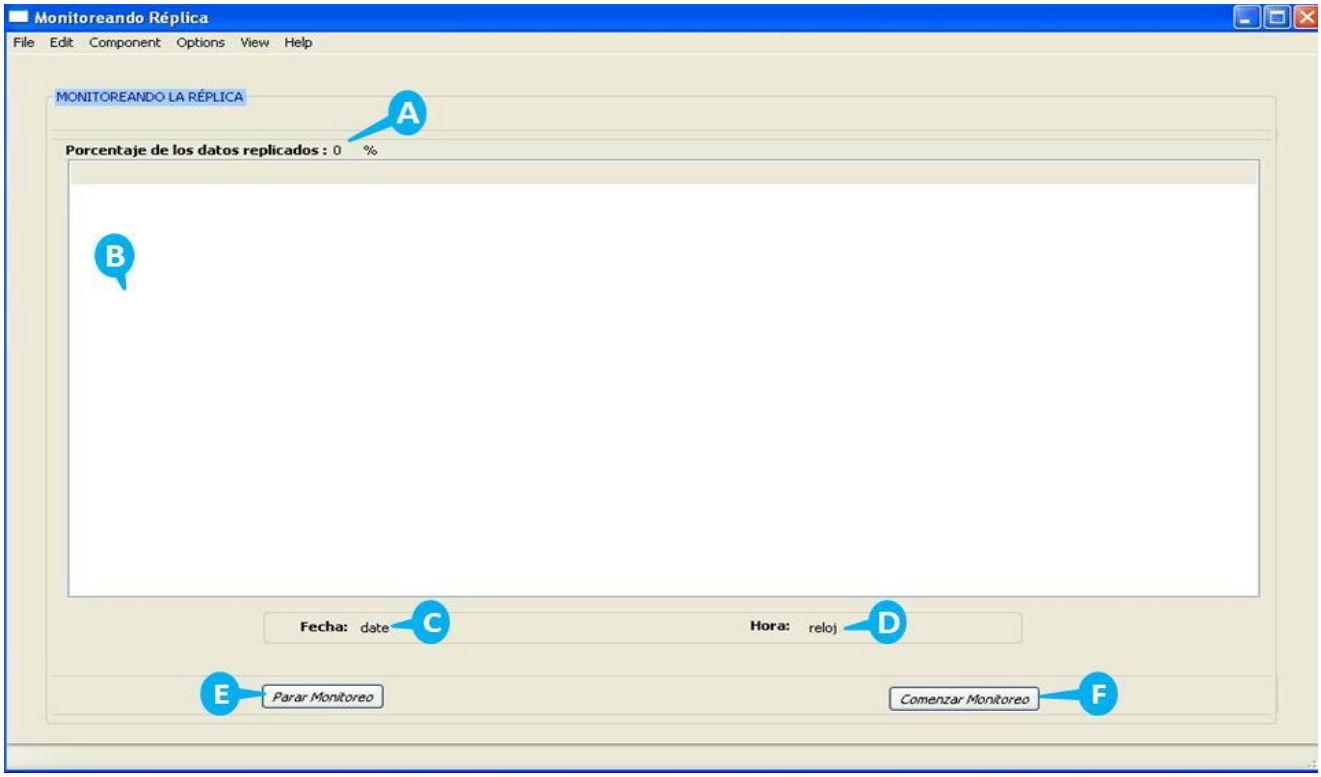
Tabla 2. 6 Descripción del CU manejar Backup

2.8.3.4 Monitorear Réplica

Caso de Uso	
CU - 4	Monitorear Réplica
Propósito	Ver en tiempo real la situación de los datos que están siendo replicados.
Actor	Administrador

Resumen	El caso de uso se inicia cuando el administrador da clic sobre la imagen 'monitoreo' o en la opción "Monitorear" del menú 'Operaciones' en la interfaz principal.
Referencias	
Pre-condiciones	El DSN local debe estar configurado y la configuración estar completa.

Interfaz



Descripción	**A**: StaticText para mostrar el por ciento de datos replicados. **B**: MultiColumnList donde se muestran las operaciones y los esclavos a los que se ha replicado la misma y a los que no. **C**: StaticText donde se muestra la fecha. **D**: StaticText donde se muestra la hora (Hora: Minutos: Segundos). **E**: Button para parar la monitorización. **F**: Button para comenzar la monitorización.

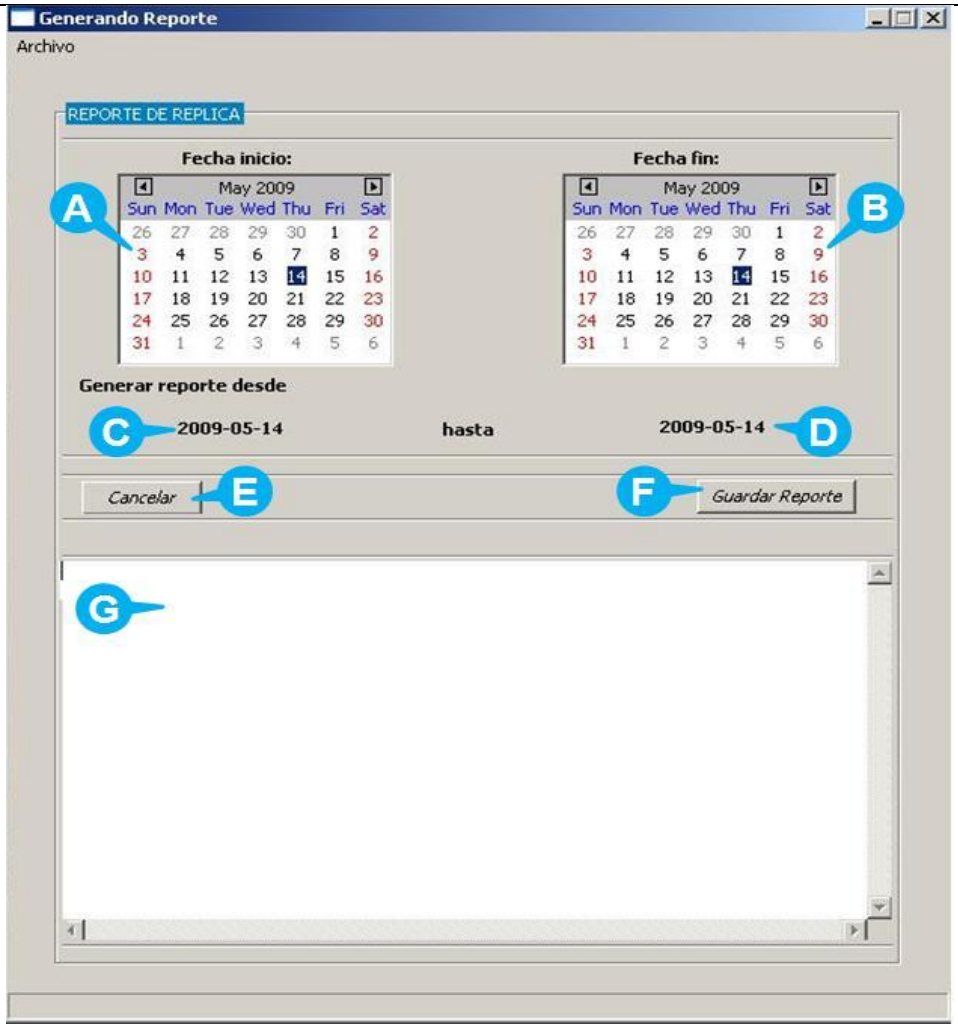
Acción del actor	**Respuesta del sistema**
	1. El sistema actualiza cada un segundo los datos desde la tabla replica-log. Y Muestra el porciento de datos

	replicados en **A**.
2. El administrador da clic en **E**.	3. El sistema deja de actualizar los datos.
4. El administrador da clic en **F**.	5. El sistema comienza la actualización periódica de la información. Cuando se han replicado todas las operaciones culmina la actualización y con esta el caso de uso.
Post-Condiciones	Se han podido monitorear los datos que deben ser replicados hacia cada uno de los sitios que forman parte de la configuración.

Tabla 2. 7 Descripción del CU monitorear replica

2.8.3.5 Generar reporte

Caso de Uso	
CU - 5	Generar Reporte
Propósito	Guardar un fichero donde muestre los datos que han sido replicados y cuáles no a partir de una fecha de inicio y una fecha de fin.
Actor	Administrador
Resumen	El caso de uso se inicia cuando el administrador da clic sobre la imagen ‘reporte o en la opción “Generar Reporte” del menú ‘Operaciones’ en la interfaz principal.
Referencias	R6
Pre-condiciones	El DSN local debe estar configurado y la configuración estar completa.
Interfaz	



Descripción	**A**: Calendar donde se selecciona la fecha de inicio del reporte. **B**: Calendar donde se selecciona la fecha de fin del reporte. **C**: StaticText donde se muestra la fecha seleccionada en A. **D**: StaticText donde se muestra la fecha seleccionada en B. **E**: Button para regresar a la interfaz principal sin hacer nada. **F**: Button para generar el fichero con los datos de la réplica en el intervalo de tiempo seleccionado. **G**: TextArea donde se muestra el reporte generado.

Acción del actor	**Respuesta del sistema**
1. El administrador selecciona una fecha de inicio en **A**.	2. El sistema la muestra en **C**.

3. El administrador selecciona una fecha de fin de **B**.	4. El sistema la muestra en **D**.
5. El administrador da clic en **F**.	6. Si la fecha de inicio es menor o igual a la de fin muestra el diálogo para seleccionar el destino del fichero de reporte.
7. El administrador selecciona la ubicación del fichero y especifica el nombre del mismo.	8. El sistema busca la información de la réplica en el intervalo seleccionado y escribe el fichero con esta información. 9. Pregunta al administrador que si desea ver el reporte generado.
10. El administrador selecciona una opción en el mensaje mostrado.	11. Si selecciona '**OK**' el sistema muestra la información del reporte en **G** y termina el caso de uso, si selecciona 'Cancel' el sistema termina el caso de uso.
Post-condiciones	Se ha creado un fichero con la información de la réplica en el intervalo que fue seleccionado.

Tabla 2. 8 Descripción del CU generar reportes

2.8.4 Conclusiones

En este capítulo se propuso el desarrollo de una propuesta de solución a partir del estudio realizado de la situación problémica. Se definieron los requisitos que debe cumplir el sistema en toda su totalidad, así como los actores del mismo, incluyendo la descripción de cada una de sus funcionalidades. Partiendo de todas las características que se han expuesto, se está en condiciones de construir el sistema siguiendo sus especificidades.

Capítulo 3. Análisis y diseño del sistema

3.1 Introducción

En el presente capítulo se describen los pasos que se llevaron a cabo en el análisis y el diseño de la aplicación siguiendo específicamente los pasos establecidos en la metodología de desarrollo seleccionada, lo que será de gran ayuda para la comprensión del sistema propuesto. Se caracterizan los patrones utilizados en el diseño tanto en la descripción de los casos de uso como en la implementación de las clases.

3.2 Análisis

3.2.1 Diagrama de clases del análisis

Las clases de análisis están centradas en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Dichas clases tienen atributos y existen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos entre ellas. Estas clases se clasifican en:

- **Entidad:** Modelan información que posee larga vida y que es a menudo persistente.
- **Interfaz:** Modelan la interacción entre el sistema y sus actores.
- **Control**: Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

El diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema, como parte del mundo real, no desde el punto de vista de la construcción del sistema.
A continuación se relacionan los diagramas de clases de análisis, asociados al problema en cuestión:

CU Configurar Réplica

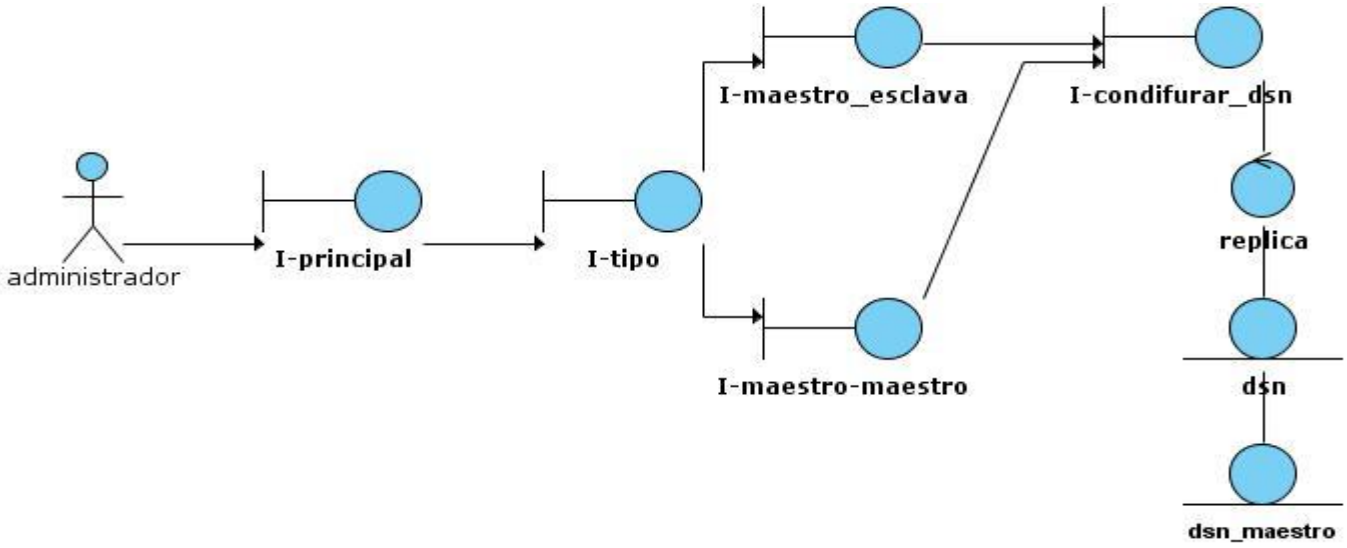


Figura 13 Diagrama de clases del análisis CU Configurar Réplica.

CU Editar Reglas

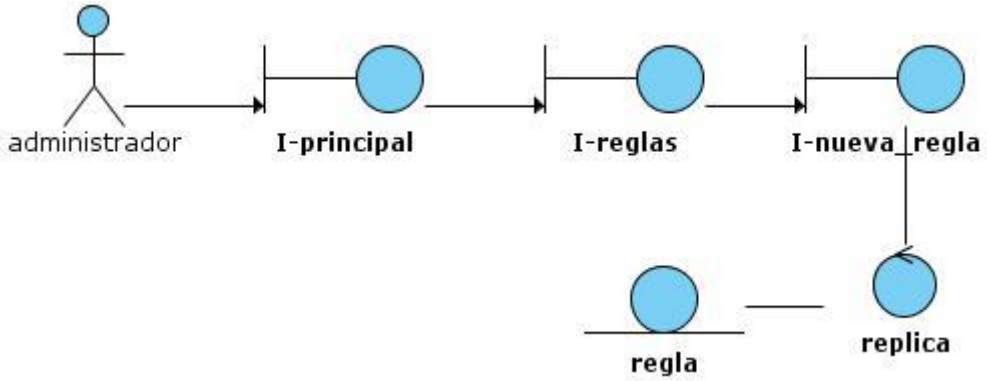


Figura 14 Diagrama de clases del análisis CU Editar Reglas

CU Manipular Backup

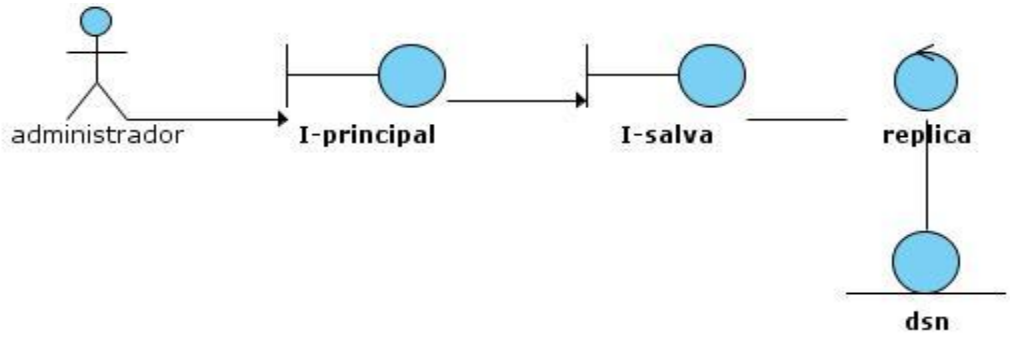


Figura 15 Diagrama de clases del análisis CU Manipular Backup

CU Monitorear Réplica

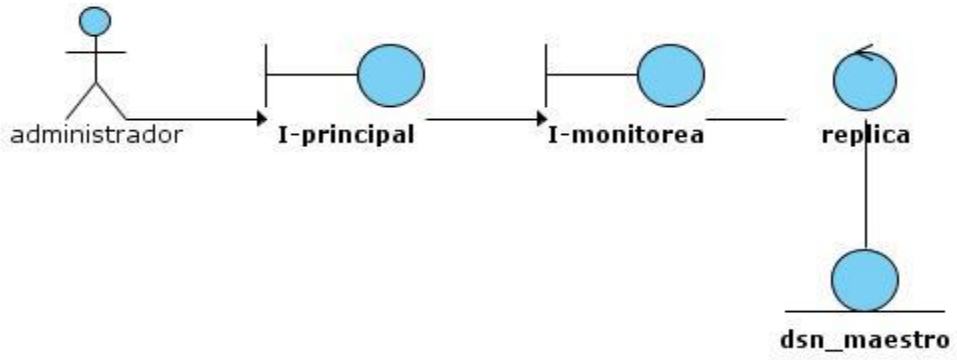


Figura 16 Diagrama de clases del análisis CU Monitorear Réplica

CU Generar Reporte

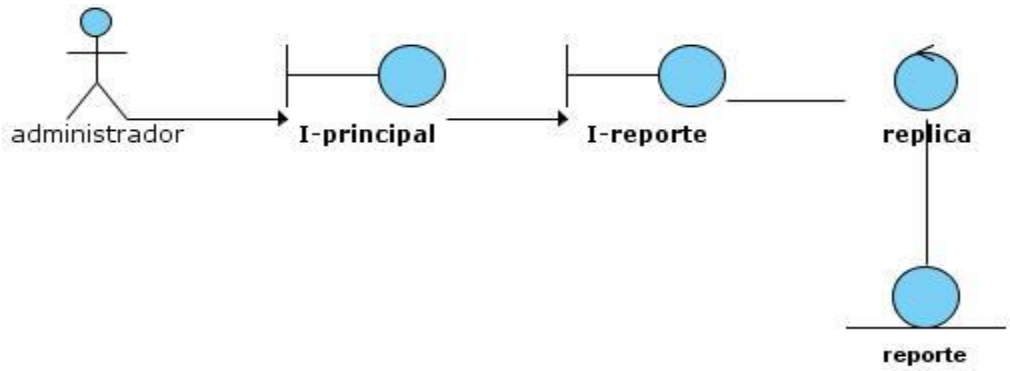


Figura 17 Diagrama de clases del análisis CU Generar Reporte

3.3 Diseño

3.3.1 Diagrama de clases

Un diagrama de clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. La definición de clase incluye definiciones para atributos y operaciones. Los diagramas de clases por definición son estáticos, esto es, representan qué partes interactúan entre sí.

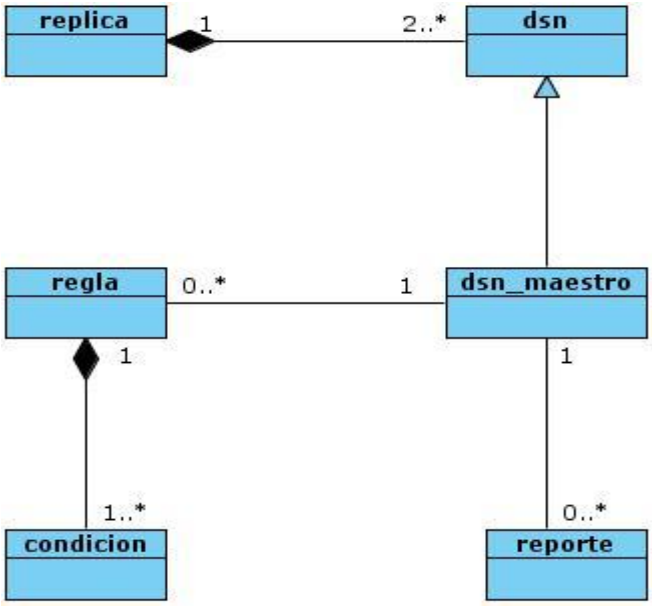


Figura 18 Diagrama de clase del diseño

Expansión de las clases del diseño

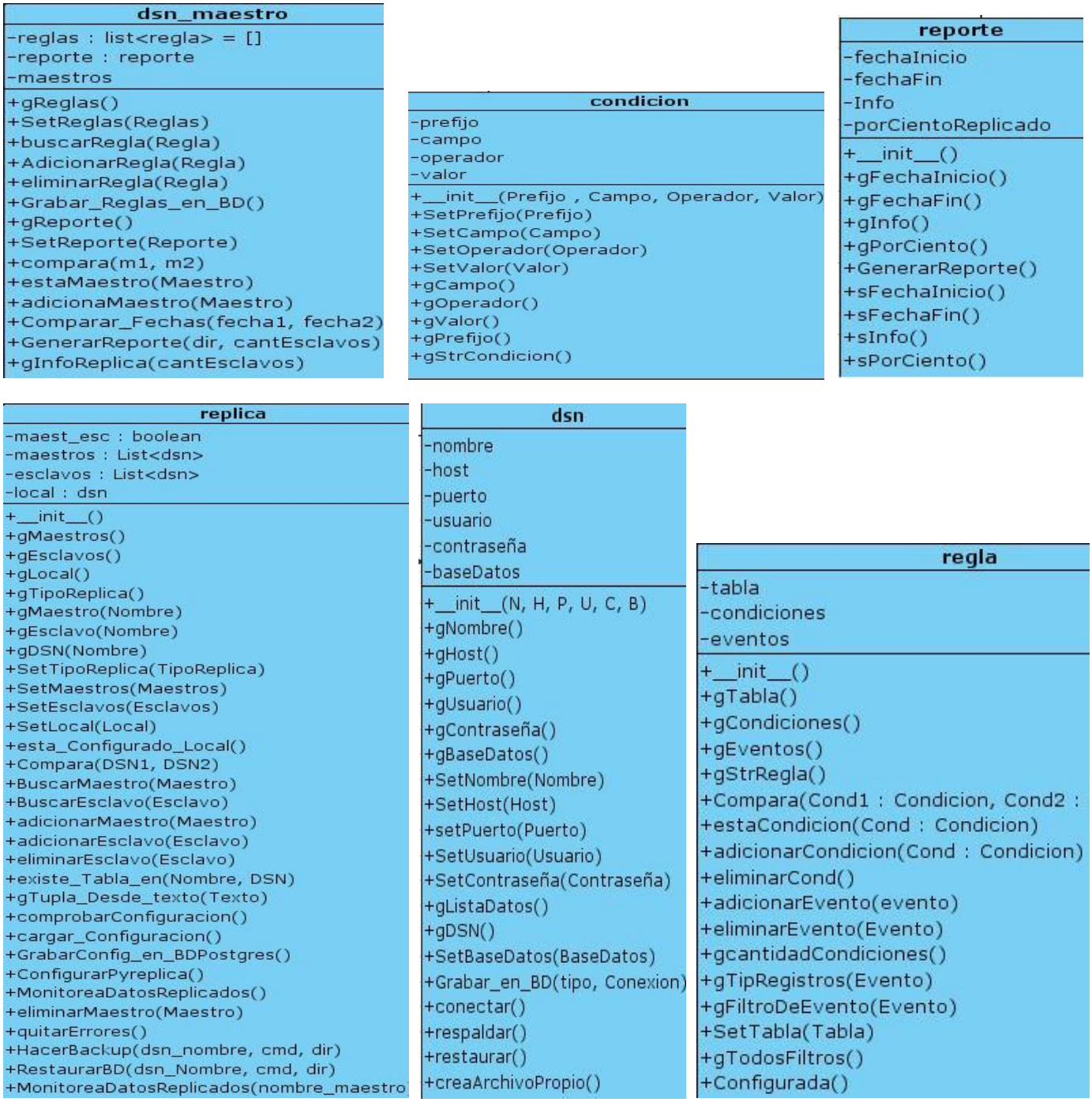


Figura 19 Expansión de las clases del diseño

3.3.2 Diagramas de interacción

Los diagramas de interacción explican gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas. Dentro de estos diagramas están los de secuencia y los de colaboración. Para representar gráficamente ésta interacción se escogió el diagrama de colaboración para una mayor claridad.

Diagramas de Colaboración

El Diagrama de Colaboración presenta una alternativa al diagrama de secuencia para modelar interacciones entre objetos en el sistema. Mientras que el diagrama de secuencia se centra en la secuencia cronológica del escenario que estamos modelando, el diagrama de colaboración se centra en estudiar todos los efectos de un objeto dado durante un escenario.

Los diagramas de colaboración para cada caso de uso se muestran a continuación:

CU Monitorear Réplica

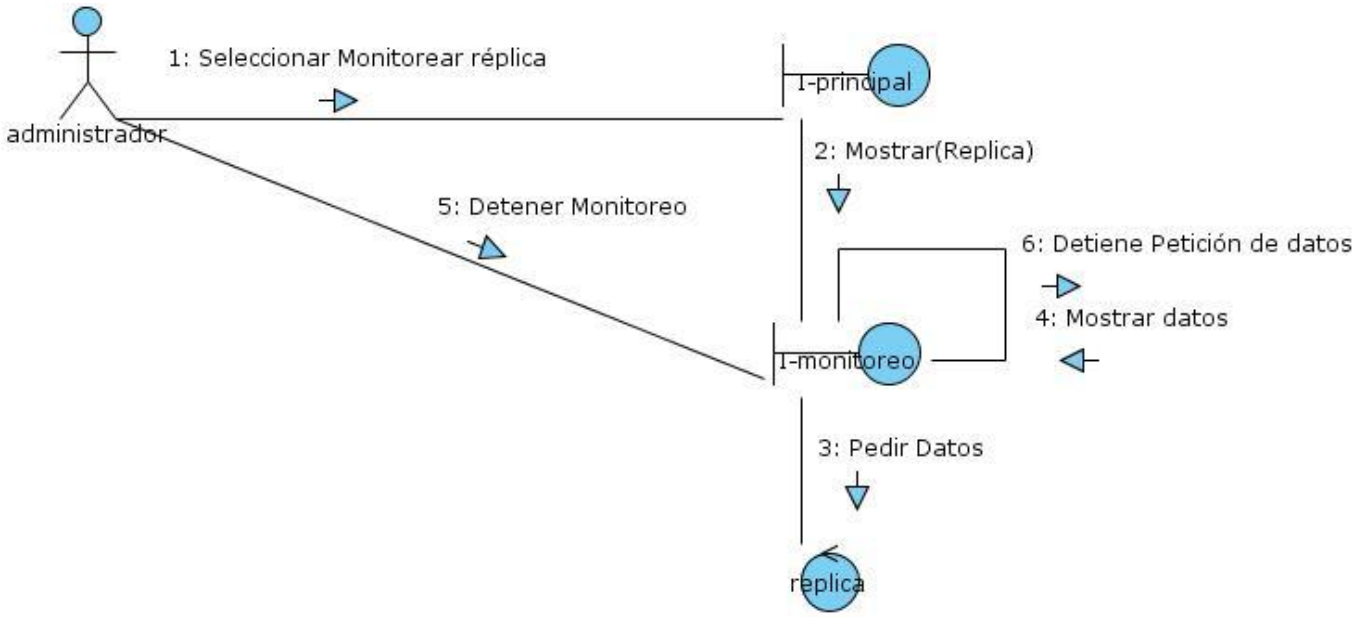


Figura 20 Diagrama de Colaboración CU Monitorear Réplica

CU Configurar Réplica

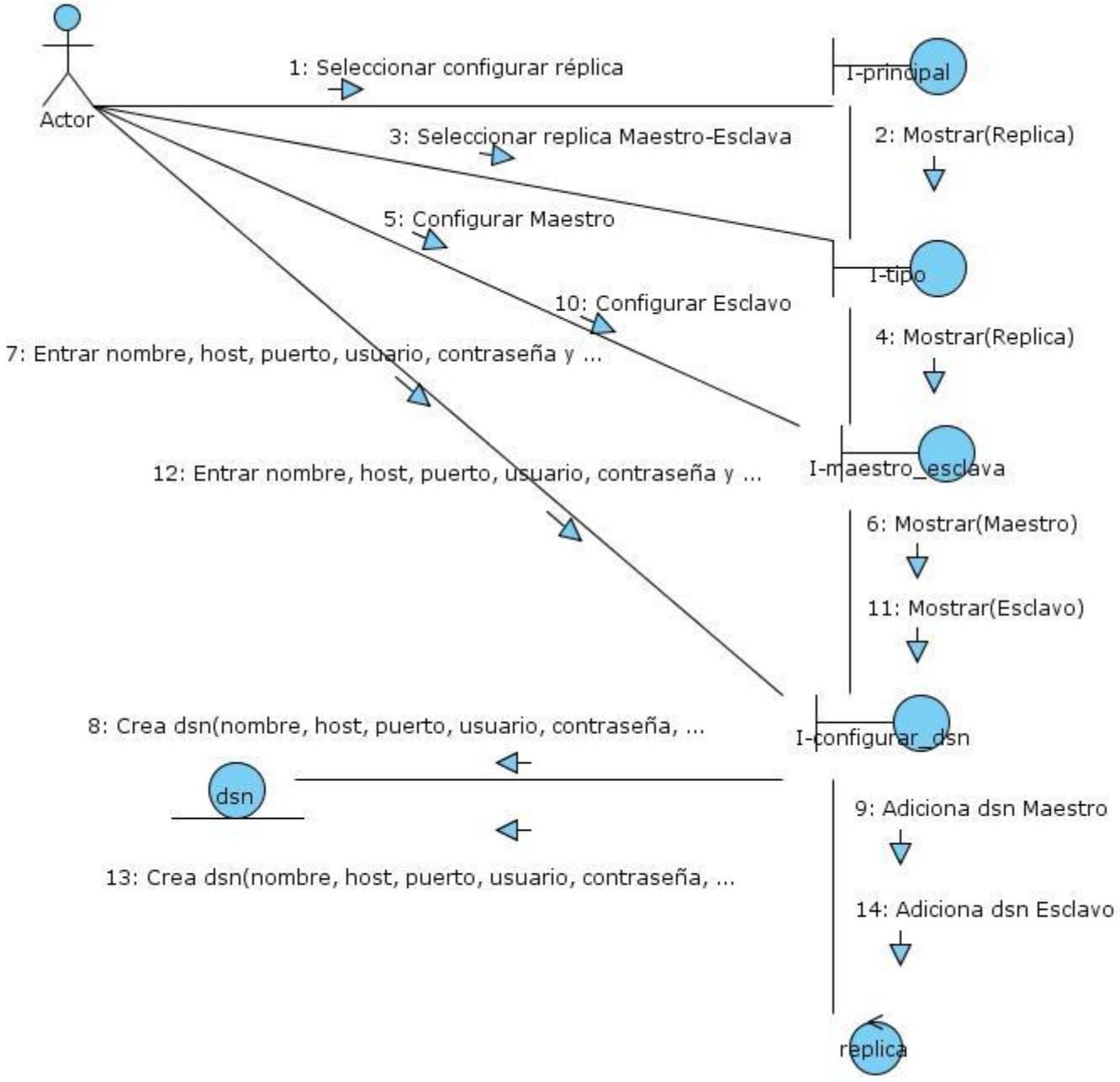


Figura 21 Diagrama de Colaboración CU Configurar Réplica (Configuración Maestro-Esclava)

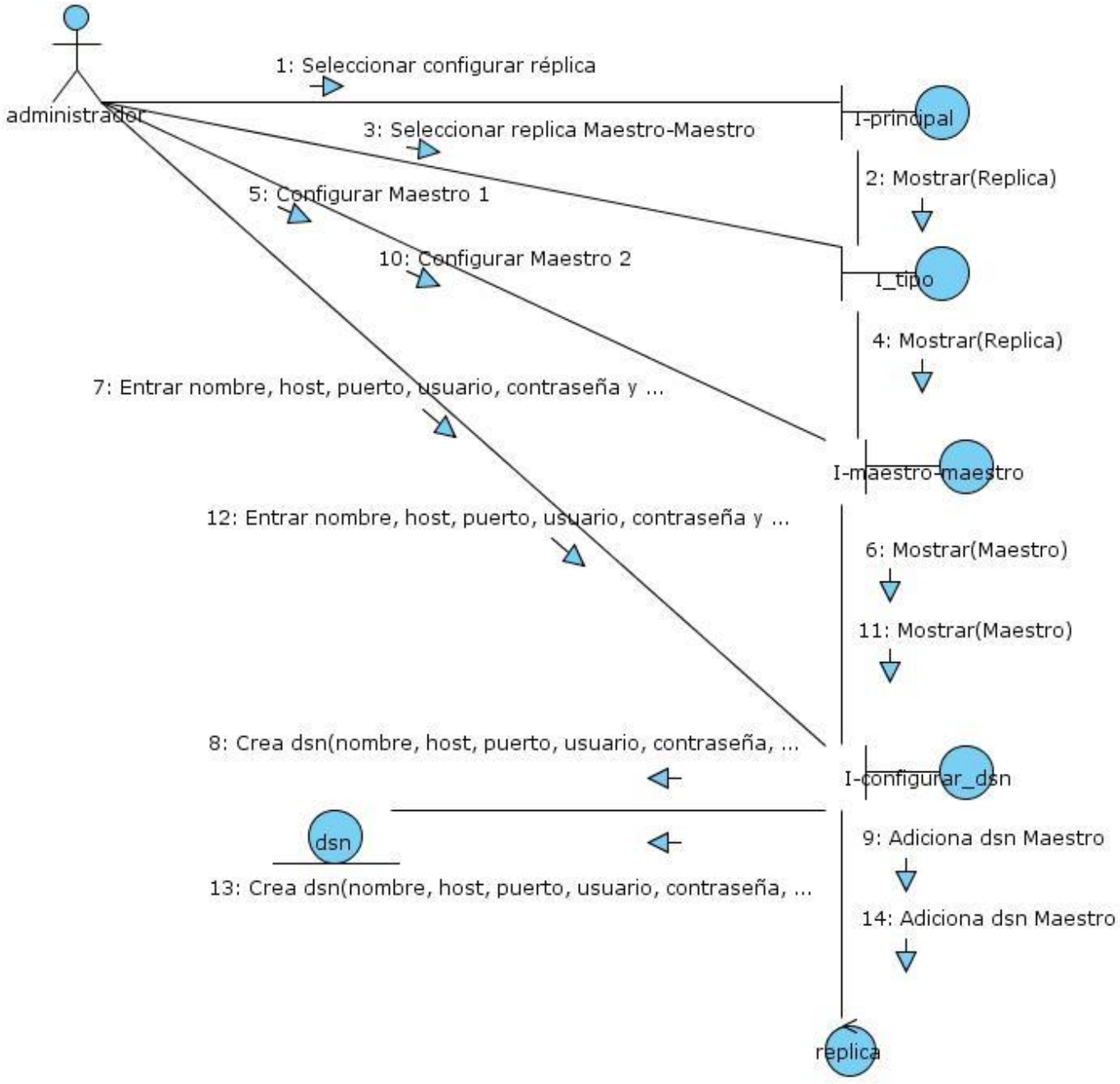


Figura 22 Diagrama de Colaboración CU Configurar Réplica (Configuración Maestro-Maestro).

CU Manipular Backup

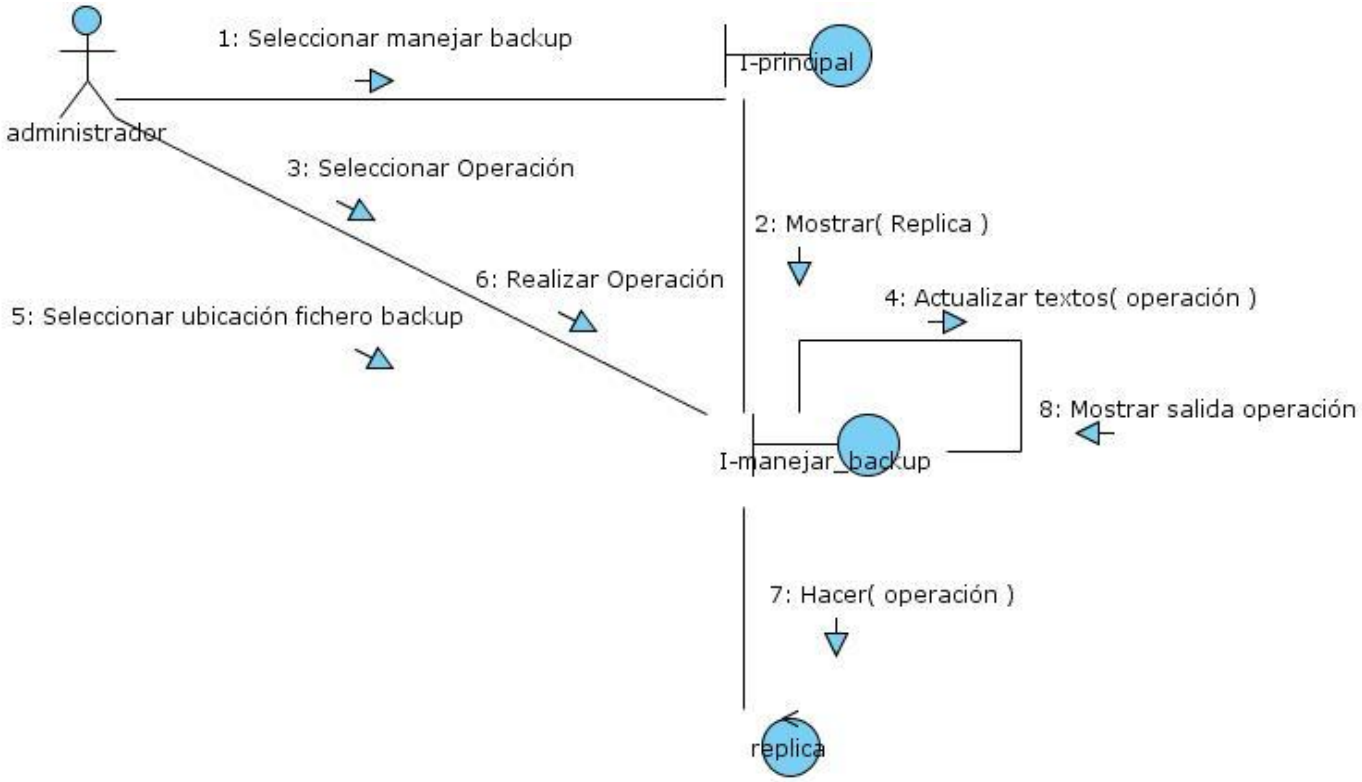


Figura 23 Diagrama de Colaboración CU Manejar Backup

CU Generar Reporte

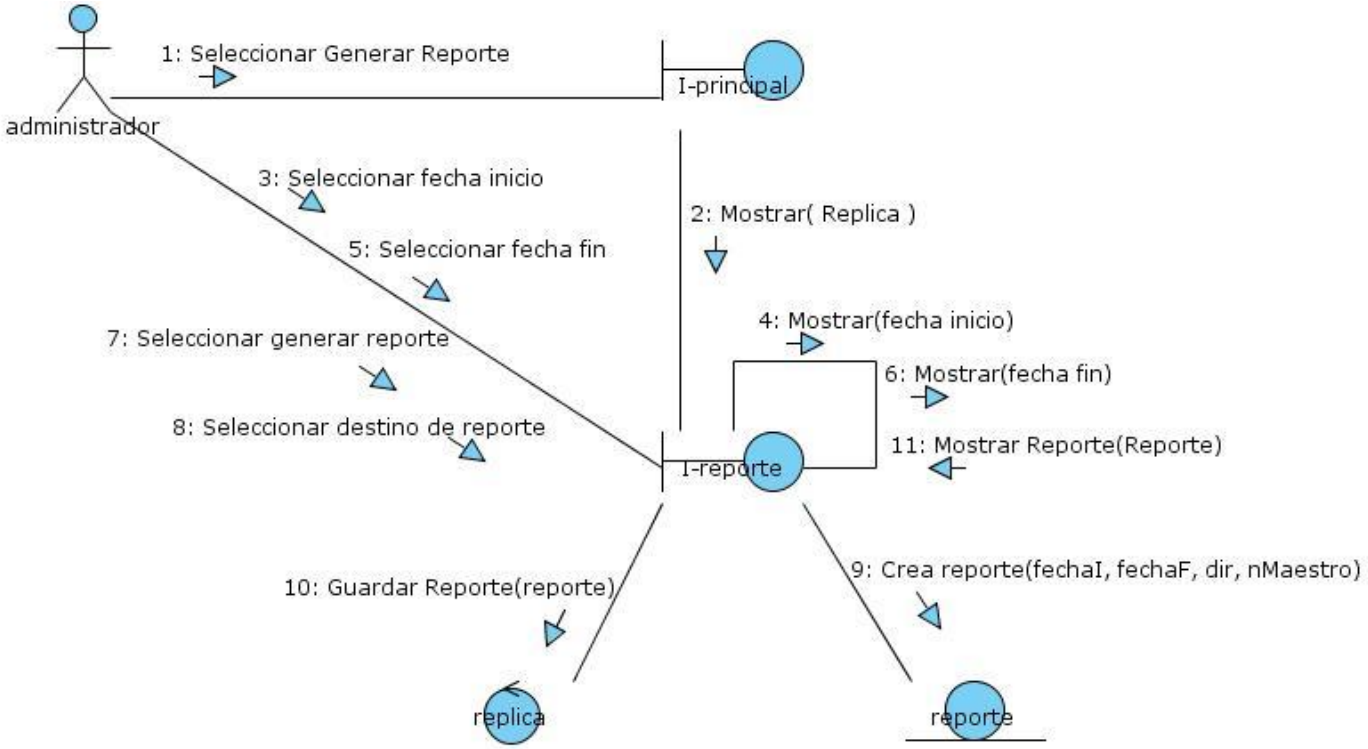


Figura 24 Diagrama de Colaboración CU Generar Reporte

CU Editar Reglas

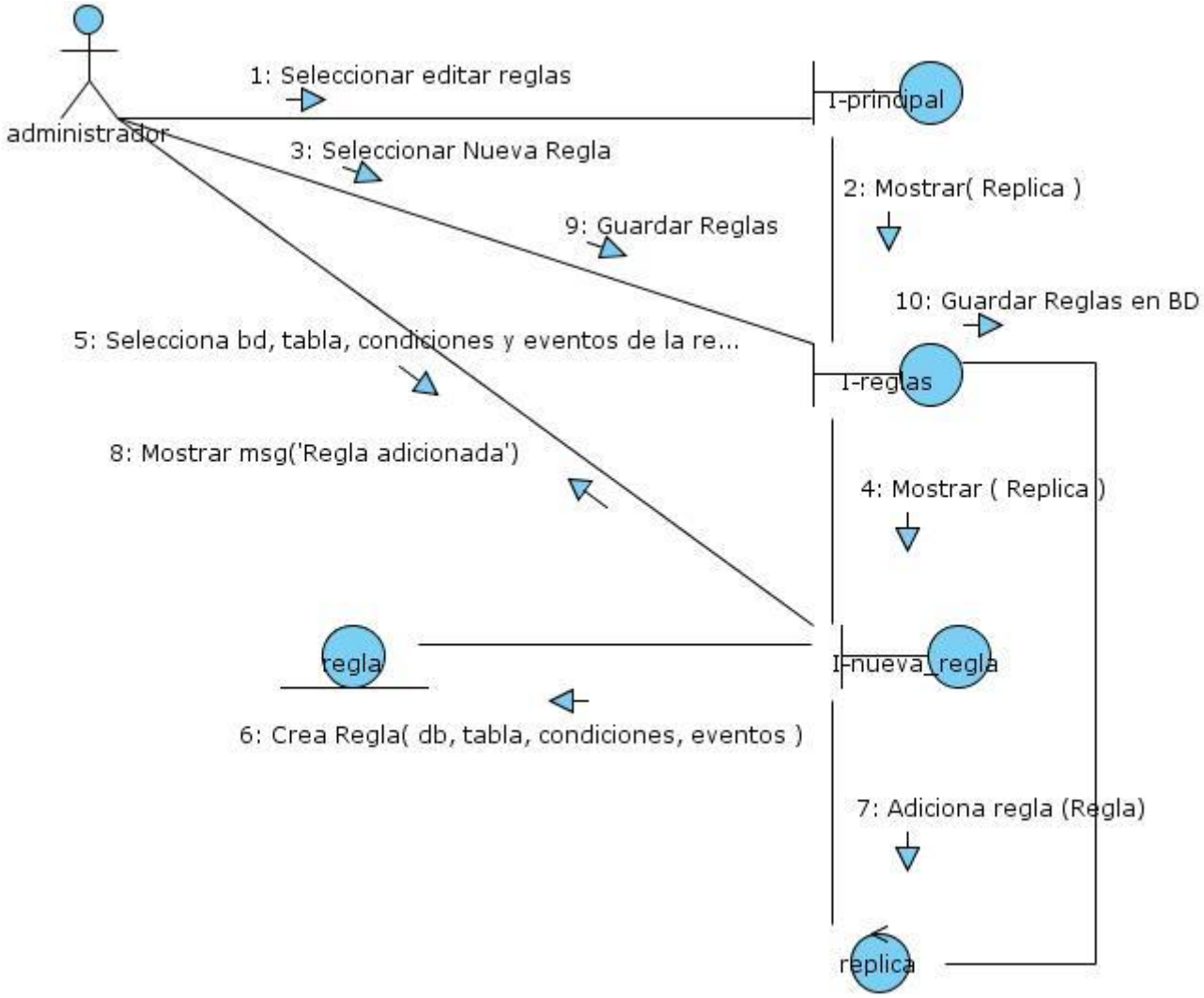


Figura 25 Diagrama de Colaboración CU Editar Reglas

3.3.3 Descripción de las clases

A continuación se describen las clases de mayor importancia en el desarrollo del sistema. Las clases entidades que se describen son la clase dsn y regla y las interfaces configurar_dsn, maestro_esclava y maestro_maestro, edit_reglas y edit_condicion por ser las que intervienen en los casos de uso Configurar Réplica y Editar Reglas. También describiremos la clase controladora replica por ser la de mayor importancia.

Caso de uso Configurar Réplica

Nombre: dns	
Tipo de Clase: Entidad	
Atributo:	**Tipo:**
nombre	String
Host	String
usuario	String
contraseña	String
puerto	Int
Base_dato	String
Para Cada Responsabilidad:	
Nombre:	def __init__(self, N, H, U, C, P, B):
Descripción:	Constructor de la clase.
Nombre:	def gNombre(self):
Descripción:	Métodos Get.
Nombre:	def gHost(self):
Descripción:	Métodos Get.
Nombre:	def gUsuario(self):
Descripción:	Métodos Get.
Nombre:	def gContrasena(self):
Descripción:	Métodos Get.
Nombre:	def gPuerto(self):
Descripción:	Métodos Get.

Nombre:	def gBaseDato(self):
Descripción:	Métodos Get.
Nombre:	def sNombre(self, N):
Descripción:	Métodos Set
Nombre:	def sHost(self, H):
Descripción:	Métodos Set
Nombre:	def sUsuario(self, U):
Descripción:	Métodos Set
Nombre:	def sContrasena(self, C):
Descripción:	Métodos Set
Nombre:	def sPuerto(self, P):
Descripción:	Métodos Set
Nombre:	def sBaseDato(self, B):
Descripción:	Métodos Set
Nombre:	def gListaDatos(self):
Descripción:	Devuelve la información como una lista.
Nombre:	def gDSN(self):
Descripción:	Devuelve un texto con esta información.
Nombre:	def conectar(self):
Descripción:	Devuelve una tupla formada por un mensaje y un objeto (DSN) con la conexión en caso de conectarse si no None.
Nombre:	def Grabar_en_BD(self, tipo, conexion):
Descripción:	Guarda los datos del DSN en la tabla 'config_pyreplica' en la base de datos 'postgres'.

Tabla 3. 1 Descripción de la clase entidad dsn.

Nombre: maestra_esclava	
Tipo de Clase: Interfaz	
Atributo:	**Tipo:**
control	
mn_file	Menu
mn_salir	MenuItem

menuOperaciones	Menu
mnmostrar	MenuItem
menuOperaciones	Menu
mnmaestro	MenuItem
mnesclavo	MenuItem
menuOperaciones	Menu
mn_elim_esclavo	MenuItem
mn_save	MenuItem
mn_config	MenuItem
StaticBox7	StaticBox
btguardar	Button
StaticBox6	StaticBox
StaticBox5	StaticBox
StaticBox4CopyCopy	StaticBox
StaticBox4Copy	StaticBox
StaticBox4	StaticBox
StaticBox3	StaticBox
esclavos	MultiColumnList
nombre	StaticText
StaticText3	StaticText
btver	Button
bt_cancelar	Button
bt_config_pyreplica	Button
bt_Eliminar	Button
bd	StaticText
lb_hos	StaticText
StaticText2	StaticText
StaticText1	StaticText
StaticBox2	StaticBox
StaticBox1	StaticBox
btEsclavo	Button
bt_maestro	Button

Para Cada Responsabilidad:	
Métodos	
Nombre:	def on_initialize(self, event):
Descripción:	Constructor de la clase.
Nombre:	def salir(self):
Descripción:	Salir de la Interfaz.
Nombre:	def configurar_DSN(self, tip):
Descripción:	Configurar un dsn, se le pasa por parametro si es maestro o esclavo o local.
Nombre:	def chequearLocal(self):
Descripción:	Chequea si el DSN local está configurado.
Nombre:	def Mostrar_Actual_Info(self, verificar):
Descripción:	Mostrando la configuración, si verificar es verdadero, ver si en postgres hay algo guardado previamente.
Nombre:	def eliminarDSN(self, tipo):
Descripción:	Parámetro tipo para futuro posible eliminación de maestro
Nombre:	def eliminandoEsclavo(self):
Descripción:	Eliminar un esclavo.
Nombre:	def guardarConfig(self):
Descripción:	Guarda la nueva configuración.
Nombre:	def config_pyreplica(self, guardar, unica):
Descripción:	Configurar Pyreplica: crear ficheros de configuracion y da la opción de guardar la configuración y de mantener otras configuraciones.
Eventos	
Nombre:	def on_btver_mouseClick(self, event):
Descripción:	Mostrar información.
Nombre:	def on_mnmostrar_select(self, event):

Descripción:	Mostrar información.
Nombre:	def on_btguardar_mouseClick(self, event):
Descripción:	Salvando la configuración.
Nombre:	def on_mn_save_select(self, event):
Descripción:	Salvando la configuración.
Nombre:	def on_bt_maestro_mouseClick(self, event):
Descripción:	Configurando dsn maestro.
Nombre:	def on_mnmaestro_select(self, event):
Descripción:	Configurando dsn maestro.
Nombre:	def on_btEsclavo_mouseClick(self, event):
Descripción:	Configurando dsn esclavos.
Nombre:	def on_mnesclavo_select(self, event):
Descripción:	Configurando dsn esclavos.
Nombre:	def on_bt_config_pyreplica_mouseClick(self, event):
Descripción:	Configurando Pyréplica.
Nombre:	def on_mn_config_select(self, event):
Descripción:	Configurando Pyréplica.
Nombre:	def on_bt_Eliminar_mouseClick(self, event):
Descripción:	Eliminando esclavo.
Nombre:	def on_mn_elim_esclavo_select(self, event):
Descripción:	Eliminando esclavo.
Nombre:	def on_bt_cancelar_mouseClick(self, event):
Descripción:	Saliendo de la interfaz.
Nombre:	def on_mn_salir_select(self, event):
Descripción:	Saliendo de la interfaz.

Tabla 3. 2 Descripción de la clase interfaz maestro_esclava.

Nombre: maestro_maestro	
Tipo de Clase: interfaz	
Atributo:	**Tipo:**
menuFile	Menu

mnsalir	MenuItem
mnope	Menu
mnmostrar	MenuItem
mnnuevo	MenuItem
mneliminar	MenuItem
mnconfigurar	MenuItem
tabla	MultiColumnList
StaticText3	StaticText
StaticBox 4	StaticBox
StaticBox 3	StaticBox
StaticBox2	StaticBox
StaticText2	StaticText
StaticText1	StaticText
cborigen	ComboBox
cbdestino	ComboBox
btrelacionar	Button
bteliminar	Button
btaceptar	Button
btcancelar	Button
StaticBox1	StaticBox
Para Cada Responsabilidad:	
Nombre:	def on_initialize(self, event):
Descripción:	Constructor de la clase.
Nombre:	def configurar_DSN(self, tipo):
Descripción:	Se configura el dsn para poder replicar.
Nombre:	def nuevoMaestro(self):
Descripción:	Realiza un nuevo maestro.
Nombre:	def mostrarMaestrosConfigurados(self):
Descripción:	Muestra todos los maestros configurados.
Nombre:	def comenzar_Replica(self):
Descripción:	Comienza la réplica después de estar todos los

	maestros configurados.
Nombre:	def configurar(self):
Descripción:	Realiza la configuración de la réplica.
Nombre:	def seleccionoOrigen(self, origenNombre):
Descripción:	Selecciona el maestro origen.
Nombre:	def seleccionoDestino(self):
Descripción:	Selecciono el maestro destino.
Nombre:	def eliminar(self):
Descripción:	Elimina el maestro seleccionado.
Nombre:	def Relacionar(self, maestroOrigen, maestroDestino):
Descripción:	Relaciona los maestros para realizar la réplica.
Eventos:	
Nombre:	def on_bteliminar_mouseClick(self, event):
Descripción:	Eliminar un maestro seleccionado.
Nombre:	def on_mneliminar_select(self, event):
Descripción:	Eliminar un maestro seleccionado.
Nombre:	def on_btrelacionar_mouseClick(self, event):
Descripción:	Relacionando los maestros seleccionados.
Nombre:	def on_cborigen_select(self, event):
Descripción:	Cuando selecciono el maestro origen.
Nombre:	def on_cbdestino_select(self, event):
Descripción:	Cuando selecciono el maestro destino.
Nombre:	def on_btaceptar_mouseClick(self, event):
Descripción:	Configurando replica.
Nombre:	def on_mnconfigurar_select(self, event):
Descripción:	Configurando replica.
Nombre:	def on_btadicionar_mouseClick(self, event):
Descripción:	Adicionando maestro a la configuración.
Nombre:	def on_mnmostrar_select(self, event):
Descripción:	Mostrando maestro
Nombre:	def on_mnsalir_select(self, event):

Descripción:	Salir de la Interfaz
Nombre:	def on_btcancelar_mouseClick(self, event):
Descripción:	Salir de la Interfaz

Tabla 3. 3 Descripción de la clase interfaz maestro_maestro.

Caso de uso Editar Reglas

Nombre: regla	
Tipo de Clase: Entidad	
Atributo:	**Tipo:**
tabla	
condiciones	
eventos	
Para Cada Responsabilidad:	
Nombre:	def __init__(self):
Descripción:	Constructor de la clase.
Nombre:	def gTabla(self):
Descripción:	Método Get.
Nombre:	def gCondiciones(self):
Descripción:	Método Get.
Nombre:	def gStr_Regla(self):
Descripción:	Método Get.
Nombre:	def gEventos(self):
Descripción:	Método Get.
Nombre:	def sTabla(self, Tabla):
Descripción:	Método Set.
Nombre:	def compara(self, cond1, cond2):
Descripción:	Compara dos condiciones pasada por parámetros.
Nombre:	def estaCondicion(self, cond):
Descripción:	Ver si una condición pasada por parámetro se encuentra en la lista.
Nombre:	def adicionarCondicion(self, condicion):
Descripción:	Adicionar una nueva condición.
Nombre:	def eliminarCond(self):

Descripción:	Hacer para que elimine la última condición (como un pop).
Nombre:	def adicionarEvento(self, evento):
Descripción:	Adiciona un nuevo evento.
Nombre:	def eliminarEvento(self, evento):
Descripción:	Elimina un evento pasado por parámetro.
Nombre:	def gcantidadCondiciones(self):
Descripción:	Retorna el número de condiciones que existen.
Nombre:	def gTipRegistros(self, evento):
Descripción:	Aquí si el evento es insert devuelve new; si es delete, devuelve old; y si es update devuelve new u old en dependencia de la selección del administrador.
Nombre:	def gFiltroDeEvento(self, evento):
Descripción:	Retorna dado un evento el filtro que le corresponde.
Nombre:	def gTodosFiltros(self):
Descripción:	Retorna una lista con todos los filtros que existen.

Tabla 3. 4 Descripción de la clase entidad regla.

Nombre: Editreglas	
Tipo de Clase: Interfaz	
Atributo:	**Tipo:**
control	
maestro	
StaticBox3	StaticBox
btmostrar_reglas	Button
cmaestros	ComboBox
StaticText1	StaticText
StaticBox2	StaticBox
StaticBox1	StaticBox

tabla_reglas	MultiColumnList
btnueva_regla	Button
bteliminar_regla	Button
btok	Button
btcancelar	Button
Para Cada Responsabilidad:	
Métodos	
Nombre:	def on_initialize(self, event):
Descripción:	El constructor de la clase.
Nombre:	def cargar(self) :
Descripción:	Carga las reglas que hay en el maestro seleccionado.
Nombre:	def on_close(self, event):
Descripción:	Cierra la conexión.
Nombre:	def salir(self):
Descripción:	Sale de la Pantalla de editar reglas.
Nombre:	def guardar(self) :
Descripción:	Guarda la regla.
Nombre:	def eliminarRegla(self):
Descripción:	Elimina la regla seleccionada.
Nombre:	def nuevaregla(self):
Descripción:	Llama a la ventana nueva regla.
Nombre:	def seleccionoMaestro(self, Maestro):
Descripción:	Cuando selecciono al maestro en el ComboBox.
Nombre:	def mostrarReglas(self):
Descripción:	Muestra las reglas que existen.
Eventos	
Nombre:	def on_cmaestros_select(self, event):
Descripción:	Seleccionando el maestro.
Nombre:	def on_btmostrar_reglas_mouseClick(self, event):
Descripción:	Mostrando reglas.

Nombre:	def on_btnueva_regla_mouseClick(self, event):
Descripción:	Realizar nueva regla.
Nombre:	def on_mnnuevo_select(self, event):
Descripción:	Realizar nueva regla.
Nombre:	def on_bteliminar_regla_mouseClick(self, event):
Descripción:	Eliminando Reglas.
Nombre:	def on_mneliminar_select(self, event):
Descripción:	Eliminando Reglas.
Nombre:	def on_btok_mouseClick(self, event):
Descripción:	Salvando en base de datos.
Nombre:	def on_mnsalir_select(self, event):
Descripción:	Saliendo.
Nombre:	def on_btcancelar_mouseClick(self, event):
Descripción:	Cancelando.

Tabla 3. 5 Descripción de la clase interfaz edit_reglas.

Nombre: editcondicion	
Tipo de Clase: interfaz	
Atributo:	**Tipo:**
maestro	Dsn
control	replica
regla	regla
act_cond	condition
menuFile	Menu
mnsalir	MenuItem
menuRegla	Menu
mnaddcond	MenuItem
mnguardar	MenuItem
mndeshacer	MenuItem
StaticLine1	StaticLine
StaticBox7	StaticBox

rg_delete	RadioGroup
StaticBox6	StaticBox
rg_insert	RadioGroup
rg_update	RadioGroup
Cdelete	CheckBox
Cupdate	CheckBox
Cinsert	CheckBox
StaticText6	StaticText
StaticBox5	StaticBox
StaticText5	StaticText
StaticText4	StaticText
StaticText3	StaticText
StaticText2	StaticText
StaticText1	StaticText
StaticBox4	StaticBox
StaticBox3	StaticBox
StaticBox2	StaticBox
StaticBox1	StaticBox
rgandnot	Radiogroup
btadicionar	Button
btdeshacer	Button
cbd	ComboBox
ctabla	ComboBox
btok	Button
btcancelar	Button
evalor	TextField
ccampo	ComboBox
coperador	ComboBox
tregla	StaticText
Para Cada Responsabilidad:	
Métodos	
Nombre:	def on_initialize(self, event):

Descripción:	Constructor de la clase, es el que inicializa la clase.
Nombre:	def mostrarRegla(self):
Descripción:	Muestra todas las reglas que existen.
Nombre:	def deshacer(self):
Descripción:	Eliminar la regla que se está creando.
Nombre:	def seleccionoBD(self, BD_Maestro):
Descripción:	Selecciona la BD maestra.
Nombre:	def seleccionoTabla(self, Tabla):
Descripción:	Selecciona la Tabla.
Nombre:	def seleccionoCampo(self):
Descripción:	Selecciona el campo de la tabla a realizar el filtro.
Nombre:	def chequearEventos(self):
Descripción:	Chequea eventos de la regla.
Nombre:	def iniciarRegla(self):
Descripción:	Inicializa la regla.
Nombre:	def guardarRegla(self):
Descripción:	Guarda la reglas con sus eventos.
Nombre:	def validaCondicion(self):
Descripción:	Valida y crea condición.
Nombre:	def adicionarCondicion(self):
Descripción:	Adicionar condición a la regla.
Eventos	
Nombre:	def on_cbd_select(self, event):
Descripción:	Seleccionando la BD.
Nombre:	def on_ctabla_select(self, event):
Descripción:	Seleccionando tabla.
Nombre:	def on_ccampo_select(self, event):
Descripción:	Seleccionando campo.
Nombre:	def on_btdeshacer_mouseClick(self, event):
Descripción:	Deshaciendo.
Nombre:	def on_mndeshacer_select(self, event):

Descripción:	Deshaciendo.
Nombre:	def on_btadicionar_mouseClick(self, event):
Descripción:	Adicionando condición.
Nombre:	def on_mnaddcond_select(self, event):
Descripción:	Adicionando condición.
Nombre:	def on_btok_mouseClick(self, event):
Descripción:	Guardando reglas.
Nombre:	def on_mnguardar_select(self, event):
Descripción:	Guardando reglas.
Nombre:	def on_mnsalir_select(self, event):
Descripción:	Salir.
Nombre:	def on_btcancelar_mouseClick(self, event):
Descripción:	Salir.

Tabla 3. 6 Descripción de la clase interfaz edit_condition

Clase controladora

Nombre: replica	
Tipo de Clase: Controladora	
Atributo:	**Tipo:**
maesto_esclav	
local	
maestro	
esclavo	
Para Cada Responsabilidad:	
Nombre:	def __init__(self):
Descripción:	Constructor de la clase.
Nombre:	def gMaestro(self, nombre):
Descripción:	Método Get de la clase.
Nombre:	def gTipoReplica(self):
Descripción:	Método Get de la clase.
Nombre:	def gMaestros(self):
Descripción:	Método Get de la clase.

Nombre:	def gEsclavos(self):
Descripción:	Método Get de la clase.
Nombre:	def gLocal(self):
Descripción:	Método Get de la clase.
Nombre:	def sTipoReplica(self, Tipo) :
Descripción:	Método Set de la clase.
Nombre:	def sMaestros(self, Ms):
Descripción:	Método Set de la clase.
Nombre:	def sEsclavos(self, Es):
Descripción:	Método Set de la clase.
Nombre:	def sLocal(self, L):
Descripción:	Método Set de la clase.
Nombre:	def esta_Configurado_Local(self):
Descripción:	Comprobar si el DSN local ha sido configurado satisfactoriamente.
Nombre:	def BuscarMaestro(self, Maestro):
Descripción:	Ver si el maestro pasado por parámetro ha sido configurado.
Nombre:	def Compara(self, dsn1, dsn2):
Descripción:	Compara dos dsn.
Nombre:	def BuscarEsclavo(self, Esc):
Descripción:	Ver si el esclavo pasado por parámetro ha sido configurado.
Nombre:	def adicionarMaestro(self, Maestro):
Descripción:	Adicionar maestro: si Rep. es .M-E. solo habrá un maestro y al menos un esclavo.
Nombre:	def adicionarEsclavo(self, Esclavo):
Descripción:	Adicionar esclavo a la configuración.
Nombre:	def eliminarEsclavo(self, Esclavo):
Descripción:	Eliminar el esclavo 'Esclavo'.
Nombre:	def existe_Tabla_en(self, nombreTabla, dsn):

Descripción:	Verificar si existe la tabla 'nombreTabla' en el DSN 'dsn'.
Nombre:	def gTupla_Desde_texto(self, texto):
Descripción:	A partir de un texto de la clase DSN. Devolver los parámetros en forma de tupla.
Nombre:	def comprobarConfiguracion(self):
Descripción:	Comprobar la información que tengo por si ha habido cambios en la base de datos.
Nombre:	def cargar_Configuracion(self):
Descripción:	carga la configuración que hay en 'congig_pyreplica' en "Postgres"
Nombre:	def GrabarConfig_en_BDPostgres(self):
Descripción:	Guardar la configuración actual en la tabla 'config_pyreplica' de la base de datos 'postgres'.
Nombre:	def ConfigurarPyreplica(self):
Descripción:	Configuro el Pyréplica, en este punto debe quedar listo para comenzar la réplica.

Tabla 3. 7 Descripción de la clase controladora replica.

3.4 Principios del diseño

Para la descripción de los casos de uso vistos en el Capítulo 2, se utilizaron los siguientes patrones para que estas descripciones fuesen los más claras posibles y así posibilitar que las personas ajenas puedan entender fácilmente cada caso de uso.

Los patrones que se utilizaron fueron:

- **El nombre revela la intención**: La utilización de este patrón permite tener los casos de uso nombrados correctamente, al proponer crear los nombres de los casos de uso con un verbo que identifique la función que se realiza o una frase que represente la meta del actor primario.

- **Escenarios más fragmentados**: Cuando se está describiendo un caso de uso, se deben escribir los eventos del flujo principal como un escenario simple sin considerar posibles fallos quedando claramente identificado. Seguido de esto se pondrán los flujos que muestran condiciones alternativas que podrán ocurrir.
- **Preciso y legible**: Este patrón plantea que las descripciones de los casos de uso sean lo suficientemente legibles como para que los clientes comprendan hasta donde se están describiendo las funcionalidades del sistema a construir. Además le permitirá al usuario evaluar y precisar cada caso de uso con el fin de definir cuál es el que está listo para comenzarse a implementar.

Para el diseño de las clases y los diagramas de interacción se emplearon algunos patrones Grasp con el propósito de lograr un diseño bien hecho donde se pueda ver claramente las relaciones esenciales de los objetos. Estos son:

- **Experto**: La aplicación de este patrón permite a cada clase desarrollar las tareas que pueden realizar según la información que poseen.
- **Creador**: Permite crear instancias de otras clases en correspondencia con la responsabilidad dada. Con esto se logró conservar el encapsulamiento ya que los objetos logran valerse de su propia información para realizar lo que se les pide.
- **Bajo acoplamiento**: Este patrón soluciona el inconveniente de dar soporte a una dependencia escasa y a un aumento de la reutilización.
- **Alta cohesión**: Este patrón es utilizado para mantener la complejidad dentro de los límites manejables.

El diseño obtenido cumple con los patrones de Bajo acoplamiento y Alta cohesión permitiendo la colaboración entre los elementos del diseño, sin verse afectados la reutilización de los mismos y el entendimiento de estos cuando se encuentran aislados. La creación de clases controladores facilitó realizar las operaciones del sistema, debido a que estas operaciones reflejan los procesos del dominio y no es factible manejarlos desde las capas interfaz.

3.4.1 Tratamiento de excepciones

El tratamiento de excepciones o errores es un detalle considerado en cada acción ejecutada por el sistema. Se evitan de esta manera operaciones innecesarias y aumenta el tiempo útil disponible por el usuario. En caso de que no se pueda continuar ejecutando la opción deseada porque una secuencia no se pueda realizar se emite un mensaje de alerta al usuario.

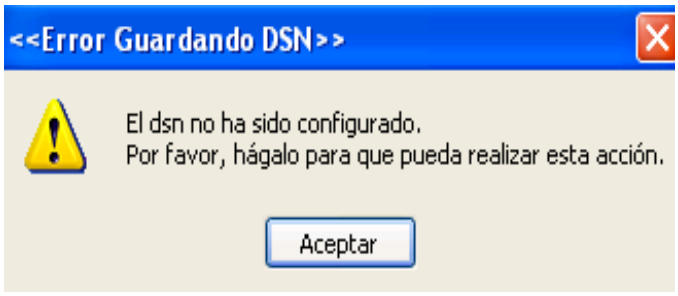


Figura 26 Mensaje de alerta

Se controla la manipulación de la información en la base de datos, informando al usuario en cada momento lo que ha ocurrido como resultado de su interacción con esta.

3.4.2 Estándares en la interfaz de la aplicación

La aplicación, a pesar de ser del tipo de escritorio y no tener tanta posibilidad como una Web para mostrar una interfaz colorida, presenta una interfaz intuitiva e interactiva que le guiará al usuario en las diferentes funcionalidades del sistema.

3.4.3 Concepción general de la ayuda

El sistema cuenta con una ayuda donde se guía a las personas que no han trabajado con la aplicación o tienen poco dominio en ella para que puedan realizar las operaciones con menor grado de dificultad.

3.5 Conclusiones

En el presente capítulo se ha descrito la solución propuesta a través de los distintos artefactos que define RUP para la construcción de aplicaciones. Han quedado detalladas las clases y las relaciones entre ellas, se representaron los diagramas de clases del análisis para un mayor entendimiento del funcionamiento de la aplicación y los diagramas de colaboración para profundizar en este mismo sentido. Se seleccionaron los patrones de diseño a emplear en la implementación y se han descrito las clases con todos sus atributos y funcionalidades

Capítulo 4. Implementación y prueba

4.1 Introducción

En el presente capítulo se muestran las relaciones y dependencias entre los ficheros de implementación, así como una descripción de cada uno. Luego se valida la solución propuesta a través de las diferentes pruebas de unidad. Dentro de estas, se utilizan los métodos de pruebas de caja blanca y pruebas de caja negra, definiéndose el objetivo de cada una de estas, así como su alcance y detalles de las mismas.

4.2 Implementación

4.2.1. Diagrama de despliegue

Muestra las relaciones entre el hardware y el software en el sistema final. Se representa como un grafo de nodos unidos por conexiones de comunicación.

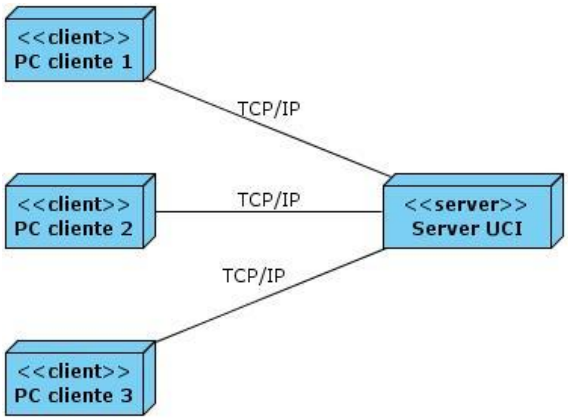


Figura 27 Diagrama de despliegue.

4.3 Diagrama de componentes

Se representa como un grafo de componentes de software unidos por medio de relaciones de dependencia, pudiendo mostrarse las interfaces que estos soporten. Es el conjunto de ficheros interrelacionados entre sí para lograr la completa funcionalidad del sistema.

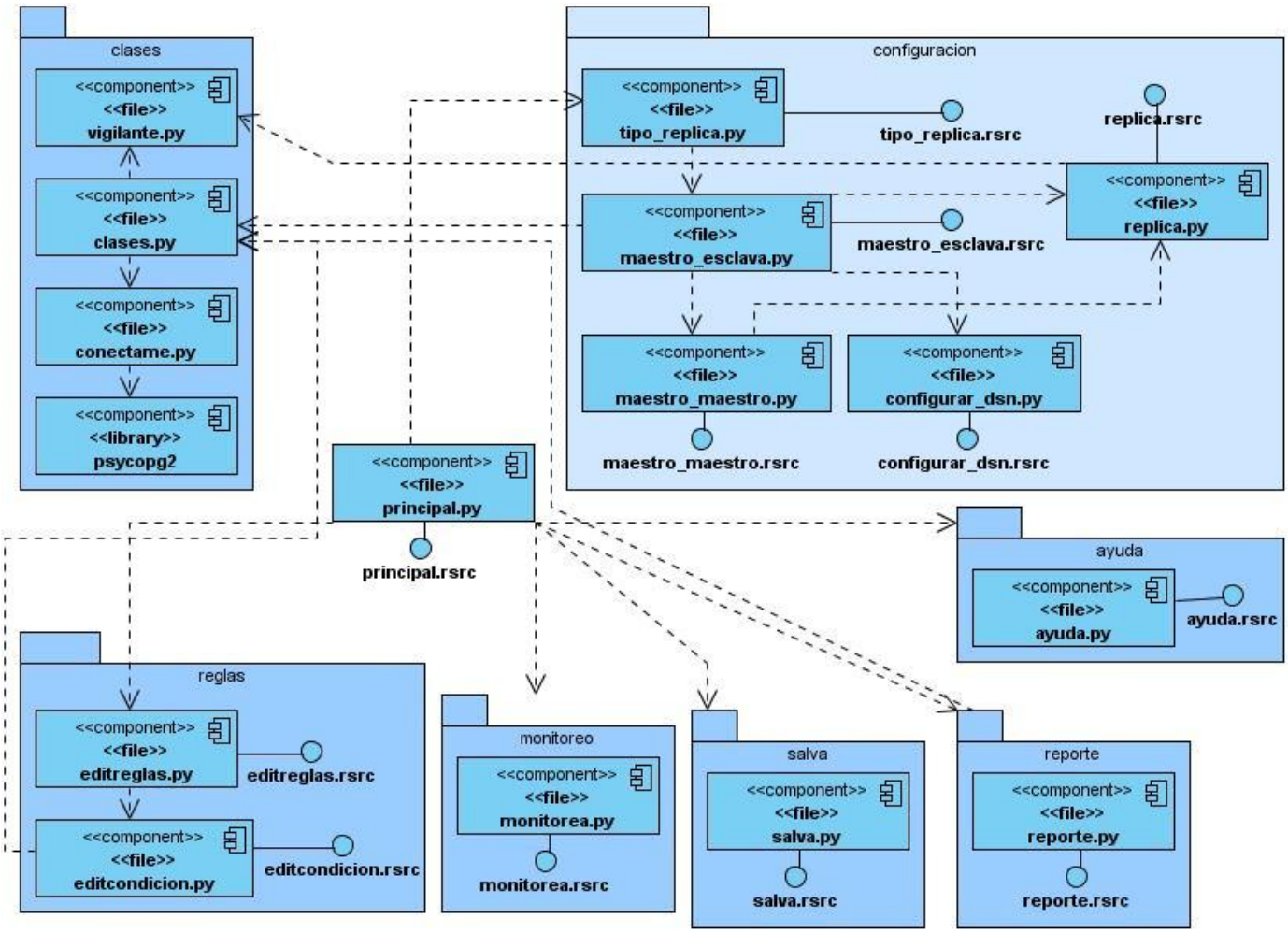


Figura 28 Diagrama de Componentes

A continuación se describen los componentes que conforman el diagrama:

En el paquete **clases**: El fichero **clases.py** es donde se encuentran las clases **condicion**, **regla**, **dsn**, **dsn_maestro**, **reporte** y **replica** implementadas.

- El fichero **conectame.py** es donde se realiza la conexión con PostgreSQL utilizando la librería psycopg2 que se importa en este fichero.
- El fichero **vigilante.py** es donde está implementada una la clase **atiende** que se encarga de ejecutar el proceso que lleva a cabo la réplica de forma paralela al sistema de configuración.

En el paquete **configuracion**:

- El fichero **tipo_replica.py** está relacionado con su interfaz **tipo_replica.rsrc** y es donde el administrador va a seleccionar el tipo de réplica que va a configurar, la maestra-esclava o la maestra-maestra.
- El fichero **maestro_esclava.py** tiene también su interfaz **maestro_esclava.rsrc** que es donde el administrador va a configurar la réplica maestro-esclavo, como el nombre lo indica.
- El fichero **maestro_ maestro.py** tiene su interfaz **maestro_ maestro.rsrc** y es aquí donde se va a configurar la réplica maestro-maestro.
- El fichero **configurar_dsn.py** con su interfaz **configurar_dsn.rsrc** tiene la responsabilidad de crear un dsn, ya sea local, esclavo o maestro.
- El fichero **replica.py** y su interfaz **replica.rsrc** tiene la función de ejecutar y mostrar los resultados de la réplica en un momento determinado seleccionado por el administrador.

En el paquete **monitoreo**:

- El fichero **monitorea.py** y su interfaz **monitorea.rsrc** es donde el administrador puede ver en tiempo real la situación de la información que está siendo replicada.

En el paquete **salva**:

- El fichero **salva.py** y su interfaz **salva.rsrc** se encargan de manipular Backups, se puede realizar un Backup de una base de datos así como restaurar una a partir de un Backup.

En el paquete **reporte**:

- El fichero **reporte.py** y su interfaz **reporte.rsrc** se encargan de permitirle al administrador guardar reportes de réplicas realizadas en un intervalo de tiempo seleccionado por este.

En el paquete **reglas**:

- El fichero **editreglas.py** y su interfaz **editreglas.rsrc** se encargan de permitir editar nuevas reglas y eliminar reglas ya guardadas.
- El fichero **editcondicion.py** y su interfaz **editcondicion.rsrc** se encargan de editar una nueva regla.

En el paquete **ayuda**:

- El fichero **ayuda.py** y su interfaz **ayuda.rsrc** son las encargadas de mostrar una pequeña ayuda al usuario.

El fichero **principal.py** y su interfaz **principal.rsrc** es la interfaz primaria de la aplicación, de donde el administrador puede realizar todas las operaciones como configurar réplica, editar reglas, manipular Backup, generar reporte y monitorear réplica.

4.4 Prueba

La prueba de software es un proceso que corre en paralelo al proceso de desarrollo y que se realiza por el convencimiento de que todo sistema debe ser “revisado” con el objetivo de establecer el nivel de calidad requerido.

En ese proceso se ejecuta el sistema a probar bajo condiciones específicas y se le aplica un conjunto de estímulos diseñados ingenierilmente, con el objetivo de detectar insatisfacción de los requerimientos planteados; todas las actividades y sus resultados son observados y registrado donde se realiza una evaluación del sistema o componente.

4.4.1 Métodos de Prueba

4.4.1.1 Pruebas de caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Objetivos

El objetivo de realizar este tipo de prueba al sistema es para revelar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaz, rendimiento y errores de inicialización y terminación.

Alcance

El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la unidad observable externamente y la calidad funcional.

Descripción

Se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Dentro de las técnicas de Prueba de Caja Negra se utilizaron la Partición de Equivalencia y el Análisis de Valores Límite (AVL). La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Por su parte, el Análisis de Valores Límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

Nombre del caso de uso: Configurar DSN

Entrada	**Resultados**	**Condiciones**
El actor introduce los datos (correctos) necesarios para conectarse al servidor de datos local (host, puerto, usuario y contraseña) y oprime el botón ***Comprobar Conexión.***	El sistema muestra la lista de las bases de datos existentes en el servidor.	Que esté instalado el SGBD PostgreSQL y corriendo.
El actor introduce los datos (incorrectos) necesarios para conectarse al servidor de	El sistema muestra un mensaje en dependencia del dato incorrecto	Que esté instalado el SGBD PostgreSQL y corriendo.

datos primario (host, puerto, usuario y contraseña) y oprime el botón ***Comprobar Conexión***.	por el que no se pudo conectar.	
El actor omite algunos de los datos necesarios para conectarse al servidor de datos primario (host, usuario, contraseña y puerto) y oprime el botón ***Comprobar Conexión***.	El sistema muestra un mensaje de alerta pidiendo al actor que teclee el dato omitido.	Que esté instalado el SGBD PostgreSQL y corriendo.
Si el actor oprime el botón **Cancelar.**	El sistema retorna a la interfaz padre.	

Tabla 4. 1 Descripción de prueba para el CU Configurar DSN

- **Nombre del caso de uso: Configurar réplica maestro-esclavo**

Entrada	**Resultados**	**Condiciones**
El actor introduce los datos (correctos) necesarios para Configurar el servidor maestro (nombre del DNS, host, puerto, usuario, contraseña) y escoge la BD maestra y oprime el botón **Aceptar**	El sistema muestra un mensaje de confirmación "***¿Desea guardarlo como maestro?"***	El dsn local tiene que estar configurado.
El actor introduce los datos (correctos) necesarios para Configurar el servidor esclavo (nombre del DNS, host, puerto, usuario, contraseña) y escoge la BD maestra y oprime el botón **Aceptar**	El sistema muestra un mensaje de confirmación "***¿Desea guardarlo como esclavo?"***	El dsn local tiene que estar configurado.
El actor introduce los datos (incorrectos) necesarios para Configurar el servidor maestro (nombre del DNS, host, puerto, usuario, contraseña) y escoge la BD maestra y oprime el botón **Aceptar**	El sistema muestra un mensaje de alerta de que no se ha podido establecer la conexión.	El dsn local tiene que estar configurado.

El actor introduce los datos (incorrectos) necesarios para Configurar el servidor esclavo (nombre del DNS, host, puerto, usuario, contraseña) y escoge la BD maestra y oprime el botón **Aceptar**	El sistema muestra un mensaje de alerta de que no se ha podido establecer la conexión.	El dsn local tiene que estar configurado.
El actor selecciona la opción de (eliminar esclavo) y selecciona el esclavo a eliminar.	El sistema elimina el esclavo seleccionado y muestra un mensaje informando esto al actor.	Tienen que haber como mínimo 1 esclavo configurado.
Si el actor oprime el botón **Cancelar**.	El sistema cierra la interfaz maestro_esclava sin hacer nada.	

Tabla 4. 2 Descripción de prueba para el CU Configurar replica maestro- esclavo

- **Nombre del caso de uso: Insertar regla de replicación**

Entrada	**Resultados**	**Condiciones**
El actor selecciona la BD maestra donde desea de realizar la regla.	El sistema muestra todas las tablas que tienen la BD maestra seleccionada.	Tiene que estar configurada la réplica.
El actor selecciona la tabla donde desea realizar la regla.	El sistema muestra todos los campos que tienen la tabla seleccionada.	Tiene que estar seleccionado el dsn.
El actor selecciona el campo que va servir para realizar la regla.	El sistema muestra los operadores para realizar los filtros.	Tiene que estar la tabla seleccionada.
El actor selecciona el operador e inserta el valor a comparar (correcto) y presiona el	El sistema almacena la condición, muestra la regla y da la opción de	

botón **Adicionar Condición.**	realizar nuevas condiciones.	
El actor presiona el botón **Guardar Regla.**	El sistema muestra un mensaje de alerta diciendo al actor que debe seleccionar los eventos para la regla.	Unas o más condiciones insertadas.
El actor oprime el botón **Deshacer**	El sistema elimina la última condición insertada.	Unas o más condiciones insertadas.
El actor oprime el botón **cancelar**	El sistema regresa a la interfaz principal sin hacer nada.	

Tabla 4. 3 Descripción de prueba para el CU Insertar regla de replicación

Nombre del caso de uso: Monitorear réplica

Entrada	**Resultados**	**Condiciones**
El actor Oprime el botón **Comenzar monitoreo.**	El sistema muestra los datos de la tabla replica_log, y el porciento de datos que han sido replicados.	Debe de haber configurado la réplica.
El actor Oprime el botón **Parar Monitoreo.**	El sistema deja de actualizar la información desde la base de datos.	Debe de haber configurado la réplica.

Tabla 4. 4 Descripción de prueba para el CU Monitorear replica

Entrada	Resultados	Condiciones
El actor selecciona la opción **Guardar Backup,** introduce los datos necesarios (Selecciona BD, ubicación para el fichero y nombre del mismo) y presiona el botón **Guardar**.	El sistema realiza la salva de seguridad y la almacena en la ubicación especificada por el actor.	Réplica configurada.
El actor oprime el botón **Cancelar.**	El sistema cierra esta ventana y sale a la interfaz principal sin hacer nada.	
El actor selecciona la opción **Cargar Backup**, introduce los datos necesarios (Selecciona BD, y ubicación del backup) y presiona el botón **Restaurar**.	El sistema restaurado la BD seleccionada.	Réplica configurada.

Tabla 4. 5 Descripción de prueba para el CU Guardar Backup

- **Nombre del caso de uso: Generar reporte de réplica**

Entrada	Resultados	Condiciones
El actor selecciona la fecha de inicio y la fecha de fin para el reporte, la fecha de inicio es menor o igual a la fecha de fin. Selecciona destino y nombre del reporte y presiona el botón **Generar Reporte.**	El sistema genera el reporte en el intervalo seleccionado.	Réplica configurada. La fecha de inicio debe ser menor o igual que la fecha final.
El actor selecciona la fecha de inicio y la fecha de fin para el reporte, la fecha de inicio es mayor a la fecha de fin. Selecciona destino y nombre del reporte y presiona el botón **Generar Reporte.**	El sistema muestra un mensaje de alerta al actor diciendo que la fecha de inicio no puede ser mayor a la fecha de fin.	Réplica configurada.
El actor oprime el botón **Cancelar.**	El sistema cierra la interfaz reporte y muestra la pantalla principal sin	

	hacer nada.	

Tabla 4. 6 Descripción de prueba para el CU generar reporte de réplica.

4.4.1.2 Pruebas de caja blanca

Objetivo

El objetivo de realizar este tipo de prueba al sistema es garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo o método, todos los bucles (ciclos) en sus límites operacionales, así como las estructuras internas de datos para asegurar su validez.

Alcance

El proceso de pruebas de caja blanca se va a concentrar principalmente en validar que cada uno de los módulos o segmentos de códigos funcione apropiadamente.

Descripción

Las pruebas de caja blanca permiten examinar la lógica interna del programa sin considerar los aspectos de rendimiento. Se diseñan casos de prueba para examinar la lógica del programa. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que:

- Se ejerciten todos los caminos independientes de cada módulo.
- Se ejerciten todas las decisiones lógicas.
- Se ejecuten todos los bucles.
- Se ejecuten las estructuras de datos internas.

La prueba de Caja Blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a un software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

Métrica de la complejidad ciclomática

La complejidad ciclomática es una métrica de software extremadamente útil, pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Esta prueba se conoce como Prueba del Camino Básico. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

Se le realizó el cálculo de la complejidad ciclomática en todos los bloques de código dentro del sistema, la cual ayudó a reflejar una medida de la complejidad del código escrito, teniendo en cuenta el número de destinos posibles. Además ayudó a conocer el esfuerzo a realizar en cada una de las pruebas, que exactamente el valor de la complejidad ciclomática en cada elemento o módulo. A partir de esta medida, se diseñaron pruebas que forzaron el recorrido de estos caminos, lo cual garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdadera y falsa.

Caso de Prueba del Caso de Uso Manejar Backup

```
# operacion, si se le pasa n=0-guarda y si es 1 carga
def Operacion( self ):
    try:
        # similar para cualquier operacion
        self.components.salida.text = ''
        if self.selected_dsn == None:
            raise Exception( 'Seleccione la base de datos, por favor.' )

        loc = self.components.tdir.text
        if loc == '':
            raise Exception( 'Debe seleccionar una ubicaci\363n v\341lida para realizar cualquier operaci\363n.' )
        loc = loc.split('\\')
        location = ''
        nombreFichero = loc.pop()
        if len ( nombreFichero.split('.') ) == 1:
            nombreFichero += '.BACKUP'
        for path in loc:
            location += path + "\\"
        location += nombreFichero
        cmd = self.gCmd()
        # de aqui en adelante es diferenet para una operacion u otra
        # si el atributo guardar esta en true, eso es lo que hago
        operacion = 'guardado'
        if self.guardar:
            cmd += '\\pg_dump.exe'
            error, returncode = self.control.HacerBackup( self.selected_dsn.gNombre(), cmd, location )
        else:
            operacion = 'restaurado'
            cmd += '\\pg_restore.exe'
            error, returncode = self.control.RestaurarBD( self.selected_dsn.gNombre(), cmd, location )

        output = self.components.salida
        output.text += error
        output.text += str (returncode)
        msg = dialog.alertDialog( self, 'Se ha %s el dsn %s correctamente.'%( operacion, self.selected_dsn.gNombre() ) )

    except Exception, e:
        a = dialog.alertDialog( self, str( e ), '<<Error Realizando Operaci\363n>>' )
```

Figura 29 Código caso de uso Manejar Backup

Análisis de la Complejidad del Algoritmo

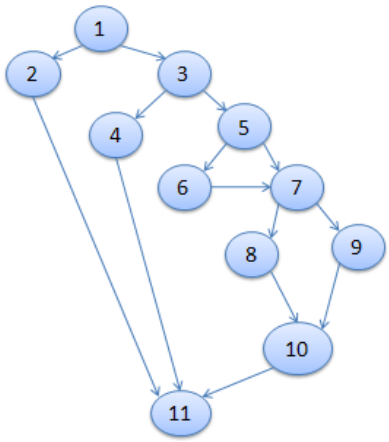


Figura 30 Grafo de flujo de código para el caso de uso Manejar Backup

Complejidad ciclomática V (G) para la Figura 31.

V (G) =A – N + 2

V (G) =14 – 11 + 2

V (G) = 5

Tabla de Posiciones de valores.

Valor de V(G)	**Evaluación**
1-10	**Un programa simple sin mucho riesgo**
11-20	Medianamente complejo
21-50	Un programa complejo
50+	Programa inestable

Tabla 4. 7 Posiciones de los valores de la complejidad dicromática

Caso de Prueba del Caso de Uso Configurar Réplica

```
def ConfigurarPyreplica( self ):
        self.GrabarConfig_en_BDPostgres()
        maestro = None
        maestro = self.maestros[0]
        msg, con = maestro.conectar()
        if msg != 'OK':
            return str( msg )
        cur = con.cursor()
        existe = self.existe_Tabla_en( 'replica_log', maestro )
        if existe == 'Si':
            sql = open( 'c:\\pyreplica\\cleanMaster.sql', 'r' ).read()
            cur.execute( sql )
            con.commit()
        sql = open( 'c:\\pyreplica\\master-install-filter.sql', 'r' ).read()
        cur.execute( sql )
        con.commit()
        cur.execute( """ select a.attname from pg_catalog.pg_attribute a left
        join pg_catalog.pg_type t on t.oid=a.atttypid left join pg_catalog.pg_class
        c on c.oid=a.attrelid left join pg_catalog.pg_constraint cc on cc.conrelid=c.oid
        and cc.conkey[1]=a.attnum left join pg_catalog.pg_attrdef d on d.adrelid =c.oid
        and a.attnum=d.adnum where c.relname='%s' and a.attnum > 0 and t.oid=a.atttypid
        """ % "replica_log" )
        campos = []
        for campo in cur:
            campos.append( campo[0] )
        campos.remove( 'id' )
        campos.remove( 'sql' )
        campos.remove( 'ts' )
        campos.remove( 'username' )
        for columna in campos:
            cur.execute( """ ALTER TABLE replica_log drop %s """ % columna )
        con.commit()
        if self.maest_esc == True:
            for esclavo in self.esclavos:
                name_columna = str ( esclavo.gNombre() + '_replicated' )
                cur.execute( """ ALTER TABLE replica_log add %s BOOLEAN DEFAULT FALSE """ % name_columna )
            con.commit()
        else:
            for maestro in maestro.maestros:
                name_columna = str ( maestro.gNombre() + '_replicated' )
                cur.execute( """ ALTER TABLE replica_log add %s BOOLEAN DEFAULT FALSE """ % name_columna )
            con.commit()
```

```
path = "C://pyreplica"
files = [file for file in os.listdir( path ) if file.endswith( '.conf' )]
for file in files:
    os.remove( path+'/'+file )
if self.maest_esc == True:
    for esclavo in self.esclavos:
        esclavo.creaArchivoPropio( maestro )
else:
    for m in maestro.maestros:
        modelo = []
        nombreFichero = maestro.gNombre() + '_' + m.gNombre()
        modelo.append( '[MAIN]' )
        modelo.append( "NAME=" + nombreFichero )
        modelo.append( "DSN0=dbname="+maestro.gBaseDato()+" user="+maestro.gUsuario()+" password="
        +maestro.gContrasena()+" host="+maestro.gHost()+" port="+str(maestro.gPuerto()) )
        modelo.append( "DSN1=dbname="+m.gBaseDato()+" user="+m.gUsuario()+" password="+m.gContrasena()
        +" host="+m.gHost()+" port="+str(m.gPuerto()) )
        modelo.append( 'SKIP_USER=' + m.gNombre() )
        modelo.append( 'SLAVE_FIELD=' + m.gNombre() + '_replicated'  )
        modelo.append( 'KEEPALIVE=True' )
        modelo.append( 'DEBUG_LEVEL=2' )
        nombreCompleto = path + '/' + nombreFichero + '.CONF'
        file = open( nombreCompleto, 'w' )
        for line in modelo:
            file.write( line + '\n' )
        file.close()
    for maest in self.maestros:
        msg, con = maest.conectar()
        if msg != 'OK':
            raise msg
        cur = con.cursor()
        sql = 'CREATE ROLE %s PASSWORD "%s" SUPERUSER INHERIT CREATEDB CREATEROLE'%( str( maest.gNombre() ),
         str( maest.gContrasena() ) )
        cur.execute( sql )
        con.commit()
        for destino in maest.maestros:
            sql = 'CREATE ROLE %s LOGIN PASSWORD "%s" SUPERUSER INHERIT CREATEDB CREATEROLE'%( str( destino.gNombre() ),
             str( destino.gContrasena() ) )
            cur.execute( sql )
            con.commit()
        con.close()
        break
```

Figura 31 Código caso de uso Configurar replica.

Análisis de la Complejidad del Algoritmo

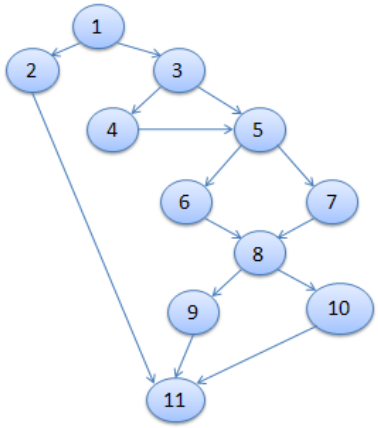


Figura 32 Grafo de flujo de código para el caso de uso Configurar replica.

Complejidad ciclomática V (G) para la Figura 33.

V (G) =A – N + 2

V (G) =14 – 11 + 2

V (G) = 5

Valor de V(G)	**Evaluación**
1-10	**Un programa simple sin mucho riesgo**
11-20	Medianamente complejo
21-50	Un programa complejo
50+	Programa inestable

Tabla 4. 8 Posiciones de los valores de la complejidad dicromática

4.3 Conclusiones

En el presente capítulo se ha quedado especificada la estructura física del sistema, así como la forma en que puede intercambiar la información con los usuarios, además se han definido los casos de prueba y métodos de prueba como lo son la prueba de caja blanca y prueba de caja negra para garantizar la correcta funcionalidad del sistema.

Conclusiones

Con la implementación de este sistema se da cumplimiento al objetivo general trazado al inicio del desarrollo del trabajo, realizando el diseño y la implementación de una aplicación que realice la réplica de datos entre la UCI y las Facultades Regionales, además se cumplieron los objetivos específicos :

- Se buscaron los conceptos de réplica de datos, bases de datos distribuidas y sistemas de replicación.
- Se hizo una búsqueda de las tecnologías y herramientas más utilizadas en la solución de aplicaciones relacionadas con el objeto de estudio.
- Se conocieron los distintos proyectos que han implementado sistemas de replicación en la universidad.
- Se recopiló información sobre las características que debía cumplir la aplicación como requisitos funcionales y no funcionales.
- Se diseñó la aplicación.
- Se implementó la aplicación.
- Se hizo un pequeño manual para que las personas ajenas pudiesen entender el funcionamiento básico del sistema.

Recomendaciones

Este trabajo da solución al problema planteado inicialmente y satisface los objetivos trazados al inicio de la investigación, pero hay cosas que se pueden incorporarle en un futuro para que su eficiencia sea aún mayor y no se vea enmarcado en determinadas áreas como el uso de un solo SGBD que es el PostgreSQL.

Se recomienda trabajar en la misma para que soporte réplicas entre los SGBD como MySQL, Oracle, y Microsoft SQL Server. Además permitir la configuración y utilización de notificaciones vía correo electrónico que viene incorporado en la herramienta Pyreplica. También el uso del protocolo SSH para configurar la réplica remotamente, en estos momentos el sistema que actuará como maestro debe ser configurado presencialmente, esto traería como beneficios que no haya que trasladarse hacia el lugar donde se encuentra este DSN cada vez que se desee configurar una réplica.

La aplicación realiza la réplica en los diferentes entornos de réplica que existen pero se debe trabajar más en el entorno de replica maestro-esclavo ya que hay temas pendiente como evitar y resolver los conflicto y eliminar bloqueos/intervención manual al producirse un error, también trabajar en base de lograr una mejor instalación en el sistema operativo de Windows.

Bibliografía

Consultada

Olarte, C. “Bases de Datos Distribuidas”. 2005. [Consultado en enero del 2007] http://atlas.puj.edu.co/~caolarte/puj/cursos/cc100/files/clases/BDDistribuidas.pdf

“FAQ de PostgreSQL en castellano actualizado”. 2005. [Consultado en enero del 2007]. http://www.dbrunas.com.ar/article.php/7394.8400337425

Citada

1. **MediaWiki.** [Online] "Base de datos " http://es.wikipedia.org/wiki/Base_de_datos.

2. Réplicas de Datos y Particiones. [Online] " Replica de datos" http://apuntes.rincondelvago.com/replicas-de-datos-y-particiones.html.

3. **Hende, G.** Aplicación para resolución de conflictos en bases de datos que no ofrezcan características de distribución de datos. [Online] 2005.

4. Sitio oficial en Argentina de PythonCard. “About PythonCard”. [Online] http://python.org.ar/pyar/PythonCard.

5. Enciclopedia Wikipedia. “Software libre”. [Online] http://es.wikipedia.org/wiki/Software_libre.

6. wikipedia. " Todo acerca de Java" [Online] [Cited: mayo 12, 2009.] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java.

7. *wikipedia.* " PHP" [Online] [Cited: mayo 12, 2009.] http://es.wikipedia.org/wiki/.php.

8. *Wikipedia.* " "Cplus plus [Online] [Cited: abril 21, 2009.] http://es.wikipedia.org/wiki/C%2B%2B.

9. Wikipedia. " Todo sobre Python" [Online] [Cited: mayo 13, 2009.] http://es.wikipedia.org/wiki/Python.

10. Wikipedia. "Wxpython " [Online] [Cited: abril 1, 2009.] http://es.wikipedia.org/wiki/WxPython.

11. Wikipedia. " MySQL" [Online] [Cited: marzo 5, 2009.] http://es.wikipedia.org/wiki/MySQL.

12. Wikipedia. "SQL Server " [Online] [Cited: abril 21, 2009.] http://es.wikipedia.org/wiki/SQLServer.

13. Enciclopedia Libre Wikipedia. " Oracle" [Online] [Cited: mayo 12, 2009.] http://es.wikipedia.org/wiki/Oracle_Corporation.

14. Enciclopedia libre Wikipedia. " Firebird" [Online] [Cited: mayo 12, 2009.] http://es.wikipedia.org/wiki/Firebird.

15. Enciclopedia libre Wikipedia. " PostgreSQL" [Online] [Cited: mayo 23, 2009.] http://es.wikipedia.org/wiki/PostgreSQL.

16. Sitio oficial de GTK. " GTK" [Online] [Cited: marzo 1, 2009.] http://www.gtk.org/.

17. Sitio oficial en Argentina de PythonCard. " PythonCard" [Online] [Cited: mayo 2, 2009.] http://www.python.com.ar/moin/PythonCard.

18. Sitio Oficial de Slony I. " Slony I" [Online] [Cited: mayo 23, 2009.] http://www.slony.info/.

19. Pgfundry. " Pgcluster" [Online] [Cited: mayo 2, 2009.] http://pgfoundry.org/projects/pgcluster/.

20. Pgfoundry. " Pyreplica" [Online] [Cited: mayo 23, 2009.] http://pgfoundry.org/projects/pyreplica/.

21. Enciclopedia libre Wikipedia. [Online] [Cited: febreri 21, 2009.] http://es.wikipedia.org/wiki/ArgoUML.

22. Sitio Devx. " Pysicopg2" [Online] [Cited: mayo 21, 2009.] http://www.devx.com/opensource/Article/29071.

23. Enciclopedia libre Wikipedia. " UML" [Online] [Cited: Junio 12, 2009.] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado.

24. Enciclopedia libre Wikipedia. " RUP" [Online] [Cited: abril 21, 2009.] http://es.wikipedia.org/wiki/RUP.

25. Enciclopedia libre Wikipedia. "XP " [Online] [Cited: mayo 12, 2009.] http://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema.

26. Enciclopedia libre Wikipedia. "Scrum " [Online] [Cited: mayo 1, 2009.] http://es.wikipedia.org/wiki/Scrum.

Glosario de Términos

Aplicación: A menudo se refiere al programa que se está ejecutando y a los archivos y bases de datos con los que se trabaja.

Atomicidad: Las transacciones son atómicas cuando al ejecutarse una operación esta se ejecuta de comienzo a fin, de llegarse a presentar algún problema deshace toda la operación (todo o nada).

Aislamiento: Cuando dos transacciones se ejecutan concurrentemente, sus efectos deben ser aislados. Esto es, no debe haber efectos que no ocurrirían si ambas transacciones se hubiesen ejecutado secuencialmente.

Cometer: Se refiere a la idea de hacer que un conjunto de cambios "tentativos, o no permanentes" se conviertan en permanentes. Un uso popular es al final de una transacción de base de datos.

Consistencia: Una transacción transforma un estado consistente de la base de datos en otro igual sin necesidad de conservar la consistencia en todos los puntos intermedios.

Clúster: En la tecnología de las computadoras, un clúster es la unidad de almacenamiento en el disco rígido. Un archivo está compuesto por varios clústeres, que pueden estar almacenados en diversos lugares del disco.

DSN Nombre fuente de datos o nombre de origen de datos (Data Source Name), que representa todo lo relativo a una fuente de datos configurada por el usuario para conectarse a una Base de datos. Es decir, por cada conexión que el usuario quiera establecer con algún fabricante, tiene que especificar una serie de información que permitan al Controlador o Driver saber con qué fabricante se tiene que conectar y la cadena de conexión que tiene que enviarle a dicho fabricante para establecer la conexión con la fuente de datos ODBC .

Durabilidad: Una vez que la transacción ha terminado, su efecto no puede perderse en caso de fallas del sistema; ni siquiera si la falla ocurre inmediatamente después de terminada la transacción.

Failover: Es la capacidad de cambiar automáticamente a una redundante o de espera ordenador servidor, sistema o red o el fracaso a la terminación anormal de los activos anteriormente servidor, sistema o red. Ocurre sin intervención humana y, en general, sin previo aviso, al contrario que la conversión.

NOTIFY: Provee un modo simple de señalar o un mecanismo de comunicación entre procesos (interprocess communication, IPC) para el conjunto de procesos que acceden a la misma base de datos Postgres. Se pueden construir mecanismos de más alto nivel usando tablas en la base de datos para pasar datos adicionales (más allá de un mero nombre de condición) desde el notificador al o a los que estén a la escucha.

Polling: Hace referencia a una operación de consulta constante, generalmente hacia un dispositivo de hardware, para crear una actividad sincrónica sin el uso de interrupciones, aunque también puede suceder lo mismo para recursos de software. Por ejemplo, se podría consultar constantemente un directorio del sistema de archivos para indicarle al usuario cuando lleguen nuevos contenidos a la misma sin embargo estas constantes consultas degradarían el rendimiento del equipo y probablemente sería mejor implementar la solución por otro medio, en particular, pidiéndole al sistema operativo que informe de transferencias a ese directorio en particular.

Respaldo: Copia de seguridad o copia de respaldo, se refiere a la copia de datos de tal forma que estas copias adicionales puedan restaurar un sistema después de una perdida de información.

Transmisión secuencial: Hace referencia a una operación de consulta constante hacia un dispositivo de hardware para crear una actividad sincrónica, sin el uso de interrupciones, aunque también puede suceder lo mismo para recursos de software.

Interrogaciones: Secuencia de interrogaciones a dispositivos con el objetivo de conocer el estado operacional o determinar si está listo para enviar o recibir datos.