Universidad de las Ciencias Informáticas

Facultad 6

Herramienta planificadora de tareas para el Servidor de Gestión de PostgreSQL

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

Autoras: *Leannys Estela Fernández Cabrera*

Yudelki Driggs Aguilera

Tutores: *Ing. Rosnel Venero Acosta*

Ing. Marcos Luis Ortíz Valmaseda

La Habana, Cuba

"Año 53 de la Revolución"

Junio, 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yudelki Driggs Aguilera

_____

Firma del Autor

Leannys Estela Fernández Cabrera

_____

Firma del Autor

Ing. Rosnel Venero Acosta

_____

Firma del Tutor

Ing. Marcos Luis Ortíz Valmaseda

_____

Firma del Tutor

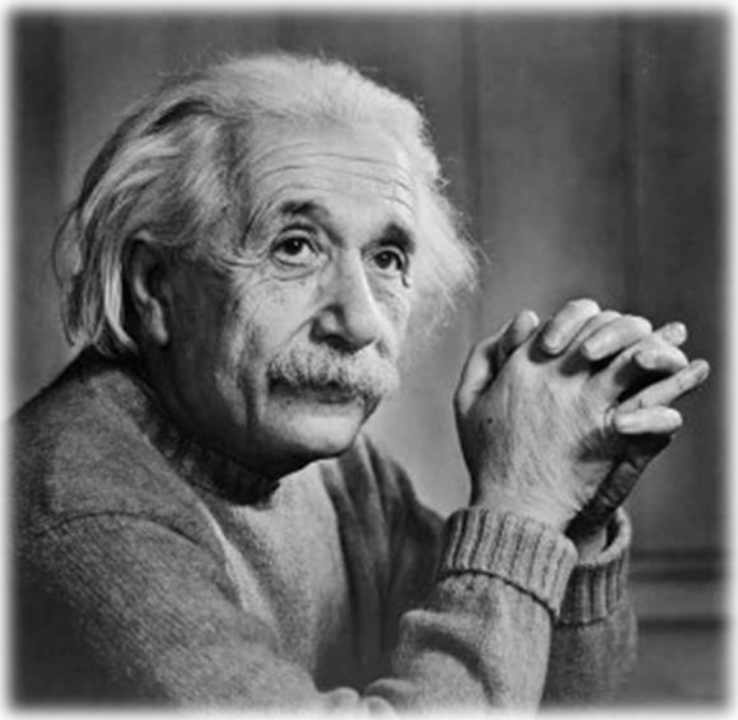
DATOS DE CONTACTO

Rosnel Venero Acosta: Ingeniero en Ciencias Informáticas, instructor recién graduado.

Correo electrónico: rvacosta@uci.cu

Marcos Luis Ortíz Valmaseda: Ingeniero en Ciencias Informáticas, instructor recién graduado.

Correo electrónico: mlortiz@uci.cu



¿Por qué esta magnífica tecnología científica, que ahorra trabajo y nos hace la vida más fácil nos aporta tan poca felicidad? La respuesta es esta, simplemente: porque aún no hemos aprendido a usarla con tino.

Albert Einstein
(1879-1955) Físico alemán.

AGRADECIMIENTOS

Leannys:

A mi papa y a mi familia por confiar siempre en mí, por darme todo el apoyo en estos años. A mi mamá, mi hermana y mi hermanito que aunque no están conmigo se que siempre estoy en sus pensamiento y en su corazón.

A mi novio siles por estar siempre cuando la necesito por brindarme toda la ayuda que necesité, por soportarme estos 3 años, por estar siempre a mi lado en los momentos difíciles.

A mis suegros por darme el aliento de seguir adelante. A mi tutor y al tribunal por servirme de guía y encaminarme siempre.

A todas mis amistades: Andy, Adriel, Julio, Palomino, Adrian, Tay, Merci, Ladi, Omar, Musa, Yixo, Raiko por confiar en mí y por darme siempre su apoyo y compartir conmigo momentos memorables.

A todos mis compañeros de los grupos en los que he estado que siempre me han brindado su apoyo. A mi compañera de tesis que ha soportado estos 10 meses.

A todos los que han contribuido de una forma u otra a la realización de este trabajo y a mi formación profesional.

Yudelki:

Por ser las personas más importantes en mi vida y haberme facilitado todo lo necesario para formarme como ingeniera agradezco a mis padres Noelia y julio y mi abuelita Cosi.

A la revolución y a Fidel por darnos la oportunidad de formarnos como jóvenes de bien, a mi compañera de tesis Leannys por no ser mi apoyo sino por ser la persona la persona con la que siempre puedo contar.

A mis hermanos Raiko y Yixander por haberme ayudado en todo y por estar ahí siempre que los he necesitado, a Taydi, Teresa, Ivette y Merci por ser las mejores amigas del mundo.

Gracias a todos mis amigos que de una forma u otra me han apoyado en toda mi carrera y gracias a mi esposo Orisbel por ser mi apoyo u mi ayuda fundamental. A mi tutor y al tribunal por servirme de guía y encaminarme siempre.

DEDICATORIA

Leannys:

Dedico esta tesis:

Especialmente a mi mamá, a mi papá.

A mi hermana y mi hermanito.

A mi novio. A toda mi familia.

A mis amigos.

A todas aquellas personas que me acompañaron en estos 5 años.

Yudelki:

Dedico esta tesis:

A mi bebé Diego Alejandro por ser esa personita que me inspira y me da todas mis fuerzas.

A mis padres.

Mi abuela Cosi.

A toda mi familia.

A mis amigos.

A todas aquellas personas que me acompañaron en estos 5 años.

RESUMEN

En el mundo existen varios Sistemas Gestores de Bases de Datos como Oracle y DB2, los mismos poseen herramientas planificadoras de tareas, estos no constituyen una solución factible para el país por ser gestores propietarios por lo que los costos de su utilización, soporte y mantenimiento son muy elevados, además dichos sistemas permiten programar tareas para un solo servidor de bases de datos. Por tales motivos el Departamento de PostgreSQL perteneciente al Centro de Tecnologías de Gestión de Datos se ha trazado la meta de planificar tareas en varios servidores de forma web, de tal forma que quede mejorada la planificación.

La presente investigación tiene como resultado una herramienta capaz de planificar tareas mediante una aplicación web, dicha herramienta cuenta además con la posibilidad de visualizar el estado de realización y planificación de las tareas, generación de reportes diarios, semanales y anuales, así como garantizar la correcta ejecución de las mismas.

Tabla de contenido

Tabla de Contenidos

INTRODUCCIÓN

Las Ciencias Informáticas surgen como respuesta a la necesidad social creciente de desarrollar métodos y medios eficaces para recopilar, conservar, buscar y divulgar la información, debido a la diversificación de las ramas científicas, así como la mezcla y surgimiento de nuevas áreas de investigación, que hicieron más complejo su proceso de organización y suministro. El siglo XX estuvo acompañado de grandes descubrimientos científicos y transformaciones tecnológicas, que ampliaron notablemente el conocimiento del hombre acerca del mundo que le rodeaba y que a su vez condujeron a cambios en la forma de interactuar con él. Poco a poco, se produjeron transformaciones en los soportes de la información y se aplicaron las nuevas tecnologías para la organización, almacenamiento y recuperación de la misma.

Estas nuevas Tecnologías de la Información y las Comunicaciones (TIC) facilitan servicios, aplicaciones, herramientas y tecnologías. Dentro de ellas la industria del software alcanza una posición relevante, por las posibilidades de acelerar y propiciar el desarrollo de los países tanto en la economía, la educación, salud y otras esferas (Graells 2008).

Cuba se ha propuesto no mantenerse aislada de los avances tecnológicos y alcanzar resultados sobresalientes en América Latina. En el país, la industria del software es relativamente nueva, se han tomado determinadas iniciativas para intentar fortalecerla, ejemplo de ello es el surgimiento en el año 2002 del ”Proyecto Futuro”, que luego pasó a ser la Universidad de las Ciencias Informáticas (UCI).

La mayoría de los productos que se implementan en la UCI requieren de varias tecnologías para un correcto funcionamiento, como es el caso de las Bases de Datos (BD), que permiten guardar grandes volúmenes de información de forma organizada, ofreciendo la posibilidad de que pueda ser consultada con facilidad. La mayoría de las BD se pueden encontrar en formato digital, lo que ofrece un amplio rango de soluciones al problema de almacenamiento de los datos.

Basado en el uso de las BD surge en el 2008 el Centro de Tecnologías de Almacenamiento y Análisis de Datos (CENTALAD) que luego se convirtió en el Centro de Tecnologías de Gestión de Datos (DATEC), apoyando la producción de software de la UCI y desempeñando varias funciones, entre ellas: proveer soluciones integrales, consultas relacionadas con tecnologías de BD y análisis de la información. Dicho

centro está estructurado por departamentos productivos, uno de ellos es departamento de PostgreSQL el cual tiene entre sus lineamientos fundamentales la extensión del gestor PostgreSQL y para ello se desarrolla el proyecto PostgreSQL Empresarial, que se enfoca en desarrollar paquetes de para los sistemas operativos más utilizados en el país, soluciones de alta disponibilidad basadas en este gestor, un Portal para la Comunidad Técnica Cubana de PostgreSQL y un Servidor de Gestión para el mismo con el objetivo de ayudar a los administradores de bases de datos a llevar el control específico del funcionamiento y rendimiento de los servidores PostgreSQL.

En apoyo al cumplimiento de los objetivos expuestos anteriormente y en correspondencia con las políticas de migración a Software de código abierto, a partir de un estudio de factibilidad, la dirección del centro DATEC decide adoptar el sistema PostgreSQL como Sistema Gestor de Bases de Datos (SGBD) (PostgreSQL 2009).

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional y de código abierto. El desarrollo de PostgreSQL no es manejado por ninguna empresa o entidad sino que es dirigido por una comunidad de desarrolladores los cuales trabajan en su desarrollo. El uso de este es cada vez más amplio en las empresas que buscan un servidor de bases de datos altamente sofisticado, con alto rendimiento, estable y capacitado para lidiar con grandes volúmenes de datos (Slideshare 2008).

La planificación de tareas ha evolucionado desde ser una aplicación más disponible para los servidores de bases de datos, a ocupar un lugar fundamental debido a que contribuye a alcanzar una mayor calidad de funcionamiento. Esta planificación consiste en construir una secuencia de tareas con la lógica necesaria y la asignación de recursos necesarios para alcanzar un objetivo.

En el mundo existen varios SGBD como Oracle y DB2 desarrollados por IBM que poseen herramientas planificadoras de tareas, dichos gestores son propietarios por lo que los costos de utilización, soporte, mantenimiento y resolución de problemas son bien elevados y esto no es favorable para el país ya que este se encuentra en su lucha por la soberanía tecnológica, además estos planificadores permiten programar tareas para un solo servidor de BD y no pueden planificar tareas desde la web.

Hoy en día se ha dado la necesitad de tener un servidor de bases de datos en los que puedan guardarse grandes volúmenes de datos, que sean lo suficientemente seguros, con respuestas rápidas y

buena disposición de los recursos. Sería eficaz planificar tareas no sólo en un servidor sino en varios y lograr llevar un control de las tareas tanto ejecutadas como a ejecutar.

Teniendo en cuenta lo anteriormente expuesto se define como **problema a resolver**:

¿Cómo programar y organizar actividades para el funcionamiento de varios servidores a partir del Servidor de Gestión de PostgreSQL?

Se define como **objeto de estudio:** Herramientas planificadoras de tareas.

El **campo de acción** que abarca el siguiente trabajo es: Herramientas planificadoras de tareas para sistemas gestores de bases de datos; para dar solución al problema de la investigación propuesto se define como **objetivo general**: Desarrollar una herramienta planificadora de tareas para el Servidor de Gestión de PostgreSQL.

Para dar cumplimiento al objetivo general se plantean como **objetivos específicos:**

- Analizar las herramientas planificadoras de tareas existentes en el mundo.
- Identificar los requisitos de la herramienta planificadora de tareas.
- Diseñar el demonio o cliente para los servidores PostgreSQL.
- Diseñar el módulo del planificador en el Dashboard.
- Implementar el demonio o cliente.
- Implementar el módulo del planificador en el Dashboard.
- Realizar las pruebas a la herramienta planificadora de tareas.

Se definen como tareas de la investigación:

- Investigación del estado del arte de los servidores de gestión de PostgreSQL existentes en el mundo.
- Identificación requisitos funcionales y no funcionales del demonio o cliente.
- Identificación requisitos funcionales y no funcionales del módulo del Dashboard.

- Diseño de la Interfaz de usuario del módulo del Dashboard.
- Diseño de las historias de usuarios.
- Diseño de clases.
- Elaboración de los modelos de diseño.
- Implementación del demonio o cliente.
- Implementación del módulo del planificador en el Dashboard.
- Elaboración del plan de pruebas y casos de prueba.
- Validación de la herramienta planificadora de tareas.

El presente documento se ha estructurado de la siguiente forma:

Capítulo 1: Fundamentos Teóricos.

El capítulo comprende un breve estudio de los planificadores de tareas para PostgreSQL, además se abordan las características principales de las herramientas, metodología y tecnologías usadas para el desarrollo de la presente herramienta.

Capítulo 2: Diseño de la solución.

En el capítulo se especifican los requerimientos funcionales y no funcionales de la aplicación, se definen y describen las historias de usuario del sistema y por último se exponen los patrones utilizados en el diseño del sistema.

Capítulo 3: Implementación y Pruebas

El capítulo contiene descripciones de la implementación de la herramienta, para dar solución a los requisitos funcionales especificados, además se define el tipo de prueba realizada a la herramienta planificadora de tareas y se diseñan los casos de prueba, además se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiéndose las tareas generadas por cada historia de usuario.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

Introducción

El capítulo comprende una breve descripción de las herramientas planificadoras de tareas así como características y conceptos de las BD y SGBD, además se abordan las características principales de las herramientas, metodología y tecnologías usadas para el desarrollo de la herramienta, se definen los roles de acuerdo con la metodología a usar.

1.1 Base de datos

Varios autores han enunciado diferentes conceptos referidos a las BD:

- Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo (García 1999).
- Serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. Las BD garantizan almacenar cualquier tipo de información, a la que se puede acceder con facilidad. Son más útiles a medida que la cantidad de datos almacenados aumentan (Valdés 2007).

Dados estos conceptos se puede llegar a la conclusión que una BD es una colección de información organizada de forma que un programa de ordenador pueda seleccionar rápidamente los fragmentos de datos que necesite, también es considerado un sistema de archivos electrónico.

1.1.1 Características propias de las Bases de Datos

Las bases de datos contienen entre sus principales características (Téllez 2005):

- Son integradas debido a que puede considerarse como la unificación de varios archivos de datos independientes.

- Poseen independencia en los datos, lo cual se refiere a la fortaleza de las aplicaciones ante cambios que pueda tener la estructura lógica de almacenamiento de los datos y la ubicación física de los mismos.

- La información puede ser compartida para múltiples usuarios y cada uno de ellos puede acceder a la misma BD y en instantes de tiempos similares, en eso consiste la concurrencia.

- Disminuye la redundancia, de manera que no existan datos repetidos y al reducir ésta al máximo se gana en un mayor aprovechamiento del espacio, además se evita que existan inconsistencias entre los datos, esto ocurre cuando se encuentra con datos contradictorios.

Cada una de las características también constituyen ventajas, destacándose el soporte de múltiples conexiones de usuarios al mismo tiempo, almacenamiento persistente por largo tiempo de determinada información y restricciones de seguridad de acceso.

1.2 Sistemas Gestores de Bases de datos

Por la necesidad de un sistema que permita administrar todo lo referente a la gestión de usuarios y el manejo de consultas complejas y no predefinidas surge para la explotación y administración de las BD los llamados SGBD, los cuales son considerados un sistema informático compuesto por hardware, software o ambos, que proporciona una técnica sistemática para la creación, el almacenamiento, el procesamiento y la consulta de la información almacenada en BD (Sicilia 2008).

1.2.1 Concepto de SGBD

Los SGBD, son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos, el usuario y las aplicaciones que lo utilizan. Se compone de un lenguaje de definición de datos (DCL), de un lenguaje de manipulación de datos (DML) y de un lenguaje de consulta (SQL). Un SGBD permite además definir los datos a distintos niveles de abstracción y manipular los mismos, garantizando la seguridad e integridad entre ellos (Sicilia 2008).

Los SGBD garantizan (Sicilia 2008):

- Un fácil acceso a los datos.

- El acceso a la información por parte de múltiples usuarios.
- La manipulación de los datos encontrados en la bases de datos: insertar, eliminar, editar y actualizar.
- Mantener las restricciones de integridad propias de la aplicación concreta.

- Mantener la seguridad evitando accesos fraudulentos a los datos, así como la extracción de información codificada.
- Permitir las copias de seguridad. Dado que un ordenador no es un sistema infalible y puede dañarse por causas propias o ajenas.

De acuerdo a lo antes expuesto se puede llegar a la conclusión que los SGBD no son más que un conjunto de programas que permiten crear y mantener una BD, asegurando su integridad, confidencialidad y seguridad.

1.3 PostgreSQL

PostgreSQL es un sistema gestor de bases de datos objeto-relacional (ORDBMS), basado en PostgreSQL 4.2, desarrollado en la Universidad de California; soporta gran parte del estándar SQL, posee características avanzadas tales como consultas complejas, llaves foráneas, disparadores, vistas, integridad transaccional, control de concurrencia multiversión y puede ser extendido por el usuario añadiéndole tipos de datos, funciones, operadores, funciones agregadas, métodos de indexado y lenguajes procedurales (Group 2009).

Está dirigido por una comunidad de desarrolladores y organizaciones comerciales que se hacen llamar PGDG (Grupo Global de Desarrollo de PostgreSQL). Es adaptable a las necesidades de los clientes por las facilidades que brinda, puede funcionar en múltiples sistemas operativos, es capaz de soportar distintos tipos de datos, el uso de índices, reglas y vistas; permite la gestión de diferentes usuarios, la declaración de funciones propias, así como la definición de disparadores. Se puede acceder al PostgreSQL con estabilidad y confiabilidad puesto que nunca se ha reportado caídas en este gestor por varios años de operación de alta actividad (Group 2009).

PostgreSQL es 100% ACID[1], funciona sobre 34 plataformas incluyendo Windows XP, Linux, FreeBSD, Solaris, Unix; y además soporta gran cantidad de lenguajes dentro de su API[2] para el desarrollo de las aplicaciones, como por ejemplo SQL, Java, Perl, Python, C, C++, Ruby y PHP (Group 2009).

[1] Conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. Un sistema de gestión de bases de datos es ACID compliant cuando cuenta con las funcionalidades necesarias para que sus transacciones tengan las características ACID (acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability).

1.4 Planificación de tareas

Un planificador de tareas consiste en diseñar un futuro deseable y seleccionar o crear una forma de lograrlo, hasta donde sea posible. Por lo tanto al planificar se construye la secuencia de tareas con la lógica necesaria y la asignación de recursos necesarios para alcanzar un objetivo (Jiménez 1982).

1.4.1 Definiciones de Planificación por varios autores

- "Es el proceso de establecer metas y elegir medios para alcanzar dichas metas" (Stoner 1996).
- "Es el proceso de establecer objetivos y escoger el medio más apropiado para el logro de los mismos antes de emprender la acción" (Goodstein 1998)
- "La planificación es un proceso de toma de decisiones para alcanzar un futuro deseado, teniendo en cuenta la situación actual y los factores internos y externos que pueden influir en el logro de los objetivos" (Jiménez 1982).

1.4.2 Herramientas planificadoras de tareas

Actualmente a nivel mundial existen varios SGBD como Oracle y DB2 desarrollado por IBM que contienen herramientas planificadoras de tareas, algunas de sus características se detallan a continuación.

Oracle es un Sistema de Gestión de Bases de Datos, desarrollado por Oracle Corporation. Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su soporte de transacciones, estabilidad, escalabilidad y soporte multiplataforma. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux (Fuh 2008).

De la ejecución de tareas programadas en Oracle se encargan los paquetes DBMS_JOB y DBMS_SCHEDULE, o se puede hacer usando el Oracle Enterprise Manager para ello. DMBS_SCHEDULE y DBMS_JOB son funcionalidades propias dentro de la base de datos con la cual se puede planificar la ejecución de tareas personalizadas como (Fuh 2008):

- Ejecución de una consulta a una hora determinada.

[2] *Application Programming Interface*. Conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

- Ejecución de backups.

- Planificación de restauración de la base de datos

DBMS_JOB son una serie de funciones y procedimientos que se usan para la definición de la tareas y DBMS_SCHEDULE extiende un poco más allá las funcionalidades brindando la posibilidad de crear las tareas, habilitarlas y deshabilitarlas (Fuh 2008).

DBMS_SCHEDULER se introdujo en la versión 10g de Oracle, y los expertos recomiendan usar este último paquete por el hecho de que DBMS_JOB puede desaparecer en un futuro. La ventaja de usar DBMS_SCHEDULER ó Oracle Enterprise Manager depende más bien del administrador de BD. El primero puede proveer más control sobre cómo la tarea es planificada y puede ser iniciada a través de la línea de comandos, mientras que el segundo ofrece un ambiente de administración completo en la web que permite al administrador de base de datos ver e interactuar directamente con la base de datos a través de un navegador web (Fuh 2008).

DB2 es una marca comercial, propiedad de IBM, bajo la cual se comercializa un sistema de gestión de bases de datos. Permite la reducción de las necesidades de consumo de alimentación, un alto rendimiento que reduce los servidores necesarios para ejecutar la BD, escalabilidad sencilla y alta disponibilidad en su arquitectura de discos de datos y otras soluciones que facilitan la colaboración entre profesionales (Zeidenstein 2009).

Principalmente, las tareas en DB2 que se pueden ejecutar son de mantenimiento y se pueden definir en el Control Center que es la interfaz principal de administración de dicho gestor.

Las principales desventajas de estos gestores (Oracle 11 g Release 2 y IBM DB2 9.7) son que (Zeidenstein 2009):

- Son sistemas gestores propietarios, por lo que los costos de utilización, soporte, mantenimiento y resolución de problemas son bien elevados.
- Hay muchas de las funcionalidades que no están habilitadas en las versiones libres de uso de los mismos. En el caso de Oracle:

- Real Aplication Cluster (más conocido como RAC): es la solución primaria de alta disponibilidad y escalabilidad-compartido de Oracle, sólo disponible en las versiones Advanced.
- Oracle Streams: Sistema que permite compartir los datos y eventos dentro de una base de datos con otra base de datos externa o dentro de la misma base de datos (sólo disponible en Oracle Enterprise Edition)
- Oracle ASM (Administración avanzada de almacenamiento): Gestión automática de almacenamiento (sólo disponible en Oracle 11g Enterprise Edition).
- Encriptación avanzada de forma trasparente (sólo disponible en Oracle 11g Enterprise Edition).

En el caso de IBM DB2 (Zeidenstein 2009):

- Automatic Maintenance (Mantenimiento automático): Simplifica la administración del almacenamiento de copias de seguridad de base de datos de rendimiento, de mantenimiento de las estadísticas actuales, y la reorganización de tablas e índices que sea necesario.
- Automatic Storage (Almacenamiento automático): Simplifica la gestión de almacenamiento en el que DB2 determina las características de almacenamiento de los espacios de tabla incluida la ubicación de los contenedores y supervisa el crecimiento de contenedores. Esto permite la flexibilidad, y un mejor rendimiento.
- Health Monitoring (Monitoreo proactivo): Controla de forma proactiva las situaciones o los cambios en el entorno de base de datos que podría resultar en una degradación del rendimiento o interrupciones potenciales. Le permite configurar umbrales de avisos y alarmas, y puede recomendar un curso de acción para resolver los problemas, ayudando a prevenir los problemas antes de que sucedan.

Existen otros planificadores como Cron que no es más que un programador de tareas para ejecución automática en un momento del tiempo definido por el usuario (Juankar 2009).

El Cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o scripts a intervalos regulares de tiempo (personalizable por el usuario). Es un demonio propio de las

plataformas basadas en Unix (Linux, Solaris, BSD, etc). En el caso de Windows, la configuración del programador de tareas se guarda en el Registro de configuraciones (Juankar 2009) .

Para la ejecución de los procesos en este planificador se debe especificar en el archivo /etc/crontab todos los datos para su ejecución como por ejemplo hora, fecha, día de la semana.

Este planificador es suficiente para gestionar las necesidades de planificación más simples, como la ejecución de un determinado programa una vez al día (para realizar, por ejemplo, Backups). Se puede usar incluso para tareas que se necesitan ejecutar a intervalos menores de tiempo (cada 15 minutos), mayores (una vez al mes), o en momentos específicas (el día 1 de cada mes) (Juankar 2009).

1.4.3 Herramientas planificadoras de tareas para PostgreSQL

PgAgent es un agente de planificación de tareas para PostgreSQL, capaz de ejecutar varios pasos de proceso por lotes / shell y tareas de SQL en programas complejos. Es como un “cron” pero de forma específica para bases de datos PostgreSQL, es de gran ayuda en la realización de trabajos programados (EnterpriseDB 2009).

Posee varias características como (EnterpriseDB 2009):

- Es gestionado por el equipo de desarrollo del PgAdmin.
- Está escrito en C++ usando wxWidgets.
- El bloqueo impide la ejecución del mismo trabajo en varios nodos.
- El motor de ejecución de múltiples hilos permite que los trabajos se ejecuten simultáneamente.
- Es un concentrador integrado de conexión, hace eficiente uso de recursos del servidor.

La desventaja fundamental de esta herramienta planificadora de tareas es que solo permite la planificación en un solo servidor y no tiene integración web.

Por tal motivo y de acuerdo con lo expuesto en los puntos anteriores referente a las herramientas planificadoras de tareas y atendiendo a las necesidades del centro DATEC se decide desarrollar una herramienta capaz de planificar tareas para varios servidores de forma web.

1.5 Metodología de desarrollo de software

El desarrollo de un buen software depende necesariamente de una serie de actividades y etapas, donde la elección de la mejor metodología es trascendental para el éxito del producto y a su vez para un equipo de desarrollo.

Una metodología de desarrollo de software no es más que un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. La metodología se puede definir como una guía que va indicando qué hacer y cómo actuar cuando se quiere obtener algún tipo de investigación. También se conoce como aquel enfoque que permite observar un problema de varias formas ya sea total, sistemática, disciplinada o con cierta disciplina (Kimmel 2002).

1.5.1 XP (Extreme Programming)

Es desarrollada como nueva disciplina de desarrollo de software .Con las teorías de su autor Kent Beck se ha conseguido el respaldo de gran parte de la industria del software y el rechazo de otra parte. La programación extrema está basada fundamentalmente en la simplicidad, la comunicación y el reciclado continuo de código, para algunos usuarios no es más que aplicar una pura lógica.

Esta metodología consiste en una programación rápida o extrema, tiene varias particularidades tales como: tener como parte del equipo al usuario final, ya que es uno de los requisitos para llegar al éxito del proyecto. Es considerada una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo (Mendoza 2004).

1.5.2 Principales características de XP

La metodología se basa en (Mendoza 2004):

- Pruebas Unitarias: pruebas realizadas a los principales procesos, se adelanta en algo al futuro para poder hacer pruebas de las fallas que pudieran ocurrir, es decir, tener una visión clara de los posibles errores a obtener.

- Refactorización: no es más que la reutilización de código, para ello se deben crear patrones o modelos y así ser más flexible a los cambios que puedan ocurrir.

- Programación en pares: consiste en que participen en un mismo proyecto dos desarrolladores y a su vez en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

1.5.3 Valores de fundamento del uso de XP

Extreme Programming (XP) está desarrollada sobre la suposición de que es posible desarrollar software de gran calidad a pesar, o incluso como consecuencia del cambio continuo. Su principal asunción es que con un poco de planificación, un poco de codificación y unas pocas pruebas se puede decidir si se está siguiendo un camino acertado o equivocado, evitando así tener que echar marcha atrás demasiado tarde (Rell 2005).

Cuatro son los valores que lo inspiran: simplicidad, retroalimentación, coraje y comunicación. XP no es un modelo de procesos ni un marco de trabajo, sino un conjunto de 12 prácticas que se complementan unas a otras y deben implementarse en un entorno de desarrollo cuya cultura se base en los cuatro valores citados (Rell 2005):

- Simplicidad: XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación, la simplicidad consiste en desarrollar sólo el sistema que realmente se necesita.

- Comunicación: XP pone en comunicación directa y continua a clientes y desarrolladores. El cliente se integra en el equipo para establecer prioridades y resolver dudas. De esta forma ve el avance día a día y es posible ajustar la agenda y las funcionalidades de forma consecuente.

- Retroalimentación rápida y continua: Una metodología basada en el desarrollo incremental de pequeñas partes, con entregas y pruebas frecuentes y continuas, proporciona un flujo de retro-información valioso para detectar los problemas o desviaciones. La retro-información es la herramienta que permite reajustar la agenda y los planes.

- Coraje: El coraje o valor existe en el contexto de los otros 3 valores (si funciona se debe mejorar). Consiste en reparar un error cuando se detecta. Tratar rápidamente con el cliente los desajustes de agendas para decidir qué partes y cuándo se van a entregar.

1.5.4 Roles de XP

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado se describirán los roles de acuerdo con la propuesta original de Kent Beck (Kimmel 2002):

Programador: Escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

Probador: El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

Se decidió usar XP porque es muy adaptable a las necesidades de la investigación y del proyecto. Además al tener un enfoque en el trabajo en grupo es más fácil dividir las contribuciones obtenidas al proyecto. Es una metodología centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, enfocada en el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo.

1.6 Lenguaje de programación

1.6.1 Python

El lenguaje fue creado por un europeo llamado Guido Van Rossum con el objetivo de cubrir la necesidad de un lenguaje orientado a objetos de un uso sencillo que sirviese para tratar diversas tareas dentro de la programación.

Durante varios años Python se estuvo desarrollando, ya en el 2000 disponía de un producto literalmente completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales (Alvarez 2007).

Éste es un lenguaje independiente de plataforma y orientado a objetos, bien preparado para realizar cualquier tipo de programa, ya sean aplicaciones Windows a servidores de red e inclusive, páginas web.

Es un lenguaje orientado a objeto, no necesariamente se necesita compilar el código fuente para poder ejecutarlo, permite mantener de forma sencilla interacción con el sistema operativo y es muy adecuado para manipular archivos de texto. Es un lenguaje sencillo pero muy potente, no genera ejecutables, sino que es el encargado de ejecutar el código (Alvarez 2007).

Python provee al producto varias ventajas entre ellas se encuentran (Clemente 2007):

- Rápido de desarrollar.
- Sencillez.
- Sus bibliotecas hacen gran parte del trabajo.
- Soporta varias bases de datos.
- Rápida curva de aprendizaje.
- Soporte a través de los módulos gran cantidad de funcionalidades.
- Tiene aplicación para algunos tipos de desarrollo: juegos en 3D, análisis de datos, desarrollo web, investigaciones científicas.
- Se puede ajustar al modelo o estilo de programación que se desee: ya sea funcional u orientado a objetos.

1.6.2 Características del lenguaje

Entre las principales características que posee Python se encuentran (Alvarez 2007):

- Propósito general: Se pueden crear todo tipo de programas.

- Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

- Interactivo: Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudar a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

- Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de cadenas de caracteres, números, archivos. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red, o cosas tan interesantes como crear archivos comprimidos en .zip.

- Sintaxis clara: Python tiene una sintaxis muy visual. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras claves Begin y End. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle[3]. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

Python está en pleno desarrollo, con él se pueden desarrollar todo tipo de programas que se ejecuten en cualquier máquina. El equipo de desarrollo de Python está trabajando de manera cada vez más organizada y cuentan con el apoyo de una comunidad que está creciendo rápidamente (Python 2007).

1.7 Tecnología usada

1.7.1 Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos (Django 2007).

Pone énfasis en la reutilización, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio DRY (del inglés Don't Repeat Yourself) su principio es hacer las cosas una sola vez y

[3]Es una sentencia que se realiza repetidas veces a un trozo aislado de código, hasta que la condición asignada a dicho bucle deje de cumplirse.

rehusarlas siempre que sea posible. Python es usado en todas las partes del framework, incluso en configuraciones, archivos y en los modelos de datos.

Mantiene de forma rigurosa un diseño limpio en su propio código y facilita que el programador siga las mejores prácticas de desarrollo web en las aplicaciones que crea. Fomenta el bajo acoplamiento, la filosofía de programación que dice que las distintas partes de la aplicación deben ser intercambiables y deben comunicarse unas con otras mediante APIs[4] claras y concisas(Django 2007).

Django es usado por muchas compañías para el desarrollo de sus aplicaciones empresariales por la rica cantidad de características con que cuenta. Fue iniciado por Jacob Kaplan-Moss y Jeremy Dunck en la World Online del periódico Lawrence Journal World, de Kansas cuando necesitaban tener un marco de trabajo común para la construcción de aplicaciones web de uso intensivo y lo hizo un proyecto de código abierto en 2005.

Algunas de las características con que cuenta este framework son (Django 2007):

- Sistema de mapeo de objetos relacionales (ORM): Permite definir los modelos de datos enteramente en Python, además de que provee una rica y dinámica API para acceso a bases de datos, aunque también da la posibilidad de construir su propio código SQL.

- Una interfaz automática de administración: Provee un simple mecanismo para la creación de interfaces para administradores que deban agregar o actualizar contenido de su sitio.

- Sistema de plantillas: Provee un mecanismo de plantillas poderoso, extensible y sin limitaciones específicas del framework para la separación del diseño, el contenido y el código de Python.

- Sistema de cacheo: Provee APIs para el cacheo de objetos en memoria usando Memcached[5] u otras aplicaciones de cacheo para obtener un mejor rendimiento de la aplicación.

- Internacionalización: Tiene un completo soporte para aplicaciones en varios idiomas, dándole la posibilidad al desarrollador de tener el control total de sus cadenas en diferentes idiomas.

[4]API es una interfaz de programación de aplicaciones.
[5]Es un sistema distribuido de cacheo de objetos en memoria en la forma clave-valor.

- Gran cantidad de aplicaciones reusables y plugins: La comunidad detrás del desarrollo del framework es inmensa, por lo que siguiendo los principios en los que está basado Django, se han desarrollado gran cantidad de aplicaciones y plugins listos para la producción de las tareas más comunes.

- Soporte para múltiples bases de datos: En la versión 1.2 fue introducida esta característica muy esperada por la comunidad, dando la posibilidad de tener varias bases de datos en un mismo proyecto de Django (Django 2007).

Django aparenta implementar el patrón MVC (Model-View-Controller) este es un patrón arquitectónico define una vía para el desarrollo de software donde el código para la definición y el acceso a los datos (modelo) es separado de la lógica del negocio (controlador), el cual es mostrado de forma separada en la interfaz del usuario (vista). En el caso de Django el controlador es llamado vista y la vista plantilla. La Vista describe "qué" datos serán presentados y no "cómo" se verán los mismos. Aquí es donde entran en juego las plantillas, que describen "cómo los datos son presentados". Se conoce que el "controlador" de un MVC clásico está representado por el propio framework, es decir, el sistema que envía una respuesta a la vista correspondiente, de acuerdo a la configuración de URL de Django (archivo de configuración). En el caso de querer hacer una correspondencia, entonces se diría que éste es un framework "MTV*":* modelo, plantilla, vista.

1.8 Herramientas de desarrollo

1.8.1 Visual Paradigm

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de un software. Contribuye a la rápida elaboración de aplicaciones de calidad y con un menor costo. Permite modelar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Características principales (Orallo 2005):

- Ingeniería inversa: Código a modelo, código a diagrama.

- Editor de Detalles: Entorno todo en uno, para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.

Esta herramienta utiliza UML como lenguaje de modelado, mediante el cual es posible establecer una serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

El lenguaje se centra en la representación gráfica de un sistema de software. El mismo indica cómo crear y leer los modelos (Orallo 2005).

Las funciones de UML se pueden sintetizar en (Orallo 2005):

- Visualizar: Permite expresar de una forma gráfica un sistema, de manera que otra persona lo puedan entender.
- Especificar: Permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.8.2 Aptana Studio

Aptana Studio (AS) es un Entorno Integrado de Desarrollo (IDE) para aplicaciones de la web 2.0, gratuito, de código abierto, con soporte Ajax, PHP, Ruby on Rails y Django. AS es una distribución enfocada en el desarrollo web, con soporte a HTML, CSS y Javascript, así como opcionalmente a otras tecnologías mencionadas. Está disponible como una aplicación independiente o como plug-in para Eclipse (Alvarez 2007).

A continuación se mencionan algunas de las características principales de Aptana Studio (Aptana 2008):

- Ayudas visuales para la escritura de scripts en diversos lenguajes, como coloreado y auto escritura del código, ayudas contextuales de referencia a medida que se escribe.
- Visualización de errores de sintaxis a medida que se escribe.
- Soporte para hacer FTP a servidores remotos, con herramientas para sincronización.

- Librerías de funciones en Javascript populares en Ajax/Javascript para utilizar en los proyectos.
- Pre-visualización de estilos CSS[6] con el editor CSS.
- Extensible por Javascript[7]. Los usuarios pueden escribir scripts para realizar acciones y macros.
- Los Snippets[8] permiten insertar fragmentos de texto que se utilizan muy a menudo.
- Acceso rápido a un terminal desde el que se pueda hacer la mayoría de las acciones que se realizan en un terminal normal, pero con la facilidad de que no tener que ir a otra ventana, que al final es uno de los objetivos cuando se opta por un IDE para desarrollar.

Muchas personas usan los entornos multiplataforma Netbeans, Visual Studio o Eclipse, pero Aptana también es relevante, ya que muchos de los desarrollares de Python, Ruby, XHTML, PHP, ASP y plataformas han probado la comodidad de este IDE, para los desarrolladores de Python este entorno contiene un soporte para todas las versiones del compilador además de las notificaciones de errores y precauciones en el código fuente (Alvarez 2007).

CONCLUSIONES DEL CAPÍTULO

En este capítulo fueron analizados los SGBD Oracle, DB2 usados por la mayoría de las corporaciones a nivel mundial pero poseen una desventaja fundamental, son gestores privativos, esto no es favorable para el país ya que este se encuentra en su lucha por la soberanía tecnológica, por tal motivo está creando sus propias soluciones para así independizarse en este sentido, atendiendo a esta desventaja se decidió desarrollar una herramienta planificadora de tareas capaz de satisfacer dichas necesidades. Para la implementación de la herramienta se utilizará Python como lenguaje de programación y Django como framework este último provee a la aplicación características como la posibilidad de tener varias bases de datos en un mismo proyecto de Django.

[6] Hojas de estilo en cascada, es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

[7] Es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

[8] Es un término de programación que se da a una pequeña región del código fuente, el código de máquina o de texto que es re-utilizable.

CAPÍTULO 2: DISEÑO DE LA SOLUCIÓN

Introducción

En el presente capítulo se especifican los requerimientos funcionales y no funcionales con los que debe contar la aplicación, se definen y describen las historias de usuario en las cuales se describen brevemente las características que el sistema debe tener, así como los prototipos de interfaz, además se exponen los patrones utilizados y se define el plan de iteraciones y entrega.

2.1 Modelo de dominio

Un Modelo de Dominio es una representación visual estática del entorno real del proyecto. Puede utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis, como paso previo al diseño de un sistema, ya sea de software o de otro tipo. El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema y ayudar a los usuarios, clientes, desarrolladores y otros interesados a utilizar un vocabulario común (Ivar Jacobson 2000).

El objetivo de este modelo es capturar los términos necesario para comprender cómo va a funcionar el sistema que se está diseñando debido a que el mismo ocupa un rol protagónico en el desarrollo moderno de software.

En la siguiente figura se muestra el modelo de dominio:

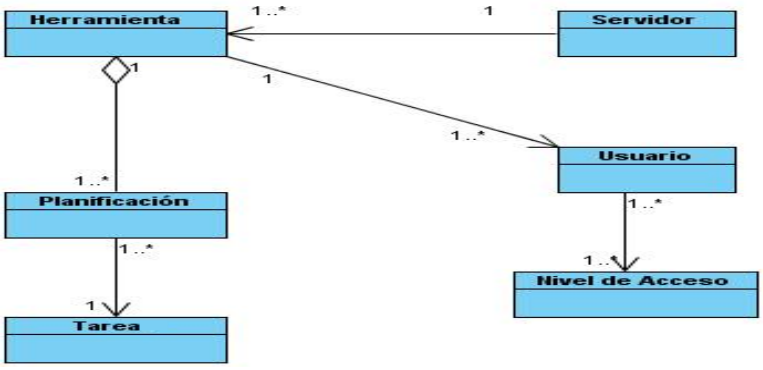


Figura 1: Modelo de dominio.

2.1.1 Definición de clases del modelo del dominio

Tabla 2.1: Definición de clases de dominio.

Nombre de la clase	Definición
Tareas	Representa la consulta SQL que se ejecuta en la bases de datos.
Planificación	Encargada de gestionar las tareas para su correcta ejecución.
Herramientas	Encargada de llevar a cabo los procesos de planificación de las tareas, es la representación visual de la planificación.
Servidor	Representa a la PC donde se encuentra el servidor de bases de datos en el cual se ejecuta tanto la herramienta como las tareas.
Usuario	Contiene como su nombre lo indica el universo de personas que accederá a la herramienta.
Nivel de Acceso	Representa los privilegios con que cuentan los usuarios cumpliendo con las políticas definidas para el acceso a la herramienta.

2.2 Requerimientos del sistema

Un requerimiento es una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema, para satisfacer un contrato, estándar u otro documento impuesto formalmente (Ivar Jacobson 2000).

2.2.1 Requerimientos funcionales

Un requerimiento funcional define el comportamiento interno del software como cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo las historias de usuario serán llevadas a la práctica. Los requerimientos funcionales se mantienen invariables sin importar con qué cualidades o propiedades se relacionen.

A continuación se muestran los requerimientos funcionales:

- RF1 - Insertar una tarea a ser ejecutada en el servidor.
- RF2 - Modificar los atributos de una tarea registrada para una correcta ejecución en el servidor.
- RF3 - Eliminar una tarea registrada en la herramienta planificadora.
- RF4 - Verificar el estado de ejecución de las tareas en los servidores.
- RF5 - Mostrar estadísticas de ejecución de las tareas en el Dashboard.
- RF6 - Listar las tareas ejecutadas y pendientes insertadas por usuario.
- RF7 - Listar las tareas ejecutadas y pendientes de una BD determinada.
- RF8 - Listar las tareas que han sido satisfactoriamente ejecutadas en los servidores.
- RF9 - Listar las tareas que se encuentran pendientes a ejecución.
- RF10 - Generar reporte de las tareas ejecutadas y pendientes por día.
- RF11 - Generar reporte de las tareas ejecutadas y pendientes por mes.
- RF12 - Generar reporte de las tareas ejecutadas y pendientes por año.
- RF13 - Re-planificar una tarea que no ha sido ejecutada, modificando las propiedades de fecha y hora.

2.2.2 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. En muchos casos, estos requerimientos son fundamentales en el éxito del producto y generalmente están vinculados a requerimientos funcionales, además corresponden a aspectos tales como la disponibilidad, soporte, seguridad, apariencia e interfaz externa.

Apariencia e interfaz externa

Se debe proporcionar un ambiente agradable y fácil para el usuario.

Usabilidad

El acceso a la aplicación debe realizarse de forma fácil y rápida. Para utilizar el sistema es necesario poseer conocimientos elementales de computación y sobre el ambiente Web en sentido general.

Rendimiento

La capacidad de respuesta de la aplicación que forma parte del Dashboard debe ser la más rápida posible.

Soporte

La aplicación podrá ser ejecutada en varias plataformas. Se debe contar el gestor de bases de datos PostgreSQL.

Seguridad

Existen niveles de acceso a la aplicación web, en dependencia del rol que desempeñe cada usuario, por lo cual se debe garantizar que las funcionalidades del sistema se muestren de acuerdo al tipo de usuario que esté activo.

Software

Para una mejor accesibilidad de la aplicación los usuarios deberán contar con el navegador Mozilla Firefox 3.0 o mayor.

Hardware

Para el desarrollo y ejecución de la aplicación se precisará contar con una RAM mínimo de 512 MB (1 GB o más recomendado), un microprocesador que tenga como propiedades mínimas Intel Pentium 4 y deberá contar con una capacidad de almacenamiento mínimo de 40 GB de espacio en disco.

Disponibilidad

La aplicación debe estar disponible a tiempo completo y debe recuperarse rápidamente ante cualquier tipo de fallo.

Requisitos legales

Las tecnologías y herramientas que se utilicen para desarrollar la aplicación web deben estar bajo licencias libres.

Restricciones de diseño e implementación

La aplicación se implementó usando el Framework Django. Como lenguaje de programación se utilizó Python, como IDE Aptana. Se utilizó un estándar de codificación para la implementación.

Persistencia

La información generada por la aplicación debe ser almacenada en bases de datos permanentemente, con el objetivo de poder generar reportes y realizar estudios posteriores.

Portabilidad

La aplicación permite ejecutarse en los sistemas operativos Windows y Linux.

2.3 Fase de exploración. Definición

El ciclo de vida de XP enfatiza en el carácter interactivo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas, que en el caso de XP corresponden a un conjunto de historias de usuarios (Beck 2000).

La metodología de desarrollo XP comienza con la fase de exploración, en esta fase los clientes plantean a grandes rasgos las historias de usuario mediante un proceso de identificación que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Beck 2000).

2.3.1 Definición de historias de usuario

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales (Fowler 2002).

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse

nuevas o ser modificadas. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Las HU conducen al proceso de creación de las pruebas de aceptación, las cuales servirán para verificar que estas historias se han implementado correctamente. Otra de sus características es que solamente proporcionan los detalles sobre la estimación del riesgo y cuánto tiempo conllevará su implementación.

2.3.2 Patrones utilizados

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Entre los patrones utilizados para el desarrollo de la herramienta se encuentran (Tedeschi 2011):

- **MVC:** Específicamente para el planificador de tareas del Servidor de Gestión de PostgreSQL el patrón de diseño MVC se estructura de modo que el archivo models.py contiene una descripción de la tabla de la bases de datos, como una clase Python. Usando esta clase se pueden crear, buscar, actualizar y borrar entradas de la BD usando código Python. A esto se llama modelo. El archivo views.py contiene la lógica de la página, en la función Naire_Server. A esta función se denomina vista. El archivo urls.py específica qué vista es llamada según el patrón URL. En este caso, la URL /Server/ será manejada por la función Naire_Server. El archivo Naire_Server.html es una plantilla HTML que describe el diseño de la página.

- **Controlador:** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado, este patrón se encuentra ejemplificado es la clase view.py, ya que la misma contiene la lógica de ejecución.

- **Experto:** En el planificador de tareas se utiliza este patrón asignándole una responsabilidad a una clase experta en algún tipo de información, esta clase es la que contiene la información necesaria para realizar la labor que tiene encomendada, como es el caso de la clase Tareas encargada de gestionar todo lo relacionado con la planificación de tareas en el servidor.

- **Creador:** El patrón creador ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos o clases. En el caso del planificador de tareas un ejemplo del patrón Creador es la clase Reporte que se encarga de crear objetos de tipo Tareas.

2.3.3 Actores del sistema

Actor	Descripción
Administrador	Posee el control total sobre el componente y puede consultar todo el proceso de planificación de las tareas. Será el único autorizado a modificar la configuración del componente, además de ser el encargado de todo el proceso de administración de las tareas (gestionar tareas, verificar tarea).
Usuario	Puede hacer consultas al planificador, así como acceder a los requisitos: generar reportes y mostrar estadísticas.

2.3.4 Descripción de las historias de usuario

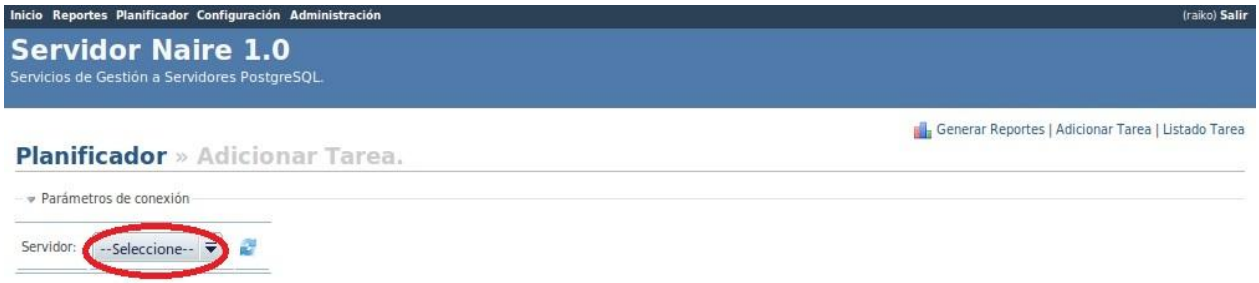
2.3.4.1 Historia de usuario – Insertar tarea

Historia de Usuario	
Número: 1	**Nombre de la Historia de Usuario:** Insertar tarea.
Cantidad de modificaciones a la Historia de Usuario: 3	
Usuario: Yudelki Driggs Aguilera	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 1 semana
Descripción: Permite insertar en la herramienta de planificación una tarea que deberá ser ejecutada en el servidor, se deben llenar correctamente los campos correspondientes la tarea a insertar, al insertar los datos se da clic en el botón “Enviar consulta” para almacenar la información	

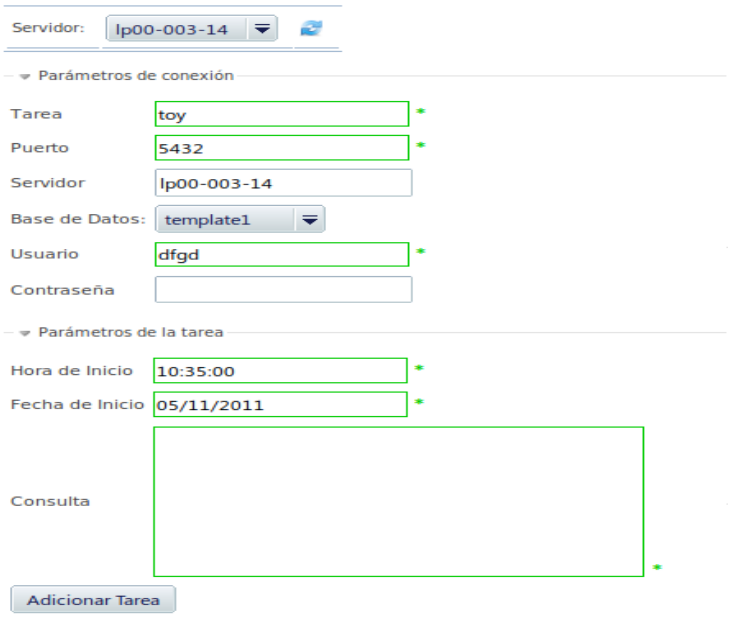
en la BD del planificador.

Prototipo de interfaces:

1- Seleccionar el servidor.



2- Llenar correctamente los datos.



2.3.4.2 Historia de usuario – Eliminar tarea

Historia de Usuario

Número: 3	**Nombre de la Historia de Usuario:** Eliminar tarea.
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Yudelki Driggs Aguilera Leannys Estela Fernández Cabrera	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 1 semana

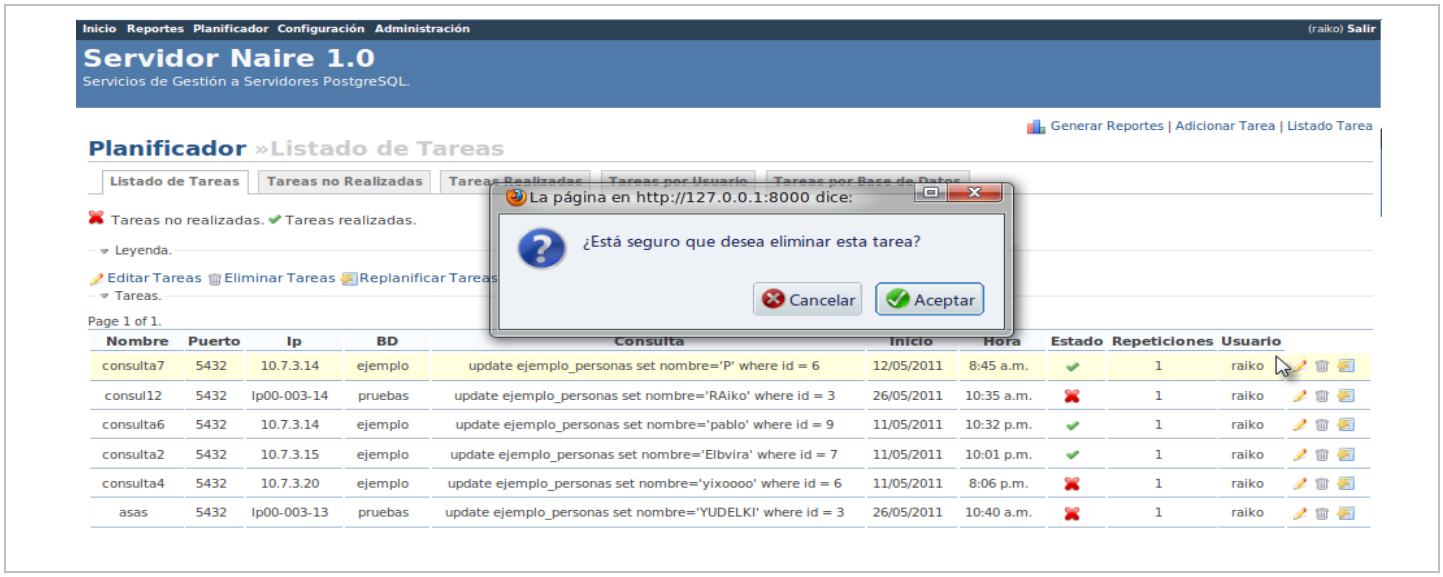
Descripción: Permite eliminar en la herramienta de planificación una tarea que deberá ser ejecutada en el servidor, se selecciona la tarea a eliminar y posteriormente saldrá una ventana que da la posibilidad al usuario de eliminar o no la tarea.

Prototipo de interfaces:

1- Seleccionar la opción eliminar.

Nombre	Puerto	Ip	BD	Consulta	Inicio	Hora	Estado	Repeticiones	Usuario
consulta	5423	10.7.3.14	prueba	update ejemplo_prueba nombre = 'cuna' where id = 1	17/05/2011	10:45 a.m.	✗	1	raiko
consulta15	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='yudisney' where id = 3	12/05/2011	10:25 a.m.	✗	2	raiko
consulta7	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='P' where id = 6	12/05/2011	8:45 a.m.	✓	1	raiko
consul12	5432	Ip00-003-14	pruebas	update ejemplo_personas set nombre='RAiko' where id = 3	26/05/2011	10:35 a.m.	✗	1	raiko
consulta6	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='pablo' where id = 9	11/05/2011	10:32 p.m.	✓	1	raiko
consulta2	5432	10.7.3.15	ejemplo	update ejemplo_personas set nombre='Elbvira' where id = 7	11/05/2011	10:01 p.m.	✓	1	raiko
consulta4	5432	10.7.3.20	ejemplo	update ejemplo_personas set nombre='yixoooo' where id = 6	11/05/2011	8:06 p.m.	✗	1	raiko

2- Hacer clic en la opción deseada.



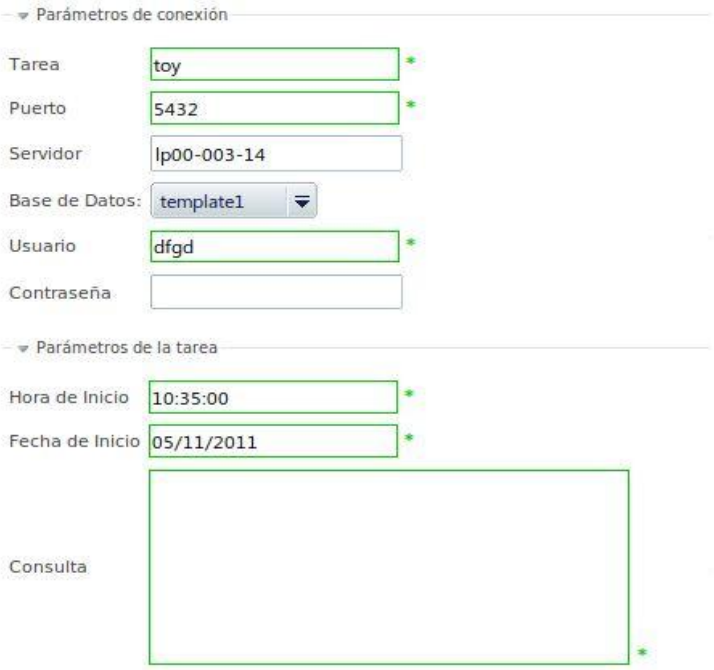
2.3.4.3 Historia de usuario – Modificar tarea

Historia de Usuario	
Número: 2	**Nombre de la Historia de Usuario:** Modificar tarea.
Cantidad de modificaciones a la Historia de Usuario: 4	
Usuario: Leannys Estela Fernández Cabrera	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 1 semana
Descripción: Permite modificar en la herramienta de planificación una tarea que deberá ser ejecutada en el servidor, se selecciona la tarea a modificar dando clic en el lápiz que aparece al final de la tarea y se modifican los campos deseados.	
Prototipo de interfaces:	

1- Seleccionar la tarea a modificar.

Nombre	Puerto	Ip	BD	Consulta	Inicio	Hora	Estado	Repeticiones	Usuario
consulta	5423	10.7.3.14	prueba	update ejemplo_prueba nombre = 'cuna' where id = 1	17/05/2011	10:45 a.m.	✗	1	raiko
consulta15	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='yudisney' where id = 3	12/05/2011	10:25 a.m.	✗	2	raiko
consulta7	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='P' where id = 6	12/05/2011	8:45 a.m.	✓	1	raiko
consul12	5432	lp00-003-14	pruebas	update ejemplo_personas set nombre='RAiko' where id = 3	26/05/2011	10:35 a.m.	✗	1	raiko
consulta6	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='pablo' where id = 9	11/05/2011	10:32 p.m.	✓	1	raiko
consulta2	5432	10.7.3.15	ejemplo	update ejemplo_personas set nombre='Elbvira' where id = 7	11/05/2011	10:01 p.m.	✓	1	raiko

2- Modificar los datos.



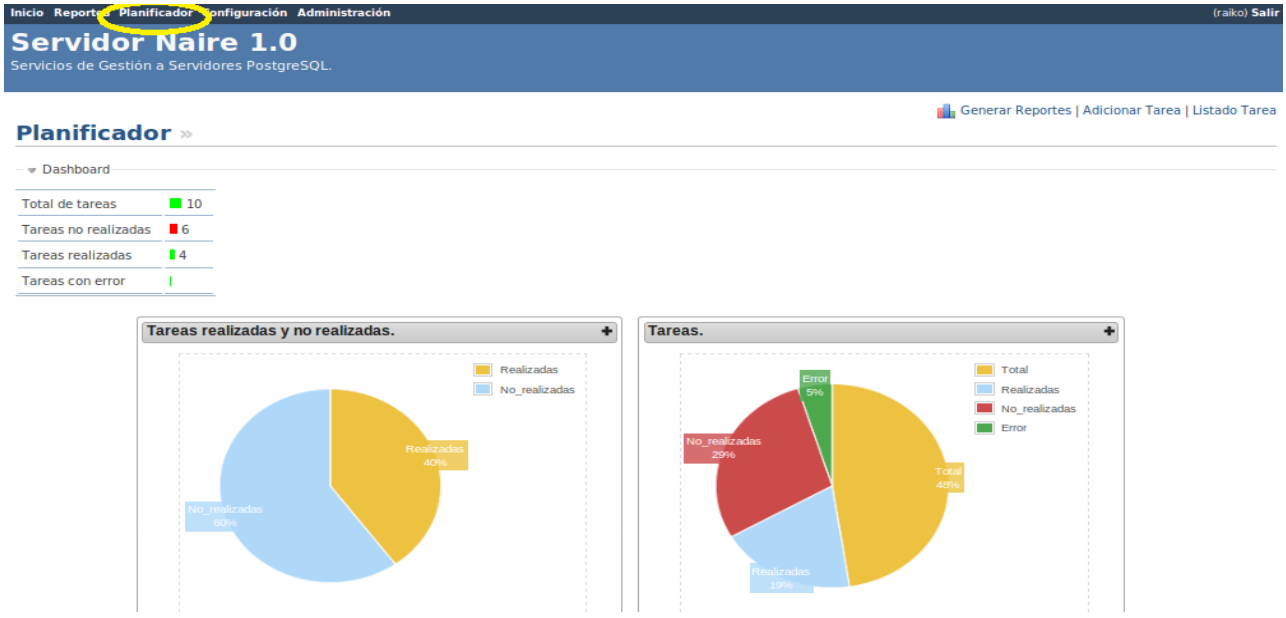
2.3.4.4 Historia de usuario – Mostrar estadísticas

Historia de Usuario	
Número: 5	**Nombre de la Historia de Usuario:** Mostrar estadísticas
Cantidad de modificaciones a la Historia de Usuario: 4	
Usuario: Yudelki Driggs Aguilera Leannys Estela Fernández Cabrera	**Iteración asignada:** 1
Prioridad en negocio: Alto	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Alto	**Puntos reales:** 1 semana

Descripción: Permite mostrar en el Dashboard las estadísticas de las tareas que están realizadas, pendientes y las que no se realizaron por algún error en datos.

Prototipo de interfaces:

1- Seleccionar la opción Planificador.



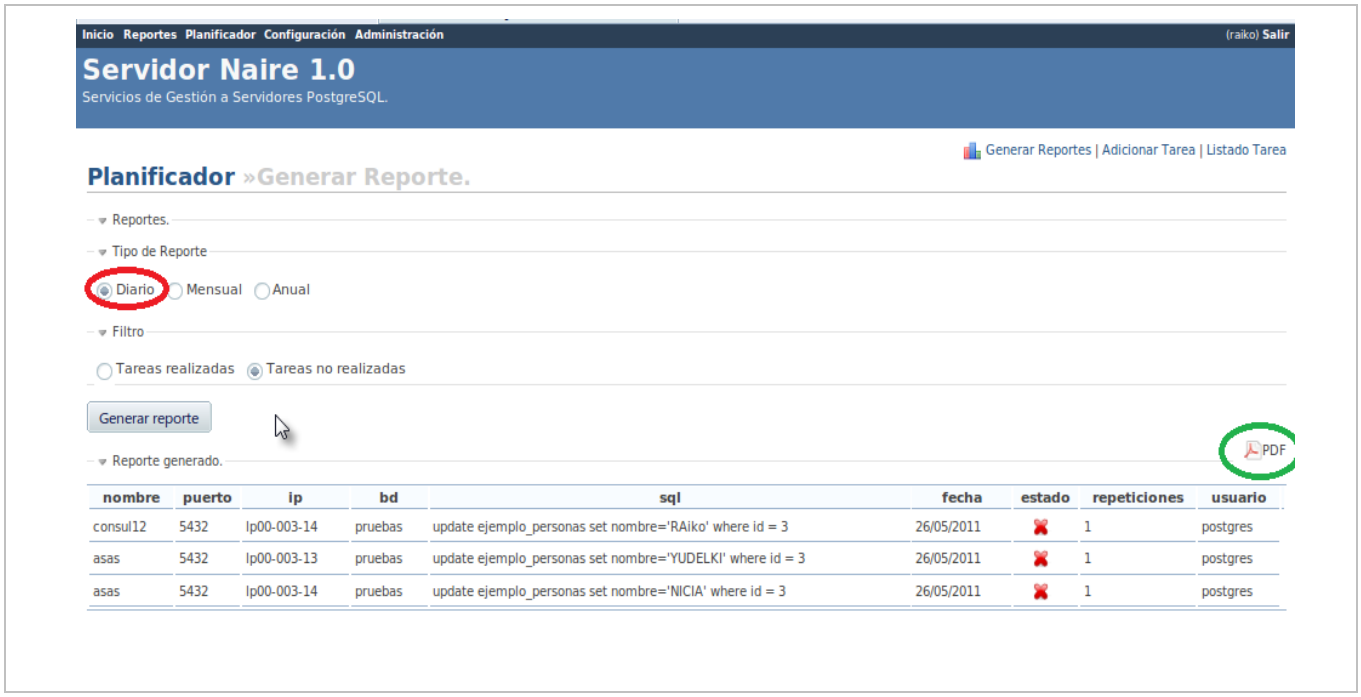
2.3.4.5 Historia de usuario – Listar por usuario

Historia de Usuario	
Número: 6	**Nombre de la Historia de Usuario: Listar tareas por usuario**
Cantidad de modificaciones a la Historia de Usuario: 2	
Usuario: Yudelki Driggs Aguilera	**Iteración asignada:** 2
Prioridad en negocio: Alto	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1 semana
Descripción: Permite ver las tareas que se han realizado y las que están pendientes insertadas por un usuario determinado. Se selecciona el usuario del cual se quieren observar las tareas y se da clic en el botón “buscar”.	
Prototipo de interfaces: 1- Seleccionar el usuario y hacer clic en la opción Buscar. 	

nombre	puerto	ip	bd	sql	fecha	estado	repeticiones	usuario
consulta	5423	10.7.3.14	prueba	update ejemplo_prueba nombre = 'cuna' where id = 1	17/05/2011	✗	1	raiko
consulta15	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='yudisney' where id = 3	12/05/2011	✗	2	raiko
consulta7	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='P' where id = 6	12/05/2011	✓	1	raiko
consul12	5432	lp00-003-14	pruebas	update ejemplo_personas set nombre='RAiko' where id = 3	26/05/2011	✗	1	raiko
consulta6	5432	10.7.3.14	ejemplo	update ejemplo_personas set nombre='pablo' where id = 9	11/05/2011	✓	1	raiko
consulta2	5432	10.7.3.15	ejemplo	update ejemplo_personas set nombre='Elbvira' where id = 7	11/05/2011	✓	1	raiko
consulta4	5432	10.7.3.20	ejemplo	update ejemplo_personas set nombre='yixoooo' where id = 6	11/05/2011	✗	1	raiko

2.3.4.6 Historia de usuario – Generar reporte por día

Historia de Usuario	
Número: 10	**Nombre de la Historia de Usuario: Generar reporte por día**
Cantidad de modificaciones a la Historia de Usuario: 3	
Usuario: Yudelki Driggs Aguilera	**Iteración asignada:** 3
Prioridad en negocio: Alto	**Puntos estimados:** 2 semanas
Riesgo en desarrollo: Medio	**Puntos reales:** 2 semanas
Descripción: Genera un reporte de todas las tareas que se han realizado en el día y las que están pendientes también, el formato del mismo será en PDF. Se selecciona la opción “Generar reporte por día”, se insertan los datos necesarios en este caso día, mes y año, se muestran en un documento PDf las tareas correspondientes a la fecha.	
Prototipo de interfaces: 1- Seleccionar tipo de reporte. 2- Seleccionar tipo de filtro y hacer clic en la opción Generar reporte.	



2.5 Fase de Planificación. Definición

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. Es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario y, asociadas a éstas, las entregas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (Beck 2000).

La planificación no debe ser estricta puesto que hay muchas variables en juego, debe ser flexible para poder adaptarse a los cambios que puedan surgir. Una buena estrategia es hacer planificaciones detalladas para unas pocas semanas y planificaciones mucho más abiertas para unos pocos meses.

2.5.1. Estimación del esfuerzo por historia de usuario

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar cada historia de usuario. Este se expresa utilizando como medida el punto. Un punto se considera como una semana ideal de trabajo donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción (Beck 2000).

Los resultados obtenidos en esta estimación se exponen en la siguiente tabla:

Tabla 2.2: Plan de estimación de esfuerzo por historias de usuario.

Historia de usuario	Puntos estimados
Insertar tarea	1
Eliminar tarea	1
Modificar tarea	1
Verificar tarea	1
Mostrar estadísticas	1
Re planificar tarea	2
Listar por usuario	1
Listar por base dato	1
Listar por tareas realizadas	1
Listar por tareas pendientes	1
Generar reporte por día	2
Generar reporte por año	1
Generar reporte por mes	1
Total	**15**

2.5.2. Plan de iteraciones

Las HU seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada HU se traduce en tareas específicas de programación. Así mismo, para cada HU se establecen las pruebas de aceptación.

Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir. Además de las entregas habituales al final de cada iteración, todas las semanas se hace un resumen con la presencia del cliente y se muestran los

adelantos del sistema y con el mismo objetivo se realizan reuniones diarias entre los desarrolladores al final de cada sesión de trabajo. Las iteraciones se detallan a continuación:

2.5.2.1. Iteración 1

En la primera iteración se implementan las HU con mayor prioridad, obteniendo al final de la misma una primera versión de prueba y dando al sistema las primeras funcionalidades.

2.5.2.2. Iteración 2

En la segunda iteración se realizará la implementación de las HU con prioridad de negocio alta. Además, se corregirán errores o inconformidades del usuario con las HU implementadas en la iteración anterior. Esta segunda versión será mostrada a los clientes con el único objetivo de realizar cambios en base a la aceptación del mismo.

2.5.2.3. Iteración 3

En la tercera iteración, ya implementadas las funcionalidades especificadas, se realiza el desarrollo de las últimas historias de usuario con prioridad. De esta manera se obtiene la versión 1.0 del producto final.

2.5.3. Plan de duración de las iteraciones

Para una mayor organización del trabajo como lo plantea el ciclo de vida de XP se crea un plan de duración de las iteraciones, en este caso se realizaría un solo plan ya que existe un único equipo de desarrolladores.

Este plan se realiza con el objetivo de reflejar cuáles serán las historias de usuario que se implementarán en cada una de las iteraciones, así como el tiempo destinado a cada una de ellas y el orden en que se implementarán, lo que ayuda a obtener una idea general del tiempo que durará la confección total del sistema.

Tabla 2.3: Plan de iteraciones.

Iteraciones	Orden de las HU a implementar	Duración total de la iteración
Iteración # 1	Insertar tarea Eliminar tarea Modificar tarea Verificar tarea Mostrar estadísticas Re-planificar tarea	7 semanas
Iteración # 2	Lista por usuario Lista por BD Lista por tareas realizadas Lista por tareas pendientes	4 semanas
Iteración # 3	Generar reporte por día Generar reporte por mes Generar reporte por año	4 semanas

2.5.4. Plan de entregas

En el plan de entrega que se plantea a continuación se hace una propuesta de la fecha aproximada en que se harán liberaciones (releases) al sistema al finalizar cada iteración en la fase de implementación.

Tabla 2.4: Plan de entregas.

Liberación	Descripción de la iteración	Orden de la HU a implementar	Duración total

1	Esta iteración tiene como objetivo darle cumplimiento a las HU insertar tarea, eliminar tarea, modificar tarea, verificar tarea, mostrar estadísticas y re-planificar tarea, que serán las de vital importancia para el planificador, pues son las que inician las labores de planificación.	Insertar tarea Eliminar tarea Modificar tarea Verificar tarea Mostrar estadísticas Re-planificar tarea	7 semanas
2	El objetivo de esta iteración darle cumplimiento a las HU Lista por usuario, Lista por BD, Lista por tareas realizadas, Lista por tareas pendientes, que son de gran importancia para el planificador pues son los requisitos que nos muestran el estado de realización de las tareas.	Lista por usuario Lista por BD Lista por tareas realizadas Lista por tareas pendientes	4 semanas
3	Esta iteración está centrada en desarrollar las HU Generar reporte por día, Generar reporte por mes, Generar reporte por año, Generar reporte por semana, Generar reporte por intervalos personalizados.	Generar reporte por día Generar reporte por mes Generar reporte por año	4 semana

2.6 Tarjetas CRC

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración).

Con el objetivo de hacer entendible las funcionalidades encapsuladas en estos módulos, se define una tarjeta CRC por cada una de las historias de usuario, con la finalidad de obtener un diseño simple y no incurrir en la implementación de características que no son necesarias.

Para un mejor entendimiento de la aplicación se muestra un diagrama de clases que representa la estructura de la herramienta.

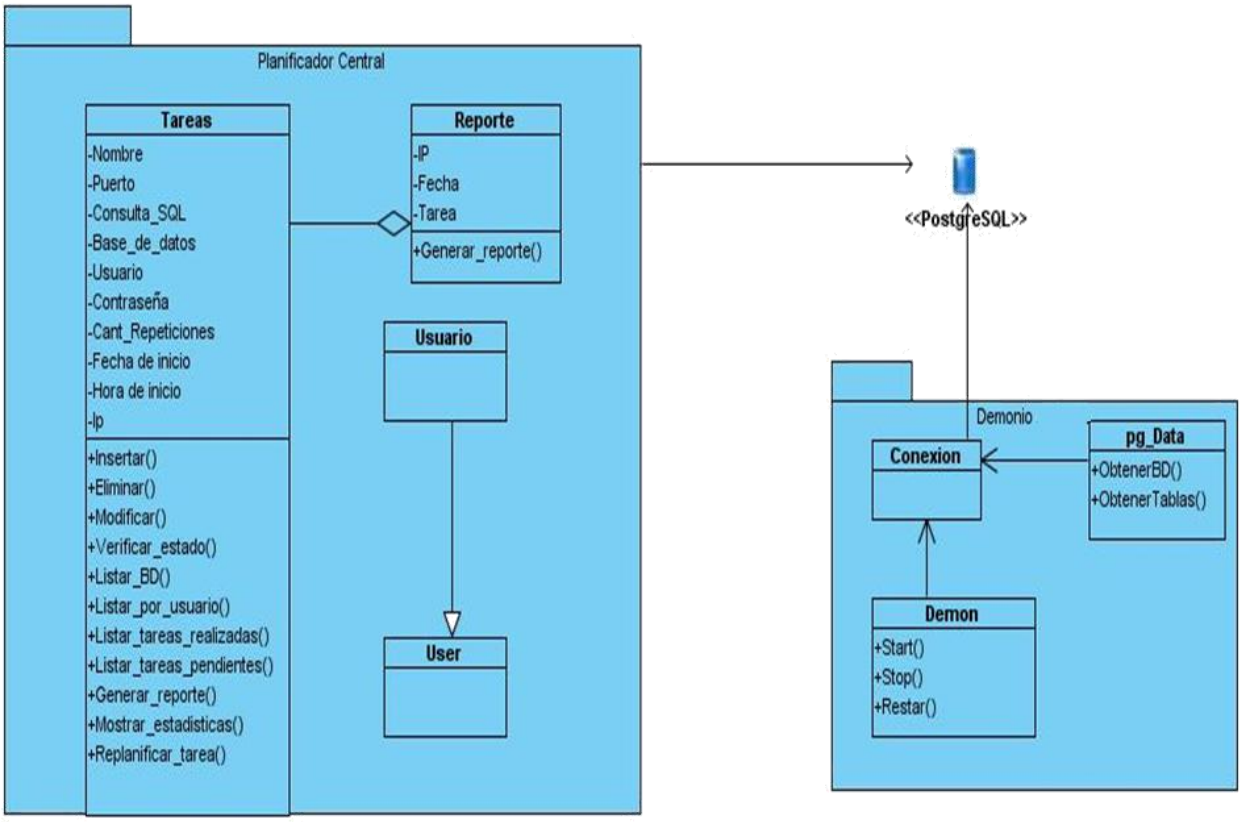


Figura 1: Diagrama de clases

2.7 Diseño de la base de datos

Diseñar la BD es algo que no se puede pasar por alto, producto de que uno de sus objetivos fundamentales es brindar la seguridad en el almacenamiento de la información a gestionar. A continuación se muestra el modelo de datos que se utilizó:

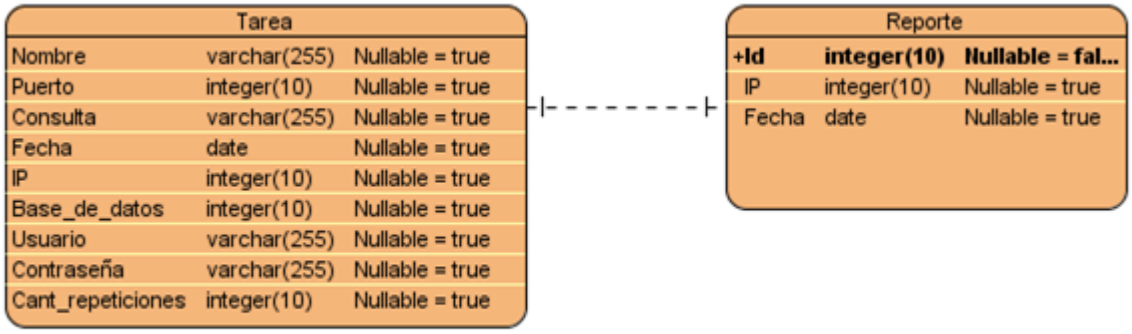


Figura 2: Diseño de la BD.

CONCLUSIONES DEL CAPÍTULO

En el capítulo se identificaron 13 requisitos funcionales de los cuales se generaron 13 historias de usuario y de ellas 5 significativas, se describieron las clases más importantes dentro del contexto del sistema. Cada historia de usuario posee un tiempo de realización el cual define la duración real de la implementación, para la herramienta se estimó un tiempo de 15 semanas. Con la finalidad de obtener un diseño simple se definió una tarjeta CRC por cada una de las historias de usuario para un total de 13 tarjetas. Se utilizó el patrón arquitectónico Modelo-Vista-Controlador que implementa Django para la implementación del software, así como varios patrones de diseño: Creador, Experto y Controlador.

CAPÍTULO 3: IMPLEMENTACION Y PRUEBAS

Introducción

La Metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para incrementar la visión de los desarrolladores con la opinión de éste. En el presente capítulo se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiéndose las tareas generadas por cada historia de usuario.

Para una producción efectiva XP plantea una serie de prácticas básicas que deben seguirse al pie de la letra para el desarrollo exitoso de un proyecto. Éstas se mencionan a continuación (Fowler 2002):

- La planeación, en la cual la opinión del cliente y del equipo de desarrollo debe fusionarse como un todo coherente. El cliente y los programadores negocian el alcance del proyecto para cada iteración, esta etapa es conocida como “Juego de Planificación”. El factor crítico es permitir al cliente tomar las decisiones de negocio y al equipo de desarrollo tomar las decisiones técnicas.
- Entregas en iteraciones pequeñas, que permitan al cliente utilizar el sistema con las funcionalidades mínimas lo antes posible, e irlo complementando gradual y continuamente.
- Manejo de metáforas, la misma ayuda a que todo el equipo de desarrollo entienda los elementos básicos del sistema y las relaciones entre ellos sin la necesidad de una arquitectura muy elaborada y detallada.
- Diseño simple, donde siempre se intenta tener el código más simple, menos redundante y con las funcionalidades estrictamente necesarias en el presente.
- Pruebas continuas, donde es imperativo que los programadores escriban pruebas por cada unidad de código y que el cliente participe en el diseño de pruebas funcionales.
- Refactorización, donde este concepto se refiere a mantener una depuración y simplificación constante del sistema. Una vez que se ha añadido alguna funcionalidad, es necesario revisar y ser críticos para encontrar puntos de simplificación del código.
- Programación en pares, donde toda la codificación debe hacerse en parejas de programadores, cada pareja compartiendo un mismo monitor y teclado. El que utiliza el teclado piensa en la mejor manera de implementar alguna funcionalidad, el otro piensa estratégicamente,

cuestionando si se puede simplificar, anticipando pruebas, o preguntándose si el enfoque es el adecuado o si debe descartarse código y replantear el problema. Se alienta la rotación continua de programadores en los pares, combinando diferentes niveles de experiencia y de conocimiento del código.

- Propiedad colectiva del código. El código puede ser modificado por cualquier elemento del equipo de programación, esto es, que no hay propiedad individual de algún programador sobre alguna sección de código. Se asume que al seguir las prácticas de XP, después de un tiempo razonable, cualquier programador conoce todo el código, principalmente por el hecho de la programación en pares.
- Integración continua (mantener una sola revisión para todo el equipo de programación), tan pronto como pequeños “saltos” en la versión actual sean concluidos. Se recomiendan lapsos entre integraciones de pocas horas a no más de un día.
- Semana de 40 horas. XP afirma que las condiciones de trabajo óptimas para programar “a máxima velocidad” es solo en un turno normal (8 horas). En este sentido las horas extras o fines de semana trabajados solo desgastan al equipo de programación, afectan el rendimiento y generan un ambiente propicio para cometer errores.
- El cliente debe estar disponible localmente, donde un representante del cliente debe permanecer por turnos completos en el sitio de programación para contestar cualquier pregunta y ayudar en el desarrollo de pruebas funcionales.
- Mantener estándares de codificación entre los programadores. Esto es esencial para la programación en pares y para la propiedad colectiva del código.

3.1 Implementación

Para llevar a cabo la correcta implementación de los módulos definidos en la propuesta de solución, con las consiguientes HU que se incluyen en cada uno de ellos, se deben definir por parte del equipo de desarrollo las tareas que serán llevadas a cabo. Lo expuesto anteriormente permite a los desarrolladores obtener un nivel de detalle más avanzado que el sugerido por las HU. A continuación se describen las tareas específicas para el desarrollo de la presente herramienta surgidas a partir de la técnica denominada “tormenta de ideas”.

Iteración 1

En esta iteración se implementaron las historias de usuario que conforman la estructura básica del sistema. A partir de ellas se realizaron las demás funcionalidades requeridas por la aplicación.

Tabla 3.1 Historias abordadas en la primera iteración.

No.	Historia de usuario	Estimación	Real
1	Insertar tarea	1	1
2	Eliminar tarea	1	1
3	Modificar tarea	1	1
4	Verificar tarea	1	1
5	Mostrar estadísticas	1	1
6	Re-planificar tarea	2	2
Total		7	7

3.1.1 Tareas efectuadas por las HU presentes en esta iteración 1.

Tabla 3.2 Tarea de HU “Insertar tarea”.

Tarea	
Numero de tarea : 1	Numero de HU: 1
Nombre de Tarea: Añadir información a la BD	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 2/enero	Fecha de fin : 4/ enero
Programador responsable: Yudelki Driggs Aguilera	
Descripción: Después de llenar los campos correspondiente se procede a guardar la información en la BD del planificador.	

Tabla 3.3 Tarea de HU “Eliminar tarea”.

Tarea	
Numero de tarea : 1	Numero de HU: 3

Nombre de Tarea: Añadir campos	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 5/ enero	Fecha de fin : 8/ enero
Programador responsable: Yudelki Driggs Aguilera / Leannys Estela Fernández Cabrera	
Descripción: Se listan cada una de las tareas de la BD, se crea un chekbox para seleccionar la tarea a eliminar.	

Tabla 3.4 Tarea de HU “Modificar tarea”.

Tarea	
Numero de tarea : 1	Numero de HU: 2
Nombre de Tarea: Añadir campos	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 10/enero	Fecha de fin : 13/ enero
Programador responsable: Leannys Estela Fernández Cabrera	
Descripción: Se crea un botón donde se ejecuta el método buscar tarea, se crean varios campos con los atributos a modificar.	

Tabla 3.5 Tarea de HU “Verificar tarea”.

Tarea	
Numero de tarea : 1	Numero de HU: 4
Nombre de Tarea: Conectar con la BD	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 15/ enero	Fecha de fin : 19/ enero
Programador responsable: Yudelki Driggs Aguilera / Leannys Estela Fernández Cabrera	
Descripción: Se va a llevar a cabo la conexión a la BD para extraer la información necesaria.	

Tabla 3.6 Tarea de HU “Mostrar estadística”.

Tarea	
Numero de tarea : 1	Numero de HU: 5
Nombre de Tarea: Crear gráficos	

Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 21/ enero	Fecha de fin : 28/ enero
Programador responsable: Yudelki Driggs Aguilera / Leannys Estela Fernández Cabrera	
Descripción: Se van a tomar los datos de la BD del planificador y se van a crear las estadísticas en un grafico de pastel.	

Tabla 3.7 Tarea de HU "Re-planificar tarea".

Tarea	
Numero de tarea : 1	Numero de HU: 15
Nombre de Tarea: Introducir datos	
Tipo de tarea: Desarrollo	Punto de estimación: 2
Fecha de inicio : 2/ febrero	Fecha de fin : 15/ febrero
Programador responsable: Leannys Estela Fernández Cabrera	
Descripción: Se introducen los datos de los campos "cantidad de repeticiones", "hora" y "fecha".	

Iteración 2

Durante el desarrollo de esta iteración se implementaron las HU con prioridad de negocio alta Además, se corrigieron los errores o inconformidades del usuario con las HU implementadas en la iteración anterior.

Tabla 3.8 Historias abordadas en la segunda iteración.

No.	Historia de usuario	Estimación	Real
7	Lista por usuario	1	1
8	Lista por BD	1	1
9	Lista por tareas realizadas	1	1
10	Lista por tareas pendientes	1	1
Total		4	4

3.1.2 Tareas efectuadas por las HU presentes en la iteración 2.

Tabla 3.9 Tarea de HU “Listar por usuario”.

Tarea	
Número de tarea : 1	Número de HU: 6
Nombre de Tarea: Seleccionar usuario	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 18/ febrero	Fecha de fin : 23/ febrero
Programador responsable: Yudelki Driggs Aguilera	
Descripción: Se selecciona el usuario para mostrar las tareas insertadas por él.	

Tabla 3.10 Tarea de HU “Listar por BD”.

Tarea	
Numero de tarea : 1	Numero de HU: 7
Nombre de Tarea: Seleccionar BD	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 25/ febrero	Fecha de fin : 1/ marzo
Programador responsable: Leannys Estela Fernández Cabrera	
Descripción: Se selecciona la BD para ver las tareas que se le han asignado.	

Tabla 3.11 Tarea de HU “Listar por tareas realizadas”.

Tarea	
Numero de tarea : 1	Numero de HU: 8
Nombre de Tarea: Buscar tareas	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 3/ marzo	Fecha de fin : 10/ marzo
Programador responsable: Leannys Estela Fernández Cabrera	
Descripción: Se busca en la BD del planificador todas las tareas cuyo estado sea “realizada”.	

Tabla 3.12 Tarea de HU “Listar por tareas pendientes”.

Tarea	
Numero de tarea : 1	Numero de HU: 9
Nombre de Tarea: Buscar tareas	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 11/ marzo	Fecha de fin : 13/ marzo
Programador responsable: Yudelki Driggs Aguilera	
Descripción: Se busca en la BD del planificador todas las tareas cuyo estado sea “pendiente”.	

Iteración 3

En la tercera iteración, ya implementadas las funcionalidades especificadas, se realiza el desarrollo de las últimas historias de usuario con prioridad.

Tabla 3.13 Historias abordadas en la tercera iteración.

No.	**Historia de usuario**	**Estimación**	**Real**
11	Generara reporte por día.	2	2
12	Generara reporte por mes.	1	1
13	Generara reporte por año.	1	1
Total		4	4

3.1.3 Tareas efectuadas por las HU presentes en la iteración 3.

Tabla 3.14 Tarea de HU “Generar reporte por día”.

Tarea	
Numero de tarea : 1	Numero de HU: 10
Nombre de Tarea: Insertar fecha	
Tipo de tarea: Desarrollo	Punto de estimación: 2
Fecha de inicio : 15/ marzo	Fecha de fin : 25/ marzo
Programador responsable: Yudelki Driggs Aguilera	

Descripción: Se inserta el día con mes y año para generar el reporte de las tareas que se insertaron y se realizaron en dicha fecha.

Tabla 3.15 Tarea de HU “Generar reporte por mes”.

Tarea	
Numero de tarea : 1	Numero de HU: 11
Nombre de Tarea: Generar pdf	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 27/ marzo	Fecha de fin : 1/ abril
Programador responsable: Leannys Estela Fernández Cabrera	
Descripción: Se genera el reporte de las tareas que se han insertado en el mes así como las que se han realizado.	

Tabla 3.16 Tarea de HU “Generar reporte por año”.

Tarea	
Numero de tarea : 1	Numero de HU: 13
Nombre de Tarea: Generar pdf	
Tipo de tarea: Desarrollo	Punto de estimación: 1
Fecha de inicio : 3/ abril	Fecha de fin : 7/ abril
Programador responsable: Yudelki Driggs Aguilera	
Descripción: Se genera el reporte de las tareas que se han insertado en el año así como las que se han realizado.	

3.2 Prueba

Uno de los pilares de la Extreme Programming es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas, reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

Para XP las pruebas del sistema que se utilizarán son las pruebas funcionales destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñada por el cliente final. Este tipo de pruebas, constituyen uno de los mayores avances en la programación desde la aparición de la orientación a objetos y aunque son recomendadas, su uso o no, queda a merced de los desarrolladores.

3.2.1 Pruebas funcionales

Las pruebas funcionales son la mejor forma de probar la aplicación completa: desde la petición realizada por un navegador hasta la respuesta enviada por el servidor. Son muy similares a lo que se hace manualmente cada vez que se añade o modifica una acción y se prueban dichos cambios para comprobar que todo funciona bien y que todos los elementos se muestran correctamente. En otras palabras, lo que se hace es probar un escenario correspondiente a la historia de usuario que se acaba de implementar en la aplicación. Una historia de usuario no se considera completa hasta que no ha pasado por sus pruebas de aceptación (Beck 2000).

A continuación se muestran las pruebas realizadas al componente planificador de tareas para las HU, las tablas que contienen los artefactos generados en estas pruebas contaran con los siguientes campos:

Escenario: Nombre del escenario a probar

Descripción: Este campo contiene la descripción del escenario de prueba.

Variables (Vn): este campo contiene los valores de los atributos que posee la HU. Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Respuesta del sistema: Se escribe el resultado que se espera al realizar la prueba.

Flujo central: Pasos a desarrollar para probar la funcionalidad que se indicó.

3.2.2 Casos de prueba.

Tabla 3.19 Caso de prueba de aceptación de las HU Insertar tarea, Eliminar tarea y Modificar tarea.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Variable 6	Variable 7	Variable 8	Variable 9	Variable 10	Respuesta del sistema	Flujo central
EC 1.1 Registrar tarea	Permite agregar una nueva tarea	v	v	v	v	v	v	v	v	v	v	El sistema muestra un mensaje de éxito.	Seleccionar Planificador/ Adicionar Tarea: 1- El usuario selecciona el servidor. 2- El usuario presiona el botón (representado por el ícono que está al lado del servidor seleccionado). 3- El usuario introduce el nombre de la
EC 1.2 Registrar tarea con datos incorrectos	No debe permitir registrar una tarea	i	v	v	v	v	v	v	v	v	v	El sistema muestra un mensaje de error.	
		v	i	v	v	v	v	v	v	v	v		
		v	v	i	v	v	v	v	v	v	v		
		v	v	v	i	v	v	v	v	v	v		
		v	v	v	v	i	v	v	v	v	v		
		v	v	v	v	v	i	v	v	v	v		
		v	v	v	v	v	v	i	v	v	v		
		v	v	v	v	v	v	v	i	v	v		

		v	v	v	v	v	v	v	v	i	v		tarea, el puerto, selecciona el servidor de base de datos, introduce el nombre de **usuario, la** contraseña, la hora de ejecución de la **tarea, la fecha** y la consulta. 4-El usuario presiona el botón Adicionar.
		v	v	v	v	v	v	v	v	v	i		
EC 1.3 Modificar tarea	Permite modificar los datos de una tarea	N/A	v	v	v	v	v	v	v	v	v	El sistema muestra mensaje de éxito.	1- El usuario introduce el nombre de la tarea, el puerto, **selecciona el** servidor de base de datos, **introduce el** nombre de usuario, la contraseña, la hora de
EC 1.4 Modificar tarea con datos incorrectos	No debe permitir modificaciones a una tarea	N/A	i	v	v	v	v	v	v	v	v	El sistema muestra un mensaje de error	
		N/A	v	i	v	v	v	v	v	v	v		
		N/A	v	v	i	v	v	v	v	v	v		
		N/A	v	v	v	i	v	v	v	v	v		
		N/A	v	v	v	v	i	v	v	v	v		

		N/A	v	v	v	v	v	i	v	v	v		ejecución de la tarea, la fecha y la consulta. 2- El usuario presiona el botón Modificar tarea.
		N/A	v	v	v	v	v	v	i	v	v		
		N/A	v	v	v	v	v	v	v	i	v		
		N/A	v	v	v	v	v	v	v	v	i		
EC 1.5 Eliminar tarea	Permite eliminar una tarea de la planificación	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	El sistema muestra un mensaje de éxito	Seleccionar Planificador/Listado de tareas: 1- El usuario presiona el botón Eliminar. 2- El sistema muestra un mensaje de confirmación. 3- El usuario presiona el botón Aceptar.
EC 1.6 Mostrar listado de tareas	Permite visualizar al usuario todas las tareas planificadas	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	El sistema muestra listado de las tareas	1- El usuario selecciona la opción Listado de tareas. 2- El sistema muestra un listado con los datos: nombre, puerto, ip, bd, consulta, inicio,

Tabla 3.23 Caso de prueba de aceptación de la HU Mostrar estadísticas.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Se muestran las gráficas de las estadísticas.	El sistema debe mostrar las estadísticas de ejecución de las tareas.	Se muestran las estadísticas de las tareas realizadas y pendientes correctamente.	Seleccionar Planificador: 1-Se muestran las gráficas de ejecución de las tareas.

Tabla 3.24 Caso de prueba de aceptación de la HU Listar por usuario.

Escenario	Descripción	V1	Respuesta del sistema	Flujo central
EC 1.1 Listar las tareas por usuario.	El sistema debe mostrar el listado de las tareas realizadas y que se encuentran pendientes dado un usuario.	v	EL sistema muestra el listado de tareas.	Seleccionar Planificador/Listado tarea/Tareas por usuario: 1- Se selecciona el usuario del que se desea listar las tareas. 2- El usuario **presiona el** botón buscar.
EC 1.n Listar las tareas por usuario con datos incorrectos	El sistema no debe permitir mostrar el listado de las tareas realizadas y que se encuentran pendientes dado un usuario.	i	EL sistema no muestra las tareas.	

Tabla 3.25 Caso de prueba de aceptación de la HU Listar por BD.

Escenario	Descripción	V1	Respuesta del sistema	Flujo central
EC 1.1 Listar las tareas por base de datos	El sistema debe mostrar el listado de las tareas realizadas y que se encuentran pendientes dada una base de datos.	v	EL sistema muestra el listado de tareas.	Seleccionar Planificador/Listado tarea/Tareas por Base de Datos: 1- Se selecciona la base de datos de la cual se desea listar las tareas. 2- El usuario presiona el botón buscar.
EC 1.n Listar las tareas por base de datos con datos incorrectos	El sistema no debe permitir mostrar el listado de las tareas realizadas y que se encuentran pendientes dada una base de datos.	i	EL sistema no muestra las tareas.	

Tabla 3.26 Caso de prueba de aceptación de la HU Listar por tareas realizadas.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar las tareas realizadas	El sistema debe permitir ver el listado de tareas realizadas.	Se muestra el listado de tareas realizadas	Seleccionar Planificador/Listado tarea/Tareas Realizadas: 1- Se muestra el listado de tareas realizadas

Tabla 3.27 Caso de prueba de aceptación de la HU Listar por tareas pendientes.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Listar las tareas pendientes	El sistema debe permitir ver el listado de tareas pendientes.	Se muestra el listado de tareas pendientes.	Seleccionar Planificador/Listado tarea/Tareas no Realizadas: 1- Se muestra el listado de tareas pendientes.

Tabla 3.28 Caso de prueba de aceptación de la HU Generar reporte por día.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte de las tareas que se han insertado en el día.	El sistema debe permitir mostrar las tareas que se han insertado en la base de datos en el día, tanto realizadas como pendientes.	Se muestra correctamente el reporte de las tareas.	Seleccionar Planificador/Generar Reportes 1- El usuario selecciona la opción de reporte diario. 2- El usuario presiona el botón generar reporte.

Tabla 3.29 Caso de prueba de aceptación de la HU Generar reporte por mes.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte de las tareas que se han insertado en el mes.	El sistema debe permitir mostrar las tareas que se han insertado en la base de datos en el mes, tanto realizadas como pendientes.	Se muestra correctamente el reporte de las tareas.	Seleccionar Planificador/Generar Reportes: 1- El usuario selecciona la opción de reporte mensual. 2- El usuario presiona el botón generar reporte.

Tabla 3.30 Caso de prueba de aceptación de la HU Generar reporte por año.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Generar reporte de las tareas que se han insertado en el año.	El sistema debe permitir mostrar las tareas que se han insertado en la base de datos en el año, tanto realizadas como pendientes	Se muestra correctamente el reporte de las tareas.	Seleccionar Planificador/Generar Reportes: 1- El usuario selecciona la opción de reporte anual. 2- El usuario presiona el botón generar reporte

Tabla 3.33 Caso de prueba de aceptación de la HU Re-planificar tarea.

Escenario	Descripción	V 1	V 2	Respuesta del sistema	Flujo central
EC 1.1 Re-planificar las tareas	El sistema debe permitir modificar la hora y fecha de ejecución ala tarea.	v	v	El sistema muestra un mensaje de éxito.	Seleccionar Planificador/Listado tarea/Presionar en la tareas que se desea re-planificar el ícono que representa esta opción: 1- Se insertan los datos de la hora y la fecha de la tarea. 2- Se presiona el botón Enviar consulta.
EC 1.n Re-planificar las tareas con datos incorrectos	El sistema no debe permitir modificar la hora y fecha de ejecución ala tarea.	i	v	El sistema muestra un mensaje de error	
		v	i		

Se realizaron las pruebas funcionales que brindarán al cliente conformidad y seguridad ante las funcionalidades del sistema, se generó un caso de prueba por cada HU arrojando los resultados que se presentan en el gráfico siguiente:

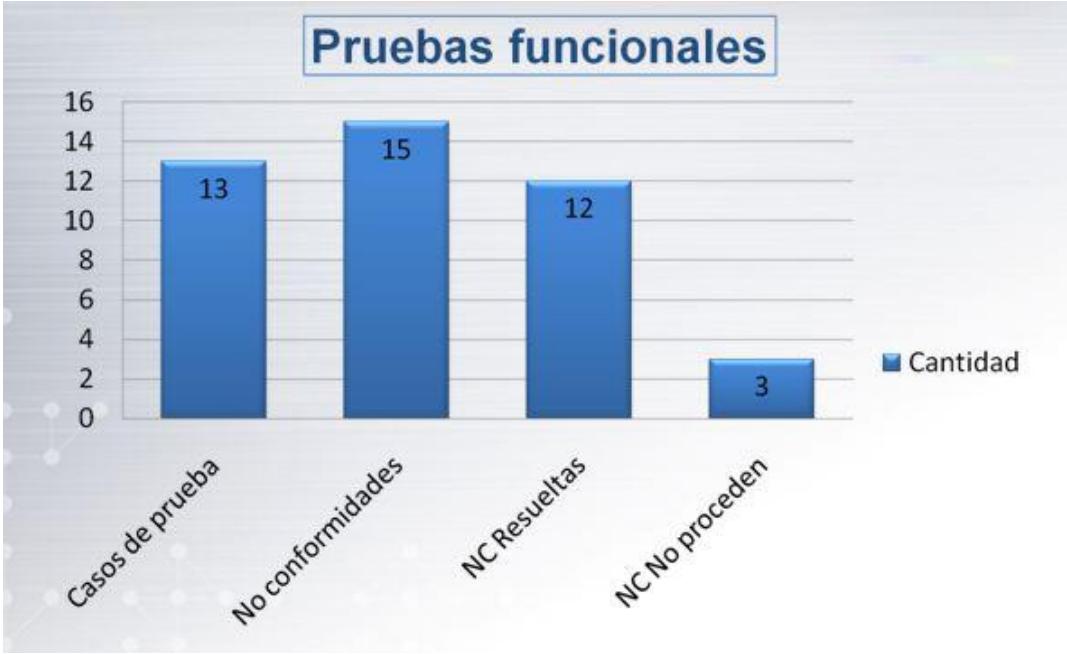


Figura 6 Pruebas funcionales

CONCLUSIONES DEL CAPÍTULO

En el capítulo se detallaron las 59 tareas que fueron generadas por las historias de usuarios para llevar a cabo la correcta implementación de los módulos de la herramienta. Se realizaron las pruebas funcionales generando un caso de prueba por cada HU para un total de 13 casos de prueba, se detectaron 15 no conformidades de las cuales 12 fueron resueltas. Se implementó la herramienta capaz de planificar tareas de forma web.

CONCLUSIONES GENERALES

La herramienta que se elaboró como resultado de la presente investigación constituye un importante aporte al Centro de Tecnologías de Gestión de Datos (DATEC), permitiendo la planificación de tareas para el Servidor de Gestión de PostgreSQL. A raíz del estudio realizado en dicha investigación se arribaron a las siguientes conclusiones:

- Se analizaron los SGBD Oracle, DB2 pero son gestores propietarios por lo cual se decidió implementar una nueva herramienta que satisfaga las necesidades de planificación del proyecto.
- Se identificaron 13 requisitos funcionales de los cuales se generaron 15 historias de usuario y de ellas 6 significativas.
- Se diseñó la solución utilizando el patrón arquitectónico Modelo-Vista-Controlador que implementa Django (MTV) para la implementación del software, así como varios patrones de diseño: Creador, Bajo Acoplamiento y Alta Cohesión que mejoran la eficiencia del proceso de codificación.
- Se implementó el demonio encargado de ejecutar las tareas en cada una de las PC clientes, además de un módulo web que realiza la planificación de las tareas en el Servidor de Gestión de PostgreSQL.
- Se realizaron las pruebas funcionales que brindarán al cliente conformidad y seguridad ante las funcionalidades del sistema, se generó un caso de prueba por cada HU para un total de 13 casos de prueba, se detectaron 15 no conformidades de las cuales 12 fueron resueltas y 3 no proceden.

Por todo lo expuesto anteriormente se concluye que los objetivos propuestos fueron cumplidos satisfactoriamente.

RECOMENDACIONES

Como resultado del proceso de investigación y realización de la herramienta han surgido ideas que serían recomendables tener en cuenta para un futuro perfeccionamiento del sistema, a continuación se listan las mismas:

- Continuar haciendo las pruebas funcionales, para certificar la eficiencia del sistema desarrollado, con el fin de elevar la calidad del producto.

- Realizar un estudio para determinar si pueden ser adicionadas nuevas funcionalidades de interés para beneficio de la administración del Centro de Tecnologías de Gestión de Datos (DATEC).

REFERENCIAS BIBLIOGRÁFICAS

- Alvarez, M. A. (2007) "Desarrollo Web." **Volumen**, 40 DOI:
- Aptana (2008) "Polargeek." **Volumen**, DOI:
- Beck (2000). Extreme Programming Explained.
- Clemente, C. (2007) "Python Software de Comunicación." **Volumen**, 50 DOI:
- Django (2007). "Python-Django."
- EnterpriseDB (2009) "PgAdmin." **Volumen**, DOI:
- Fowler, M. (2002) "Progamacion Extrema." **Volumen**, DOI:
- Fuh, G. (2008) "Oracle." Sitio Oficial de oracle **Volumen**, 25 DOI:
- García, M. (1999). Diseño de Bases de Datos.
- Goodstein, L. (1998). Planeación Estratégica Aplicada. México.
- Graells, D. P. M. (2008) "Grandes aportes de las TIC." **Volumen**, 24 DOI:
- Group, p. G. D. (2009) "PostgreSQL." **Volumen**, 20 DOI:
- Ivar Jacobson, G. B., James Rumbaugh (2000). El Proceso Unificado de Desarrollo de Software Madrid.
- Jiménez (1982). Introducción al estudio de la teoría administrativa. México.
- Juankar (2009) "Planificador de tareas." **Volumen**, 23 DOI:
- Kimmel, B. (2002) "Metodología XP." **Volumen**, 50 DOI:
- Mendoza, M. A. (2004) "Extreme Programming." **Volumen**, DOI:
- Orallo, E. H. (2005) "Lenguaje unificado de modelado UML." **Volumen**, DOI:
- PostgreSQL, C. T. C. d. (2007) "PostgreSQL." **Volumen**, 12 DOI:
- Python, S. o. d. (2007) "Python." **Volumen**, DOI:
- Rell, T. (2005) "Servicios de Ingeniería de Software." **Volumen**, DOI:
- Sicilia, M. A. (2008) "Cosexions." Sistemas Gestores de Bases de Datos **Volumen**, 44 DOI:
- Slideshare (2008) "Bases de Datos." **Volumen**, 20 DOI:
- Stoner, J. A., F, Freeman , Edward y Gilbert. (1996). Administración México.
- Tedeschi, N. (2011) "Microsoft." **Volumen**, DOI:
- Téllez, R. (2005). Bases de Datos Distribuidas Cuidad de la Habana. Cuba.
- Valdés, D. P. (2007) "Maestros del Web. Bases de Datos." **Volumen**, 40 DOI:
- Zeidenstein, K. R. y. K. (2009) "IBM." Sitio Oficial de la IBM **Volumen**, 50 DOI:

BIBLIOGRAFÍA

- **Apesol. 2002.** APESOL Asociación Peruana de Software Libre. [En línea] 2002. [Citado el: 1 de 11 de 2010.] Disponible en : http://www.apesol.org.

- **Comunicaciones, Ministerio de la Informática y las.** Ministerio de la Informática y las Comunicaciones. *Ministerio de la Informática y las Comunicaciones.* [En línea] [Citado el: 22 de noviembre de 2009.] Disponible en : http://www.mic.gov.cu//hmicfunciones.aspx..

- **Craig Walls, Ryan Breidenbach.** *Spring in Action (Second Edition).*

- El Modelo de Dominio es una representación visual estática del entorno real objeto del proyecto. *Modelo de Dominio .* 2007 .

- **Encamina. 2005.** Encamina Serivios de ingeniería de software. *Encamina Serivios de ingeniería de software.* [En línea] 2005. [Citado el: 23 de 10 de 2010.] Disponible en :http://www.encamina.com/boletines/images/MKT-PF-2005-0003-01%20-%20ENCAMINA%20y%20la%20Metodolog%C3%ADa%20XP.pdf.

- **González, Carlos Sánchez.** *Seguridad no intrusiva con Acegi Security System for Spring.*

- **Graells, Dr. Pere Marques. 2008.** Grandes aportes de las TIC. [En línea] 2008. [Citado el: 22 de 10 de 2010.] Disponible en : http://www.peremarques.net.

- **Hetch, Nielsen R. 1988.** *Neurocomputing.* EE.UU : s.n., 1988.

- **Ivar Jacobson, Grady Booch, James Rumbaugh. 2000.** *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

- **Orallo, Enrique Hernández.** El Lenguaje Unificado de Modelado (UML).

- **PostgreSQL. 2007.** Comunidad Técnica Cubana de PostgreSQL. *Comunidad Técnica Cubana de PostgreSQL.* [En línea] 2007. [Citado el: 15 de 10 de 2010.] Disponible en : http://postgresql.uci.cu/.

- **Pressman, Roger S. 2005.** *Ingenieria de Software: Un enfoque práctico.* 2005.

- **Rivas, Lornel A., María Pérez, Luis E. Mendoza y Anna Grimán. 2002.** Herramientas de Desarrollo de Software. Maracay, Aragua, Venezuela : s.n., 2002.

- 2004. Tecnologia educativa. [En línea] 2004. [Citado el: 22 de octubre de 2010.] Disponible en : http://peremarques.pangea.org/tic.htm