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

Facultad 8

Análisis y diseño del módulo encargado de la comunicación, sincronización e historial de la plataforma educativa Dolphin.

Autor:
Dairelys Enriquez García

Tutores:
Ing. José Antonio Soto Pérez
Ing. Yoennis Garrido Vargas
Ing. Jorge Antonio Díaz Gutiérrez

Ciudad de La Habana, junio del 2010.
"Año 52 de la Revolución".

"…y si alguna vez nuestro trabajo nos pareciera bueno, debemos de luchar porque sea mejor, debemos de luchar porque sea perfecto, sabiendo de antemano que ninguna obra humana será lo suficientemente buena, ni lo suficientemente perfecta…"

Fidel Castro Ruz

Declaración de autoría.

Declaro que soy el único autor del trabajo “Análisis y diseño del módulo encargado de la comunicación, sincronización e historial de la plataforma educativa Dolphin” y autorizo a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____

Autor: Dairelys Enriquez García

_____

Tutor: José Antonio Soto Pérez

_____

Tutor: Yoennis Garrido Vargas

_____

Tutor: Jorge Antonio Díaz Gutiérrez

Dedicatoria

A mi mamá por todo el amor y la confianza depositada en mí, por estar en cada momento cuando la necesité.

A mi papá, por todo el amor que siento por él.

A mis hermanos jimaguas por formar parte de mi vida.

A mi novio pues con su amor, me ha hecho feliz durante estos años.

A mis abuelas y a mi abuelo por darme todo el amor del mundo.

A mis tías, tíos y a mis primos.

En fin a toda mi familia.

Agradecimientos

Quisiera agradecer a todas las personas que de una forma u otra han colaborado con la elaboración de este trabajo, y con mi formación como profesional, en especial a:

Mis padres María Elena y Pedro Fidel, por haber vivido cada momento de mi vida a mi lado, por su apoyo, firmeza, dedicación, por darme las fuerzas y el valor de llegar al final del camino y sobre todo por no perder la fe en mí.

A Yosbel, mi novio, pues sin su amor, ayuda y comprensión no lo hubiese logrado.

A su familia, que ya forma parte de la mía y en especial a mis suegros y a mi cuñada por su apoyo de forma incondicional.

A mis hermanos, que de una forma u otra siempre han estado conmigo.

A mis abuelas Neya y Caridad y a mi abuelo Yito.

A mi tía Mayda por ser mi segunda mamá.

A toda mi familia por brindarme su apoyo y comprensión.

A mis tutores Jorge, José y Yoennis por toda la ayuda y paciencia demostrada, a pesar del trabajo. Mil gracias…

Al tribunal y al oponente por toda la ayuda brindada.

A todos los profesores que me han dado su apoyo, para que hoy pudiera lograr este sueño.

A mis amigos y amigas por estar siempre conmigo.

A todos los que contribuyeron de una forma u otra a mi formación como ingeniera, porque sin la ayuda de ustedes no hubiera podido realizar este sueño.

A la Revolución y a Fidel por permitirme estudiar en una universidad como esta.

Resumen

En la actualidad la ciencia y la tecnología son elementos indispensables en el desarrollo del país. Esto ha motivado a que se inserte el uso las Tecnologías de la Información y las Comunicaciones (TICs) en las diferentes ramas de la vida, como en el proceso educativo. A partir de la creación del software educativo Español para no hispano hablantes, el cual, en su primera versión no satisface la principal necesidad del cliente: permitir a un profesor realizar el seguimiento pedagógico de un estudiante de manera centralizada; donde otra deficiencia del producto es que el entorno de trabajo presenta fallas en la conectividad, es que se propone la investigación que lleva por tema “Análisis y diseño del módulo encargado de la comunicación, sincronización e historial de la plataforma educativa Dolphin”.

Para guiar el desarrollo de la investigación se empleó la metodología ágil Extreme Programming (XP), apoyado del Lenguaje Unificado de Modelado (UML) para modelar los artefactos durante las etapas de análisis y diseño.

Con la realización del presente trabajo se pretende centralizar y actualizar la información generada por las interacciones realizadas con los contenidos de la plataforma educativa Dolphin y que el profesor pueda darle seguimiento a cada una de ellas, comunicar el servidor y el cliente, sincronizar las bases de datos y obtener los artefactos necesarios para su futura implementación.

Palabras claves

Comunicación, sincronización, historial, bases de datos, servidor, cliente.

Índice

Introducción 1
1.1 Introducción 4
1.2 Descripción general del objeto de estudio 4
1.3 Proceso de replicación de datos 5
1.3.1 Tipos de replicación 5
1.3.3 Tipos de réplicas 7
1.4 Proceso de sincronización. 8
1.4.1 Los retos de la sincronización de datos 8
1.5 Entornos de replicación y sincronización 9
1.5.1 Entorno homogéneo 9
1.5.2 Entorno heterogéneo 9
1.5.3 Entorno a utilizar 9
1.6 Proceso de comunicación 9
1.7 Soluciones existentes 10
1.7.1 Soluciones existentes a nivel internacional 10
1.7.2 Soluciones existentes en la UCI 11
1.7.3 Desventajas generales 12
1.8 Metodologías de desarrollo y lenguaje de modelado 12
1.8.1 Extreme Programming 12
1.8.2 Proceso Unificado de Desarrollo 14
1.8.3 Lenguaje de modelado y herramienta CASE para el modelado 16
1.9 Herramientas y tecnologías a utilizar 17
1.9.1 Adobe Integrated Runtime 17

1.9.2 Drupal 18

1.9.1 TortoiseSVN 1.4.5 19

1.10 Sistemas gestores de base de datos 19

1.10.1 Gestor de base de datos para la aplicación cliente 19

1.10.2 Gestor de base de datos para la aplicación servidor 19

1.11 Explicación de la selección de MySQL 21

1.12 Conclusiones 21

Capítulo 2 22

2.1 Introducción 22

2.2 Descripción del despliegue 22

2.3 Descripción de la comunicación cliente-servidor 23

2.4 Descripción del proceso de sincronización 23

2.4.1 Sincronización de los contenidos 30

2.4.2 Sincronización de los Ficheros 33

2.4.3 Sincronización de trazas 35

2.5 Conclusiones 41

Capítulo 3 43

3.1 Introducción 43

3.2 Configuración de Drupal. Instalación 43

3.2.1 Vistas necesarias 44

3.3 Flex 45

3.3.1 Diseño de clases de comunicación con Drupal 45

3.4 Diseño de la base de datos 47

3.5 Conclusiones 49

Capítulo 4 50
4.1 Introducción 50
4.2 Exploración 50
4.2.1 Historias de usuario 50
4.3 Planificación 54
4.3.1 Estimación de esfuerzos por historias de usuario 54
4.3.2 Iteraciones 54
4.3.3 Plan de entregas 55
4.4 Conclusiones 55
Capítulo 5 56
5.1 Introducción 56
5.2 Desarrollo de las iteraciones 56
5.2.1 Iteración 1 56
5.2.2 Iteración 2 58
5.2.3 Iteración 3 60
5.3 Conclusiones 62
Conclusiones 63
Recomendaciones 64
Referencias Bibliográficas 65
Bibliografía 67
Glosario de Términos 68

Introducción

En la actualidad la ciencia y la tecnología son elementos indispensables en el desarrollo del país, estas abarcan la mayor parte de las esferas de la vida del hombre. Esto ha motivado a que se inserte el uso las Tecnologías de la Información y las Comunicaciones (TIC) en las diferentes ramas de la vida, en las cuales se incluye el proceso educativo. El surgimiento de las TIC ha impuesto nuevas exigencias en el proceso de enseñanza tradicional, donde el profesor era el centro de este, siendo a su vez el principal trasmisor de información. Con la introducción de las TIC se establecen condiciones para que cada estudiante se pueda interrelacionar con los materiales docentes, según sus intereses, posibilidades y motivaciones.

Dentro de los acuerdos realizados entre Cuba con otros países, se incluyen la educación y formación profesional de estudiantes extranjeros en nuestro país. Actualmente a ellos se les imparten cursos para su formación como profesionales, pero lo hacen de forma tradicional, con extensa documentación y sin soporte de tecnologías digitales.

La Universidad de la Ciencias Informáticas (UCI) está inmersa en el proceso de informatización que se está llevando a cabo en el país. Desde su surgimiento la Facultad 8 se ha especializado en la producción de multimedia y software educativo, con el objetivo de apoyar el proceso docente educativo de una forma más amena, mediante combinación de textos, imágenes, videos, sonidos, juegos o actividades interactivas a los estudiantes; y a los profesores facilitarle la forma de controlar la evolución en la adquisición de conocimientos de cada estudiante.

Uno de los productos que ha desarrollado la facultad 8 es Español para no hispano hablantes, el cual, en su primera versión no satisface la principal necesidad del cliente: permitir a un profesor realizar el seguimiento pedagógico de un estudiante de manera centralizada, pues hasta el momento, esas trazas (o huellas que deja el estudiante, en su interacción con el producto antes mencionado) se guardaban en cada una de las computadoras donde trabajaban los estudiantes, trayendo como consecuencia que los profesores debían revisar cada computadora para actualizarse respecto al estado de los mismos, algo que provoca pérdida de tiempo, además de la descentralización y desactualización de la información de los estudiantes al sentarse en puestos de trabajo diferentes. Otra deficiencia del producto es que el entorno de trabajo presenta como problemática fallas en la conectividad, lo que puede provocar pérdida de

información, dificultando y en ocasiones imposibilitando el seguimiento de los estudiantes por parte de los profesores. La situación actual define como **problema de la investigación** ¿Cómo centralizar y actualizar la información generada por la interacción del estudiante con los contenidos y que el profesor pueda darle seguimiento?

Se define como **objeto de estudio** proceso de desarrollo de sistemas que permitan la comunicación y sincronización entre bases de datos.

Siendo el **objetivo general** de la presente investigación: realizar el análisis y diseño del módulo encargado de la comunicación, sincronización e historial de forma dinámica de las interacciones realizadas con los contenidos de la plataforma educativa Dolphin.

Como **campo de acción** el diseño de un módulo que permita la comunicación y sincronización entre servidores y clientes para la plataforma educativa Dolphin.

Para darle cumplimiento al presente trabajo se cuenta con los siguientes **objetivos específicos**:

- Definir un estándar para la comunicación entre servidores y clientes.
- Definir un estándar para la sincronización entre servidores y clientes.
- Realizar los artefactos necesarios del análisis y diseño.

Como **idea a defender** se tiene que: si se realiza el análisis y diseño del módulo encargado de la comunicación, sincronización e historial de forma dinámica de las interacciones realizadas por los estudiantes con los contenidos de la plataforma educativa Dolphin, entonces se centralizará y actualizará la información generada por dichas interacciones y el profesor podrá darle seguimiento.

Para darle cumplimiento a los objetivos se realizarán las siguientes **tareas:**

- Realizar un estudio y selección de las metodologías y herramientas existentes para realizar el proceso de sincronización.
- Realizar un estudio y selección de los entornos de comunicación que existen entre clientes y servidores.

- Estudiar y definir la metodología de desarrollo de software y el lenguaje de modelado a utilizar.
- Realizar el análisis del módulo.
- Realizar el diseño del módulo.
- Realizar el documento de tesis.

El presente trabajo debe tener como **posibles resultados:**

- Comunicar el servidor y el cliente.
- Sincronizar las bases de datos.
- Actualizar la información por parte del profesor.
- Centralizar los datos.
- Obtener los artefactos necesarios del análisis del módulo.
- Obtener los artefactos necesarios del diseño del módulo.

Capítulo 1

Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realizará la fundamentación teórica de la investigación, donde se expondrá el estado del arte de las soluciones existentes y se analizará las tendencias tecnológicas y los aspectos a tener en cuenta en la selección de las herramientas, tecnologías, metodologías y lenguajes de modelado.

1.2 Descripción general del objeto de estudio

El creciente desarrollo de las tecnologías ha generado en el mundo actual una tendencia a los procesos de sincronización, los cuales pueden estar dados por la sincronización de contenidos entre dispositivos y los de datos entre aplicaciones y servicios web.

Este proceso se enmarca generalmente en el intercambio de los datos que debe existir entre un cliente y un servidor, permitiendo que todos los cambios realizados independientemente de que sea en uno o en el otro, sean reflejados de igual forma en ambos. El objetivo principal de este, es mantener una coherencia entre los datos que están almacenados en un lugar y en otro.

Sin lugar a dudas una de las grandes ventajas de un software online, es poderlo utilizar desde cualquier parte en la que dispongamos de una conexión, lo que se convierte en un inconveniente grave si no tenemos esa posibilidad.

Actualmente muchos de los problemas que se presentan en los programas informáticos online son los problemas de conexión, pues, cuando estos suceden, le imposibilitan al usuario continuar interactuando con los contenidos del servidor. De ahí la importancia de trabajar local y después sincronizar las copias locales con las que están en la red, por lo que se debe disponer de una base de datos local en cada cliente, donde esté una copia de los contenidos que se encuentran en el servidor, para en el caso de que no exista conectividad, los usuarios puedan seguir interactuando localmente con los contenidos. Además, se deben almacenar las trazas o las huellas

de la interacción de cada estudiante con la aplicación, para posteriormente realizar el proceso de replicación y sincronización una vez que exista conexión.

Para resolver los problemas presentados en la primera versión de Español para no hispano hablantes, surge la plataforma educativa Dolphin, que persigue como objetivo principal la creación de una plataforma de aprendizaje en línea, así como un módulo que permita la comunicación, sincronización e historial, aspectos de gran importancia para una aplicación online, donde el entorno analizado presenta fallas de conectividad.

Seguidamente se hace un análisis de los aspectos antes mencionados, para la creación del módulo encargado de ellos en la plataforma educativa Dolphin.

1.3 Proceso de replicación de datos

La replicación de datos en la actualidad es de gran importancia para todo software que posea dos bases de datos y quiera mantener los mismos datos en ambas. Es el proceso de guardar una copia de los datos en otra Base de datos.

En este intercambio de datos se trata de capturar los cambios realizados en una base de datos determinada y trasmitir estas modificaciones a las instancias de esta base de datos. (Corporation, 2010)

1.3.1 Tipos de replicación

El proceso de replicación tiene varias formas de realizarse y estas pueden ser instantánea, transaccional y de mezcla.

1.3.1.1 Replicación instantánea

En las réplicas instantáneas los datos son distribuidos exactamente tal y como aparecen en un momento determinado.

“El uso independiente de la réplica de instantáneas es más apropiado cuando se cumple una o más de las siguientes condiciones:

- Los datos no cambian con frecuencia.
- Es aceptable disponer de copias de datos desfasados respecto al publicador durante un período de tiempo.

- Se duplican pequeñas cantidades de datos.
- Se producen muchos cambios durante un breve período de tiempo."(Microsoft Corporation, 2008)

1.3.2.2 Replicación transaccional

Este tipo de replicación consiste en distribuir inicialmente una instantánea de los datos a los suscriptores y cuando se produce algún cambio en el publicador estos se propaguen a los suscriptores.

"La réplica transaccional se utiliza por lo general en entornos de servidor a servidor, y es apropiada en los siguientes casos:

- Se desea que se propaguen cambios incrementales a los suscriptores en el momento en que ocurren.
- La aplicación requiere una latencia baja entre el momento en que se realizan los cambios en el publicador y el momento en que llegan los cambios al suscriptor.
- La aplicación necesita acceso a los estados intermedios de los datos. Por ejemplo, si una fila cambia cinco veces, la réplica transaccional permite que una aplicación responda a cada cambio y no sólo al cambio de datos neto en la fila.
- El publicador tiene un volumen elevado de actividad de inserción, actualización y eliminación." (Microsoft Corporation, 2008)

1.3.2.3 Replicación de mezcla

Este tipo de replicación consiste en distribuir inicialmente una instantánea de los datos a los suscriptores. Cuando el suscriptor y el publicador están conectados a la red, son sincronizados todos los cambios en los datos ocurridos entre ellos desde la última sincronización.

"La réplica de mezcla se suele utilizar en entornos de servidor a cliente. La réplica de mezcla es adecuada en las siguientes situaciones:

- Varios suscriptores actualizan los mismos datos en diferentes ocasiones y propagan los cambios al publicador y a otros suscriptores.

- Los suscriptores necesitan recibir datos, realizar cambios sin conexión y sincronizar más adelante los cambios con el publicador y otros suscriptores.
- Cada suscriptor requiere una partición de datos diferente.
- Se pueden producir conflictos. Cuando esto ocurre, debe poder detectarlos y resolverlos.
- La aplicación requiere el cambio de datos neto en lugar de acceso a los estados intermedios de los datos. Por ejemplo, si una fila cambia cinco veces en un suscriptor antes de que éste se sincronice con un publicador, la fila cambiará sólo una vez en el publicador, con el quinto valor, para reflejar el cambio de datos neto."(Corporation, 2008)

1.3.2.4 Replicación a utilizar

Después del análisis de los tipos de replicación se escoge para utilizar en el proyecto la replicación de mezcla, pues posee las ventajas de la replicación instantánea y transaccional, además le permite al usuario trabajar online/offline, ya que si se presenta el problema de no existir conexión, se le debe permitir al estudiante poder trabajar en la plataforma, a su vez se deben ir guardando las trazas que se van generando a partir de la interacción del estudiante con la plataforma y más adelante realizar la sincronización de todos los cambios ocurridos desde la última sincronización.

1.3.3 Tipos de réplicas

Las réplicas de datos se dividen en dos categorías, ejemplos de estas tenemos la réplica de datos en un servidor para un entorno de servidor (Multi-maestro) y la réplica de datos entre un servidor y los clientes (Maestro-Esclavo).

1.3.3.1 Réplicas Multi-maestro (Multi-Master)

La réplica Multi-maestro es también conocida como *peer to peer*. En el ambiente de réplicas Multi-Maestro cada sitio es maestro y cada uno se comunica con otros sitios maestros. Cada uno de los sitios copia los cambios directamente de los demás sitios.

Este tipo de réplica puede ser usada para mantener sitios recuperables ante posibles desastres o caídas, así como para proveer sistemas con alta disponibilidad y para balancear la carga de consultas a través de las distintas ubicaciones. (Oracle, 2001)

1.3.3.2 Réplicas Maestro-Esclavo (Master-Slave)

En el caso de las réplicas Maestro-Esclavo, los datos se pueden modificar solo en el servidor maestro, este es quien distribuye los cambios a todos los servidores esclavos para que se actualicen. Estos últimos solo tienen permiso de lectura, mientras que el servidor maestro tiene de escritura. (Ossa, 2009)

1.3.3.3 Tipo de réplica a utilizar

Después del análisis de los tipos de réplica, se escoge para utilizar en el proyecto la réplica Maestro-Esclavo, pues el servidor maestro es el que debe distribuir todos los cambios realizados en la base de datos central hacia las bases de datos locales. Una de las especificaciones a tener en cuenta en la realización del proyecto es que el servidor esclavo puede enviarle al servidor central cierta información, la cual va estar relacionada con las trazas.

1.4 Proceso de sincronización.

Se puede entender como el proceso continuo que debe existir en los datos que están ubicados en el servidor maestro, es decir, los datos que son utilizados en las diferentes transacciones del negocio se actualicen de forma constante.

La sincronización de los datos puede ser monodireccional o bidireccional. En el caso del proyecto la que se va a utilizar es la bidireccional, ya que esta se va a realizar en ambas direcciones, pues es necesario llevar contenidos y ficheros del servidor al cliente y trazas del cliente al servidor.

1.4.1 Los retos de la sincronización de datos

“Existen numerosos desafíos para implementar unos procesos de sincronización de datos eficaces y fiables.

- La sincronización de datos en **tiempo real**. Así como reducir el tiempo de procesamiento.

- Los entornos implicados son mayormente **heterogéneos**, presentándose como una gran dificultad que las estructuras de datos varían ampliamente en todos los sistemas que necesitan mantener sincronizados.

- Cuando se producen **conflictos de datos** se deben gestionar y resolver teniendo en cuenta la prioridad de actualización de los registros."(Talend, 2008)

1.5 Entornos de replicación y sincronización

Tanto la replicación como la sincronización de los datos se pueden ver en diversos entornos, dependiendo del sistema operativo y los gestores de bases de datos que se utilicen en cada uno de los servidores fuentes y destinos involucrados. Estos entornos pueden ser homogéneos o heterogéneos. (González, 2000)

1.5.1 Entorno homogéneo

La réplica y la sincronización de los datos en un entorno homogéneo se realizan entre servidores de un mismo gestor de base de datos y un mismo sistema operativo o entre servidores de un mismo gestor de base de datos, pero donde los sistemas operativos son diferentes.

1.5.2 Entorno heterogéneo

La réplica y la sincronización de los datos en un entorno heterogéneo se realizan entre servidores de diferentes gestores de base de datos y un mismo sistema operativo o entre servidores de diferentes gestores de base de datos, pero donde los sistemas operativos son diferentes.

1.5.3 Entorno a utilizar

Después del análisis de los entornos se escoge a utilizar en el proyecto el entorno heterogéneo, pues se van a usar gestores de bases de datos diferentes, teniendo en cuenta que como gestor de base de datos local se va a utilizar SQLite y como gestor de base de datos central MySQL.

1.6 Proceso de comunicación

La comunicación entre el cliente y el servidor está basada fundamentalmente por protocolos. El cliente no es más que una aplicación de escritorio basada en entornos

RIA (Rich Internet Applications). Generalmente en la arquitectura para la comunicación está presente: el cliente, el cual maneja la interacción entre el usuario y la *interfaz del usuario*. El usuario invoca comandos, actualiza vistas y carga datos. Desde aquí se mantiene el estado de la aplicación, se manejan todas las peticiones de datos hacia el servidor y se controla como se presentan los datos. Los servicios, es el otro de los elementos con que cuenta y pues estos no son más que los encargados de procesar todas las peticiones de la aplicación cliente y delegar las acciones en el servidor. A partir de la comunicación que exista entre ambos se podrán guardar datos en la base de datos, actualizar los archivos del sistema, retornar datos al servidor. Determina y le da formato a los datos que son retornados al cliente. (Grajeda, 2009)

1.7 Soluciones existentes

Ante el creciente avance de la tecnología, los desarrolladores de Software de sincronización tienen nuevos retos que enfrentar. De ahí que dentro de estos se encuentre el desarrollo de aplicaciones que realicen este proceso, así como las réplicas entre base de datos. A continuación se hace mención de algunos sistemas informáticos que realizan uno o ambos de los procesos antes mencionados, en dependencia del caso.

1.7.1 Soluciones existentes a nivel internacional

Cross-DataBase Studio

Descripción

Compara, sincroniza, migra, replica estructura y datos con el mismo producto y entre diferentes motores de bases de datos.

Características

- “Cross-DataBase Studio es un entorno de trabajo sencillo, intuitivo e integrado para comparar, sincronizar y replicar bases de datos entre diferentes motores de datos.
- Utilizando Cross-DataBase Studio puedes comparar, sincronizar y replicar estructura y datos entre cualquier tipo de Base de datos. Los productos de DBBalance te quitan de la difícil tarea de definir reglas de comparación y réplica. De esta manera estos procesos se reducen y simplifican radicalmente.

- Cross-DataBase Studio incluye informes de resultado de la comparación, sincronización y réplica en formato XML y HTML. Estos informes se pueden integrar fácilmente en cualquier sistema de informes o de control de procesos.

- No hay necesidad de instalar y configurar por separado ningún driver ODBC. Los productos de DBBalance incluyen un conjunto de drivers ODBC de las bases de datos más importantes del mercado.

- Las soluciones de DBBalance incluyen una amplia gama de configuraciones predefinidas para la comparación y réplica. No hay necesidad de configurar ninguna relación entre diferentes tipos de datos y objetos para la comparación y réplica entre diferentes motores de bases de datos. Simplemente selecciona los objetos que quieres comparar, sincronizar o replicar, y ya está. Siempre puedes controlar la configuración y definir tus propias reglas, si tu proceso requiere cualquier definición especial. También puedes planificar el tiempo de inicio de ejecución del proceso."(DBBalance, 2010)

La compañía DBBalance ofrece tres productos, donde el más completo es al que se hace mención anteriormente. Esta herramienta tiene como inconvenientes que nuestra universidad no puede utilizarla debido a que para adquirirla hay que pagar precios muy elevados.

DBSync for SQLite & MySQL

Descripción

Es una herramienta para la conversión de bases de datos MySQL y SQLite, que ofrece una sincronización bidireccional.

Dentro de sus principales características están: permite una alta velocidad de conversión de datos, verifica los posibles errores de conversión, realiza la sincronización sin pérdidas de datos y asegura la integridad de la base de datos.

Esta herramienta, al que la anterior, tiene el mismo inconveniente de los precios, por lo que la universidad no puede utilizarla.

1.7.2 Soluciones existentes en la UCI

CONTACC

Descripción

"CONTACC, el sistema de control de acceso a comedores en la UCI. Este sistema tiene como objetivo fundamental gestionar los accesos a comedores, ello consiste en: registrar accesos, comprobando que dicha persona existe, tiene autorización para pasar por esa puerta y no ha accedido con anterioridad, sincronizar accesos o actualizar cada cierto tiempo la información registrada en las bases de datos locales con las bases de datos centrales a través del Servicio Web; y brindar un servicio de reportes donde se muestre al usuario la cantidad de personas (por tipo) que han tenido acceso hasta el momento."(Adisley Flores Coronado, 2007)

Este sistema tiene como inconveniente, que está desarrollado bajo la plataforma .NET, la cual no es la tecnología que se utiliza en el proyecto.

1.7.3 Desventajas generales

Las 3 herramientas analizadas anteriormente son sistemas especializados en la sincronización de contenidos de bases de datos MySQL y SQLite, pero están orientadas a usuarios de altos conocimientos de bases de datos y no son integrables bajo ninguna circunstancia, con un sistema de aprendizaje debido a que sería necesario ejecutarlas de forma paralela para sincronizar las bases de datos.

A ello se suman las desventajas particulares de cada una de estas herramientas, en las que prevalece el costo de adquisición y la incompatibilidad de la plataforma sobre la cual es necesario construir el sistema de sincronización.

Finalmente, la solución seleccionada consiste en desarrollar un sistema de sincronización heterogéneo que utilice el sistema de réplicas Maestro-Esclavo.

1.8 Metodologías de desarrollo y lenguaje de modelado

Las metodologías de desarrollo de software proporcionan procedimientos, técnicas, herramientas y soporte documental para la realización de proyectos. Existen actualmente numerosas propuestas de metodologías, las cuales pueden ser tradicionales (en la cuales se resalta la preocupación de documentar exhaustivamente el proyecto) y ágiles o ligeras (están destinadas enfrentar la producción en entornos dinámicos e inconstantes).

1.8.1 Extreme Programming

Programación Extrema o Extreme Programming (XP) es una metodología ágil de desarrollo de software que ha alcanzado gran éxito entre los desarrolladores en la actualidad. Surge por un proyecto desarrollado por Kent Beck. Está basada en la simplicidad, en la comunicación y retroalimentación constante con el cliente.

Las características fundamentales de la metodología son:

- "Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera el código es revisado y discutido mientras se escribe es más importante que la posible pérdida de productividad inmediata.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que en más sencillo hacer algo simple y tener un poco de trabajo

extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo."(Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera, 2008)

Ventajas:

- "Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente.
- Permite tener realimentación de los usuarios muy útil.
- La presión está a lo largo de todo el proyecto y no en una entrega final."(Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera, 2008)

Desventajas:

- "Delimitar el alcance del proyecto con nuestro cliente."(Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera, 2008)

Después del estudio realizado y teniendo en cuenta las ventajas y desventajas que presenta esta metodología, se puede decir que esta se ajusta a las necesidades y condiciones de un ambiente de trabajo compuesto por equipos pequeños, donde se puede enfrentar a los requisitos cambiantes y se puede realizar el proyecto en corto período de tiempo. Estableciéndose una retroalimentación continua entre el cliente y el equipo de los desarrolladores, una comunicación fluida entre los participantes y una simplicidad en las soluciones implementadas.

1.8.2 Proceso Unificado de Desarrollo

Proceso Unificado de Desarrollo (RUP) es una metodología tradicional, la cual constituye una de las metodologías más utilizadas para desarrollar sistemas orientados a objetos. Esta cuenta con tres características fundamentales, las cuales son: dirigido por Casos de Usos, iterativo e incremental y centrado en la arquitectura.

RUP divide el proceso en cuatro fases: Inicio, Elaboración, Construcción y Transición. Además, define nueve flujos de trabajos; seis de Ingeniería: Modelamiento del

Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue; y tres de apoyo: Gestión de la Configuración, Gestión de Proyecto y Ambiente. Utiliza UML como lenguaje de modelado. (J. Rumbaugh, 2000)

Ventajas:

- "Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases."(Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera, 2008)

Desventajas:

- "La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- Estamos poniendo a nuestro cliente en una situación que puede ser muy incómoda para él.
- Nuestro cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él."(Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera, 2008)

Después del estudio realizado y a pesar de las ventajas que nos proporciona esta metodología, uno de los inconvenientes para no utilizarla es que principalmente se usa en proyectos de gran magnitud, donde los requisitos no sean muy cambiantes y donde el proceso de desarrollo es a largo plazo, provocando ante un cambio que se vuelva a generar toda la documentación, alargando el tiempo de desarrollo, pues no es flexible para desarrollar proyectos pequeños que puedan estar sujetos a cambios.

1.8.3 Selección de la metodología de desarrollo

En ambas metodologías se puede hacer un reajuste amplio de roles y artefactos para adaptar el proyecto a equipos de trabajos compuestos por solo 1 persona. RUP es conocido por la robustez de su proceso de desarrollo a largo plazo y XP por la rapidez

a corto plazo de las entregas. Después del estudio realizado sobre las posibles metodologías de desarrollo a utilizar, se seleccionó XP de acuerdo con las políticas establecidas por el proyecto plataforma educativa Dolphin, a partir de la experiencia de algunos de los miembros del equipo de desarrollo en su utilización, además de los elementos que aparecen a continuación:

- El período de desarrollo es corto: El desarrollo de la solución se limita a solamente 4 meses de trabajo continuo.
- El cliente forma parte del equipo de desarrollo.
- Las dimensiones del proyecto son pequeñas.

- Uno de los objetivos específicos es publicar versiones de pruebas para obtener retroalimentación de diferentes probadores de la universidad y así optimizar el producto. Para ello es necesario tener un período de entregas corto, respaldado por iteraciones cortas y un proceso de desarrollo bastante ágil.

1.8.3 Lenguaje de modelado y herramienta CASE para el modelado

1.8.3.1 Lenguaje Unificado de Modelado

"El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales."(J. Rumbaugh, 1998)

Este lenguaje de modelado a través de diagramas puede mostrar de una forma detallada la solución al problema planteado.

1.8.3.2 Visual Paradigm for UML

Como herramienta a utilizar en el modelado de la solución del sistema se va a usar Visual Paradigm. Pues dentro de las características principales que presenta están:

- Generación de código ActionScript 3.0 desde los diagramas, es una de las pocas herramientas que genera esta versión del lenguaje ActionScript.
- Interoperabilidad entre diagramas: facilita el intercambio de diagramas UML y modelos con otras herramientas.
- Tiene soporte para la versión 2.1.
- Genera documentación.
- Es multiplataforma.
- La universidad posee su licencia.

1.9 Herramientas y tecnologías a utilizar

1.9.1 Adobe Integrated Runtime

Adobe AIR (en adelante AIR) es capaz de portar Flash Player al escritorio y a su vez a las aplicaciones ejecutándose sobre este. Otras de sus características son:

- "La capacidad de crear bases de datos locales para la gestión del software en el ordenador del cliente sin necesidad alguna de comunicarse con la red.
- Sus funcionalidades más comunes son similares a las de cualquier otra aplicación instalada previamente y que utilice otra tecnología.
- Permite un manejo total del sistema de archivos, el control de los contenidos del portapapeles del sistema y la aplicación puede ser completamente actualizada de forma automática utilizando una conexión a internet."(Incorporated, 2009)

La selección del framework de AIR para el soporte de Flash Player, permite lograr, sin problema alguno, la gestión externa de un entorno de desarrollo integrado como aplicación de escritorio. Su principal desventaja se encontraba en el hecho de no ser multiplataforma y solamente contar con una versión de prueba para Linux. Ya hace más o menos un año que Adobe hizo pública la versión estable de AIR para Linux, eliminando esta limitante y haciéndolo completamente multiplataforma.

1.9.2 Adobe Flex Open Source SDK

Flex es una tecnología reciente y de gran aceptación en el desarrollo de RIAs y aplicaciones para Flash Player. En poco menos de 3 años, ha inundado la red con soluciones que oscilan entre sencillas aplicaciones y grandes sistemas empresariales, gracias a la calidad de su arquitectura en general. Su gran impacto vino impulsado por la aparición de la versión 2.0. El framework actual está compuesto por una librería de componentes de interfaz de usuario que ofrecen un alto grado de interactividad y la capacidad de comunicarse con los sistemas de datos más comunes que existen. Al igual que Flash Player, el SDK libre de Flex es desarrollado por Adobe y mantenido por diversas comunidades a lo largo del mundo que crean continuamente nuevos proyectos y mejoran los existentes. La principal línea de desarrollo de estas es la elaboración de nuevos componentes para las comunicaciones y de frameworks arquitectónicos para el desarrollo de aplicaciones on-line.

“Su principal debilidad residía en que, al ejecutarse sobre Flash Player, se encontraba obstaculizado por las limitantes de acceso que presenta el reproductor, la cual fue eliminada con la aparición de AIR a inicios de 2008 haciendo posible que se pueda seleccionar a Flex como uno de los frameworks a utilizar.”(Inc, 2009)

Flex es clave para la creación de la presente solución debido a que aporta la librería de componentes de interfaz de usuario necesaria para la gestión interna del software a desarrollar y, a su vez, el compilador y el depurador necesario para la programación del proyecto.

1.9.2 Drupal

Drupal es un programa libre escrito en PHP, el cual es mantenido por una activa comunidad de usuarios. Publicado bajo la licencia GNU/GPL. Es el sistema de gestión de información más popular que existe en la actualidad debido a la flexibilidad que tiene para adaptarse a diversos tipos de contenidos, los cuales se les llama *nodos*. Estos nodos son almacenados en 3 tablas fundamentales y permiten una amplia variedad de modificaciones, por lo que en la construcción del proyecto se decidió utilizarlo como interfaz del sistema central de almacenamiento de datos.

En el proyecto su utilización es debido a las facilidades que brinda Drupal, pues es un sistema para la gestión de contenidos modular multipropósito y muy configurable. Muy conocido por la calidad de su código y por la seguridad que brinda, es estable y de actualización continua, configuración sencilla, instalación ágil, posee una importante cantidad de módulos, una excepcional documentación, una comunidad activa y muy

amigable. Es a fin a uno de los lenguajes a utilizar en el proyecto, el cual es PHP. Además, se integra perfectamente al gestor de base de datos de MySQL.

1.9.1 TortoiseSVN 1.4.5

Es un cliente de subversión, se utilizará para el sistema de control de versiones del proyecto. Está desarrollado bajo la Licencia Pública General GNU/GPL y se integra perfectamente en el Shell de Windows. Esta herramienta es de gran utilidad para que no se pierda información.

1.10 Sistemas gestores de base de datos

1.10.1 Gestor de base de datos para la aplicación cliente

1.10.1.1 SQLite

"SQLite es una pequeña librería programada en lenguaje C que implementa un completo motor de base de datos multiplataforma que no precisa configuración. Se distribuye bajo licencia de dominio público. Es muy rápido y la ventaja fundamental es que permite utilizar un amplio subconjunto del lenguaje estándar SQL. SQLite destaca también por su versatilidad. El motor de PHP 5 incluye soporte interno para SQLite."(Educativas)

Se utilizará SQLite como gestor de base de datos local en una computadora cliente por su rapidez y simplicidad, escaso tamaño y por su completa portabilidad. Además, porque Adobe AIR permite crear localmente bases de datos SQLite y ser manejadas utilizando su lenguaje de programación: ActionScript 3.0.

1.10.2 Gestor de base de datos para la aplicación servidor

1.10.2.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional. Cumple completamente con las características ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

A continuación se exponen algunas de las características de este gestor de bases de datos:

1. "Implementación del estándar SQL92/SQL99.

2. Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP...), cadenas de bits, etc. También permite la creación de tipos propios.

3. Incorpora una estructura de datos array.

4. Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.

5. Permite la declaración de funciones propias, así como la definición de disparadores.

6. Soporta el uso de índices, reglas y vistas.

7. Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto relacionales.

8. Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos."(Pecos, 2010)

1.10.2.3 MySQL

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. (Pecos, 2010)

Dentro de sus principales características están las siguientes:

1. "Aprovecha la potencia de sistemas multiprocesadores, gracias a su implementación multihilo.

2. Soporta gran cantidad de tipos de datos para las columnas.

3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc.).

4. Gran portabilidad entre sistemas.

5. Soporta hasta 32 índices por tabla.

6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos." (Pecos, 2010)

1.11 Explicación de la selección de MySQL

Para la construcción del sistema central de datos se seleccionó MySQL debido a la facilidad de instalación utilizando un paquete de aplicaciones integradas, por los conocimientos previos del equipo de trabajo en manejo de dicho gestor de bases de datos, además MySQL es un gestor de base de datos multiplataforma, de fuente abierta lo que significa que cualquier persona puede usarlo y modificarlo para satisfacer sus necesidades, puede ser utilizado gratuitamente. Por su sencillez y sus características es usado por muchas personas, consume muy pocos recursos, se usa tanto en aplicaciones sencillas como complejas. Es utilizado en aplicaciones Web como Drupal, del cual se hace alusión su selección anteriormente por las facilidades que le proporciona a los desarrolladores.

1.12 Conclusiones

Una vez concluido el presente capítulo se pudo observar que ninguna de las soluciones existentes se ajusta a las necesidades, ya que la mayoría están suscritas a algún tipo de restricciones, en la mayoría de los casos es el precio para adquirirlas. Después de haber analizado las metodologías existentes se decidió que por sus facilidades, documentación, y flexibilidad se utilizaría XP. Como herramienta CASE se seleccionó Visual Paradigm 6.04 apoyado en UML para la generación de los artefactos. Para construir la propuesta de solución, se seleccionaron las tecnologías: Adobe AIR y Adobe Flex Open Source SDK como framework, Drupal 6 como CMS y TortoiseSVN, así como los gestores de base de datos para la aplicación cliente SQLite y para la aplicación servidor MySQL Una vez analizados y definidos estos aspectos, se iniciaría la descripción de la solución propuesta en el próximo capítulo.

Capítulo 2

Presentación de la solución propuesta

2.1 Introducción

El presente capítulo tiene como objetivos describir las principales características de la solución propuesta. Teniendo en cuenta la comunicación entre el cliente y el servidor central, detallando como se realizará el proceso de sincronización, particularizando como se lleva a cabo el de contenidos, ficheros y trazas, además se definirán las tablas que se usarán en la base de datos local SQLite, a partir del análisis realizado del Diagrama Entidad Relación (DER) de la base de datos del CMS Drupal.

2.2 Descripción del despliegue

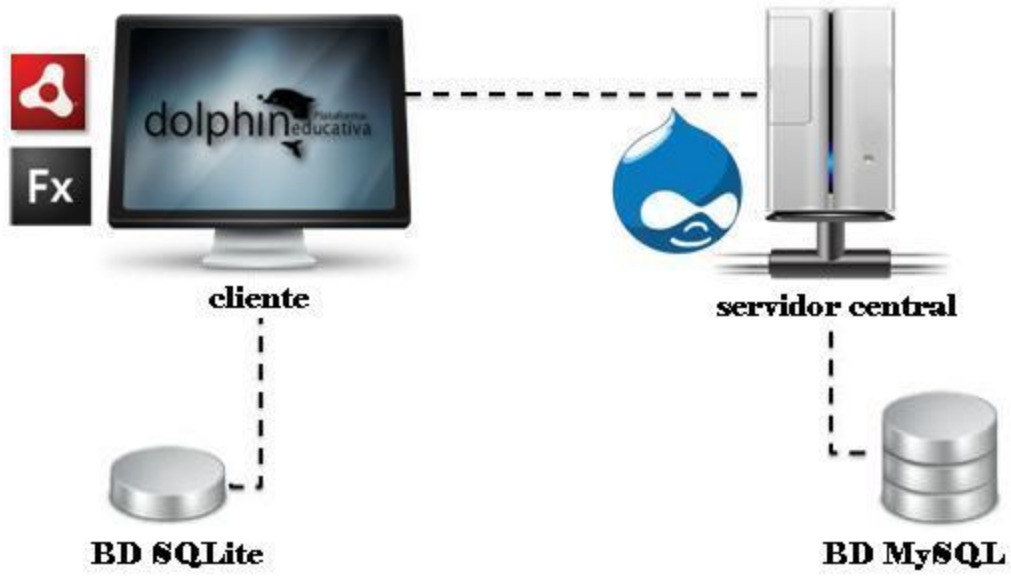

Fig. 2.1 Solución propuesta.

La solución propuesta cuenta con un cliente, que no es más que una aplicación de escritorio realizada con AIR y Flex, el cual se comunica con la base de datos local en SQLite, este estará presente en las computadoras de todos los laboratorios, donde los estudiantes interactuarán con el curso que se impartirá. Además, contará con un servidor central donde estará instalado el CMS Drupal que permitirá la gestión de los

contenidos, el mismo se conectará con la base de datos MySQL, en la cual se encuentran todos los contenidos del curso a impartir. También entre el cliente y el servidor central existirá una comunicación que servirá para el proceso de sincronización de los contenidos y ficheros, así como el de las trazas de la interacción del estudiante con el cliente.

2.3 Descripción de la comunicación cliente-servidor

Fig. 2.2 Comunicación cliente-servidor.

AMF(Action Message Format) es un formato binario basando en SOAP (Simple Object Access Protocol). AMF fue creado para intercambiar datos entre aplicaciones Adobe Flash y bases de datos, básicamente usando RPC (Remote Procedure Call).

AMFPHP es un protocolo que nos permite comunicar datos de aplicaciones y funciones entre un cliente que puede estar en JavaScript, Flash, Flex, entre otros y un servidor PHP. En este caso AMFPHP realiza la comunicación de procesos remotos entre Flash (por extensión de aplicaciones RIA en Flex y AIR) y PHP. AMFPHP es una implementación de código abierto en PHP para el protocolo AMF de Adobe.

2.4 Descripción del proceso de sincronización

En el proceso de sincronización, el flujo de información ocurre en dos sentidos. Uno, que es entre el servidor central y el cliente, en el que fluyen los contenidos y ficheros, y se puede ver en la fig. 2.3 que se muestra a continuación.

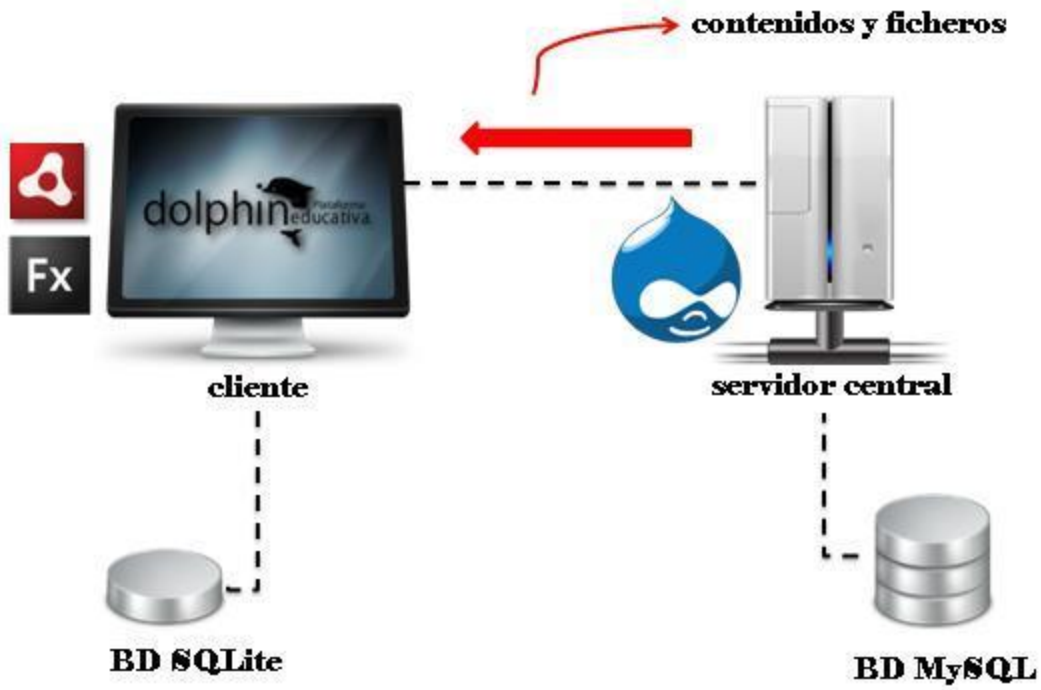

Fig. 2.3 Sentido de la sincronización de contenidos y ficheros.

Y el otro es entre el cliente y el servidor central, que es el sentido en que fluyen las trazas, como se muestra en la fig. 2.4 siguiente:

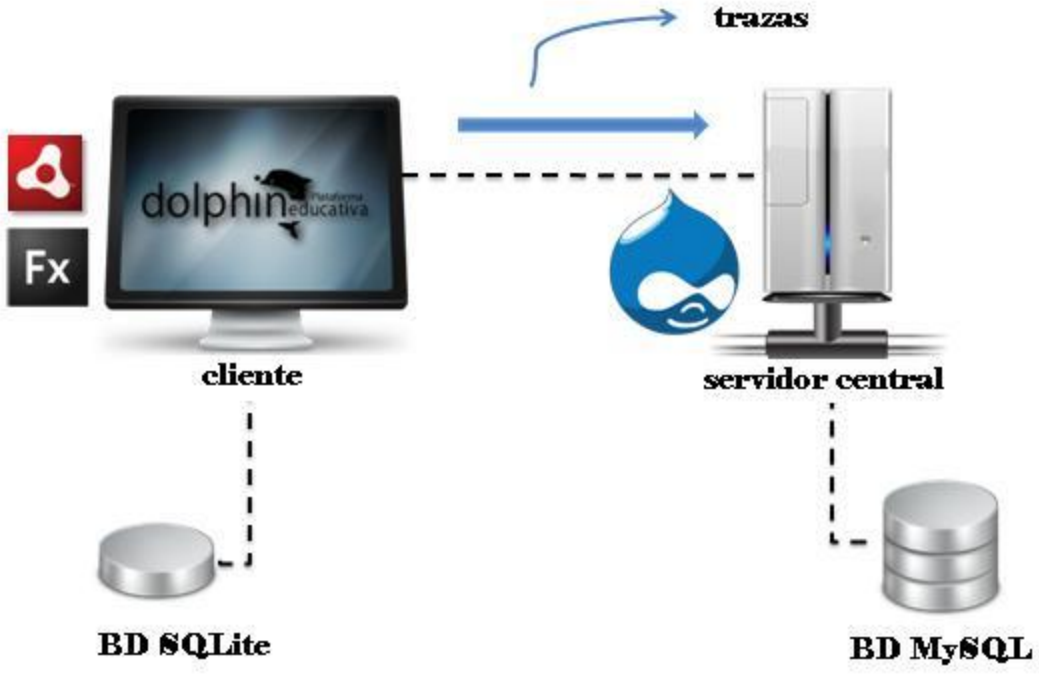

Fig. 2.4 Sentido de la sincronización de trazas.

Una vez que el usuario por primera vez interactúa con el cliente, este se comunicará a través del protocolo AMFPHP con el servidor central, en el cual está instalado el CMS Drupal 6, y este último interactuará con la base de datos de MySQL, para inicialmente descargar todos los contenidos para la base de datos local soportada por SQLite, el principal objetivo de esto es mejorar el rendimiento y la rapidez de la interacción del usuario con el cliente. En el caso en que no exista conexión entre la aplicación de escritorio y el servidor central, como se muestra en la fig. 2.5.

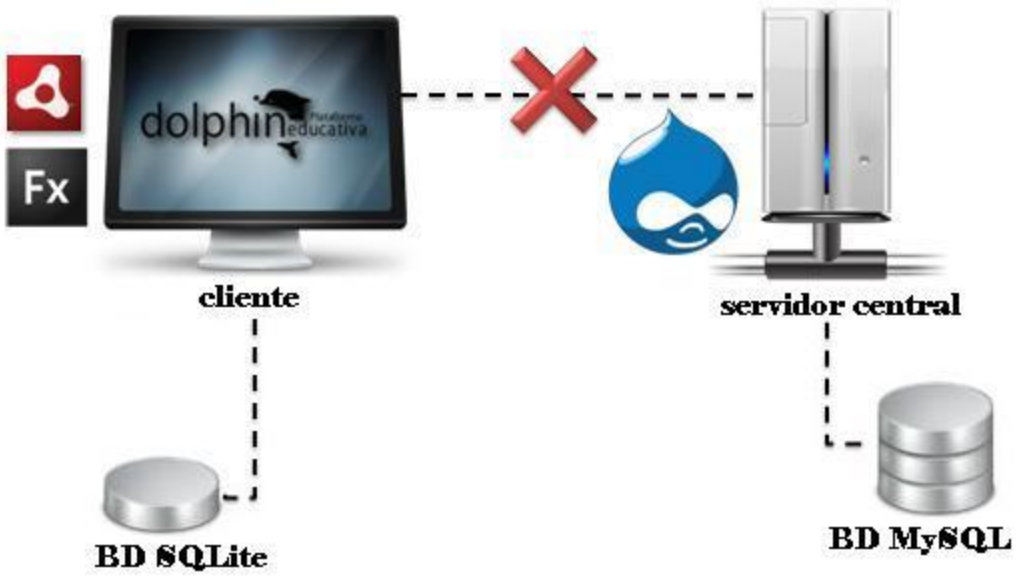

Fig. 2.5. Estado del sistema cuando no existe conexión entre el cliente y el servidor.

En situaciones de conectividad nula, el usuario puede seguir trabajando con la copia local de los contenidos que tiene en la base de datos SQLite, y una vez que se recupere la conexión, se envían al servidor todos los cambios ocurridos desde la última sincronización. Para una mejor ilustración del proceso, en caso de que no exista conexión se muestra una descripción detallada a continuación:

Fig. 2.6 Petición del cliente a la base de datos SQLite.

Paso 1: Inicialmente el usuario interactúa con el cliente y este a su vez le hace peticiones de los contenidos a la base de datos SQLite.

Fig. 2.7 Respuesta de SQLite al cliente.

Paso 2: La base de datos SQLite provee los datos de respuesta de la petición hecha por el cliente.

Fig. 2.8 Situación de conectividad.

Paso 3: En situaciones de conectividad entre el cliente y el servidor, y la aplicación lo haya verificado.

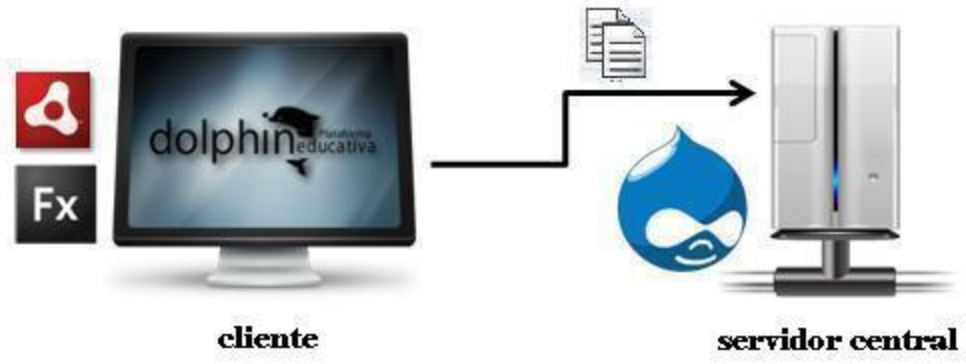

Fig. 2.9 Envío de datos del cliente al servidor.

Paso 4: Se envían al servidor central los datos (en este caso las trazas del historial), de los cambios ocurridos desde la última sincronización.

Fig. 2.10 El servidor almacena los datos en la base de datos MySQL.

Paso 5: Se almacenan en la base de datos central MySQL los datos enviados por el servidor.

Para un mejor entendimiento del proceso en caso de que exista conexión se muestra una descripción detallada a continuación:

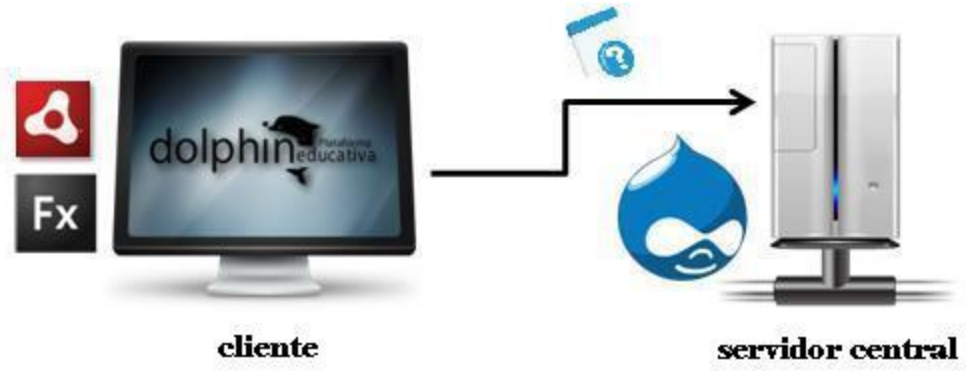

Fig. 2.11 Petición del cliente al servidor.

Paso 1: Inicialmente el usuario interactúa con el cliente y este a su vez se comunica con el servidor central, para hacerles las peticiones de los contenidos a la base de datos.

Fig. 2.12 Petición del servidor a la base de datos MySQL.

Paso 2: La aplicación de Drupal es la que accede a los contenidos de la base de datos MySQL.

Fig. 2.13 Respuesta de la base de datos MySQL al servidor.

Paso 3: La base de datos MySQL le provee los datos de respuesta de la petición hecha por Drupal.

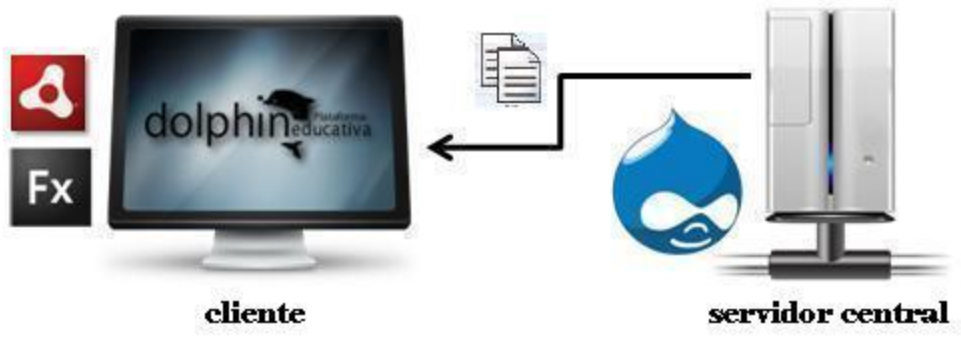

Fig. 2.14 Envío de datos del servidor al cliente.

Paso 4: Drupal envía al cliente todos los contenidos de dicha petición.

Fig. 2.15 El cliente almacena los datos en la base de datos SQLite.

Paso 4: El cliente inserta los contenidos en la base de datos local SQLite.

2.4.1 Sincronización de los contenidos

Para la sincronización de los contenidos se van a utilizar los mismos pasos previamente definidos. Partiendo de que se han identificado dos situaciones, al final se unen en una para darle solución a ambos casos:

2.4.1.1 Situación de Recién instalada

La situación de Recién instalada es cuando por primera vez se instala en el cliente la aplicación y este le hace al servidor la petición de todos los contenidos que se

encuentran en la base de datos central MySQL, para después ser insertados en la base de datos local SQLite que inicialmente se encuentra vacía. A continuación, en la fig. 2.16 aparece un diagrama de secuencia en el cual se describe el proceso de obtención de contenidos en una aplicación recién instalada.

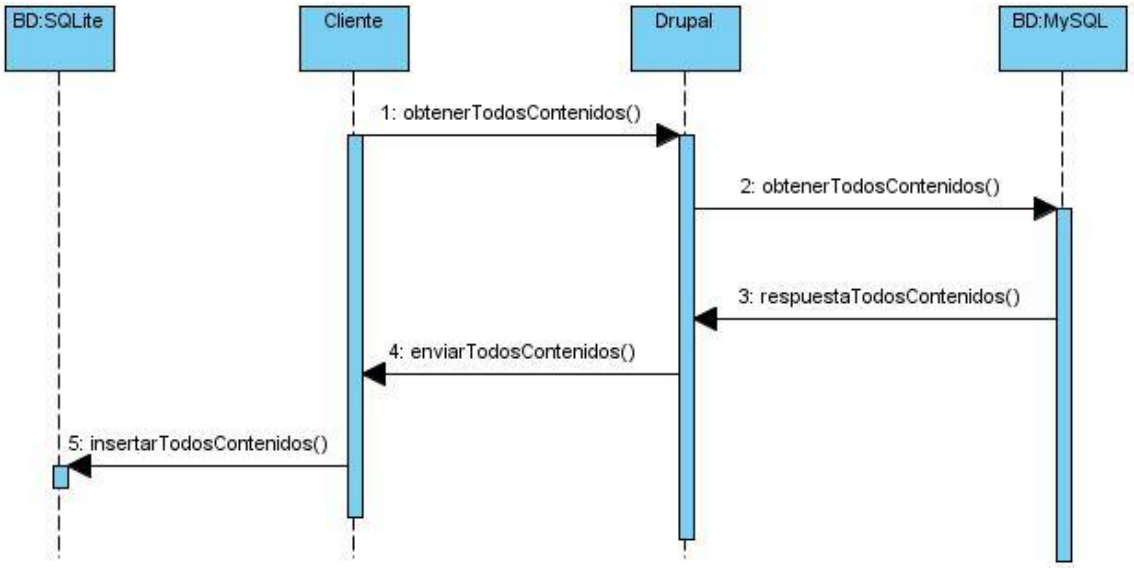

Fig. 2.16 Diagrama de secuencia. Proceso de obtención de contenidos en una aplicación recién instalada.

2.4.1.2 Situación de Aplicación previamente utilizada

La situación de Aplicación previamente utilizada es cuando el cliente ya ha sido usado anteriormente, por lo que este le pide a su base de datos local SQLite cuál es su último contenido y le envía a la base de datos central MySQL la fecha de este último a través del servidor. Luego este le envía al cliente los contenidos actualizados o agregados desde esa fecha, para ser posteriormente insertados en la base de datos SQLite. A continuación, en la fig. 2.17 se muestra el diagrama de secuencia que describe el proceso de obtención de contenidos en una aplicación previamente utilizada.

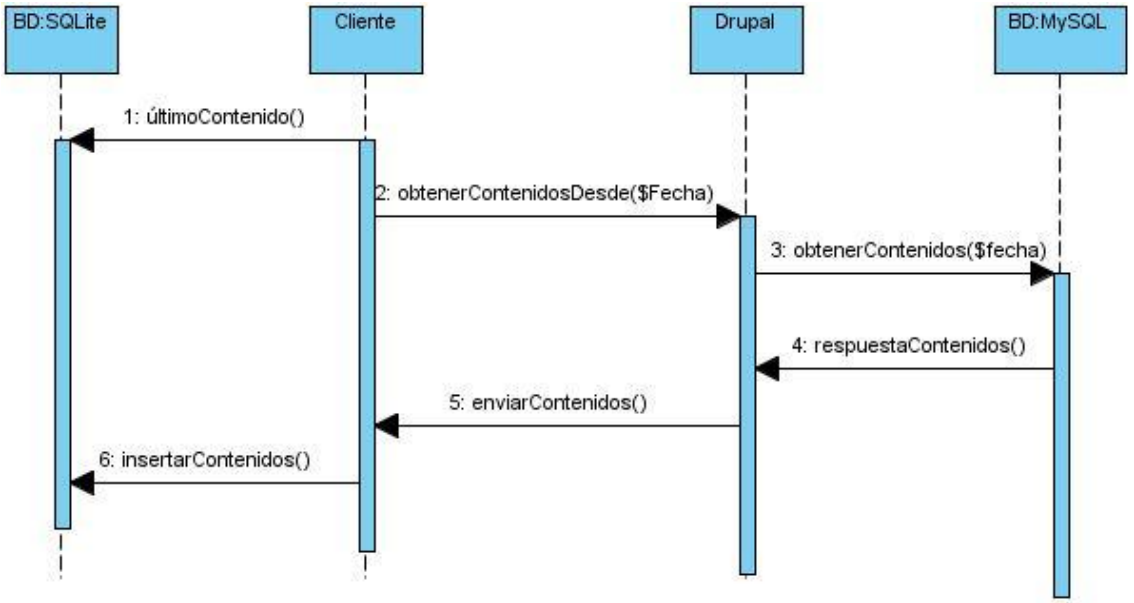

Fig. 2.17 Diagrama de secuencia. Proceso de obtención de contenidos en una aplicación previamente utilizada.

2.4.1.3 Situación general

La situación general es muy parecida a la de la situación de Aplicación previamente utilizada, solo que en esta se incluye el caso de la situación de Recién instalada y es aquí cuando el cliente le pide los contenidos a su base de datos local SQLite que inicialmente se encuentra vacía y esta le devuelve la *fecha cero*, con la cual se hace la petición al servidor central de todos los contenidos, para después ser insertados en la base de datos local SQLite.

Fecha cero: Esta fecha no es más que la que va indicar que la petición que se le haga al servidor central sea de todos los contenidos, que se encuentran en la base de datos.

2.4.1.4 Diagrama Entidad Relación

En la realización del Diagrama Entidad Relación se tuvo en cuenta la base de datos con que cuenta Drupal que se muestra en el anexo 1.

Teniendo en cuenta el Diagrama Entidad Relación de Drupal y después de un análisis, se va a utilizar para gestionar los contenidos entre la base de datos central y la base de datos local, las siguientes tablas representadas en el diagrama entidad relación que

se muestra en la fig. 2.18. Las cuales son un extracto del DER de Drupal antes mencionado.

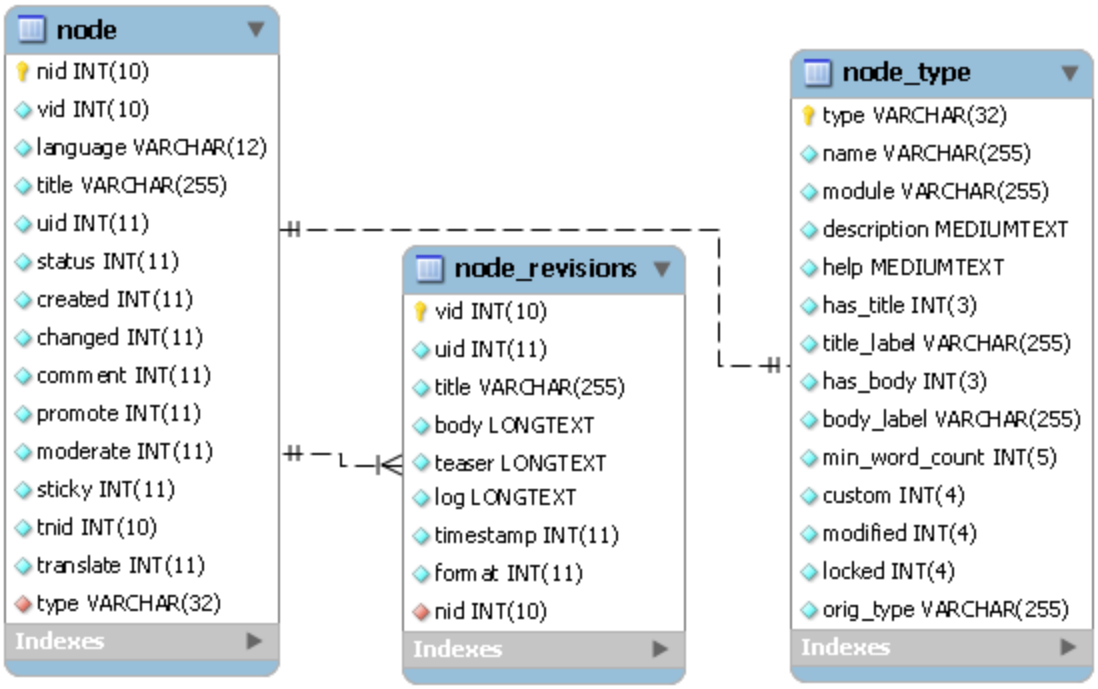

Fig. 2.18 Diagrama Entidad Relación para gestionar los contenidos.

2.4.2 Sincronización de los Ficheros

La sincronización de los ficheros es muy similar a la de los contenidos, pues se realizan los mismos pasos, solo que aquí se utiliza el sistema de archivos, que es donde van a estar guardadas todas las medias a utilizar como imágenes, videos, sonidos, las cuales se encuentran referenciadas desde los contenidos que están en la base de datos central. Para este proceso también se han definido dos situaciones, donde ambas van a estar unidas para dárseles solución en una situación general.

2.4.2.1 Situación de Recién instalada

En la situación de Recién instalada es cuando por primera vez se configura la aplicación de escritorio, esta le hace al servidor la petición de todos los ficheros que se encuentran en el sistema de archivos remoto, para después ser descargados hacia el sistema de archivos local, el cual inicialmente se encuentra vac ío. A continuación, en la fig. 2.19 se muestra el diagrama secuencia para la obtención de todos los ficheros de una aplicación que se instala.

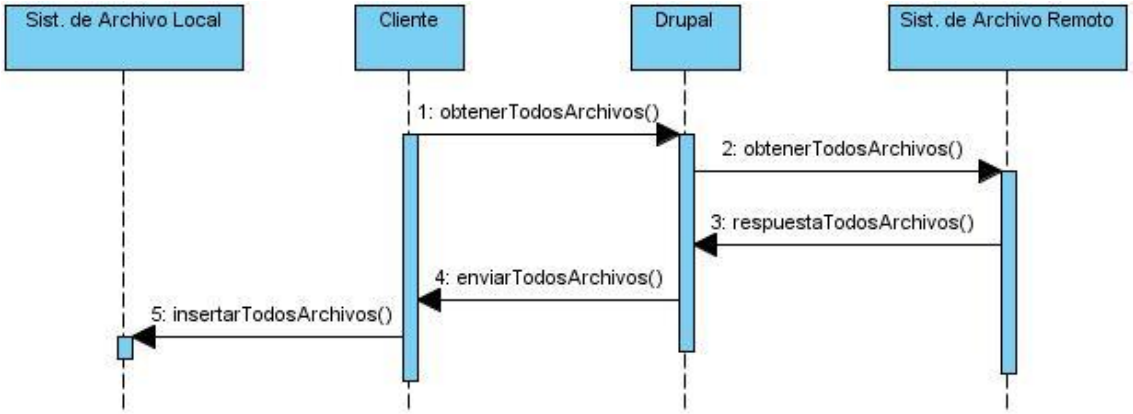

Fig. 2.19 Diagrama de secuencia. Proceso de obtención de todos los ficheros en una aplicación recién instalada.

2.4.2.2 Situación de Aplicación previamente utilizada

La situación de Aplicación previamente utilizada es cuando el cliente ya ha sido usado anteriormente, es aquí cuando este le pide a su sistema de archivos local cuál es su último fichero y le envía al servidor central la fecha de este, para que le haga la petición de los ficheros que fueron actualizados o agregados desde esa fecha en el sistema de archivos remotos y luego este se los envía al cliente a través del servidor, para después ser insertados en el sistema de archivos local. A continuación, en la fig. 2.20 se muestra el diagrama de secuencia de descarga de ficheros en una aplicación previamente utilizada.

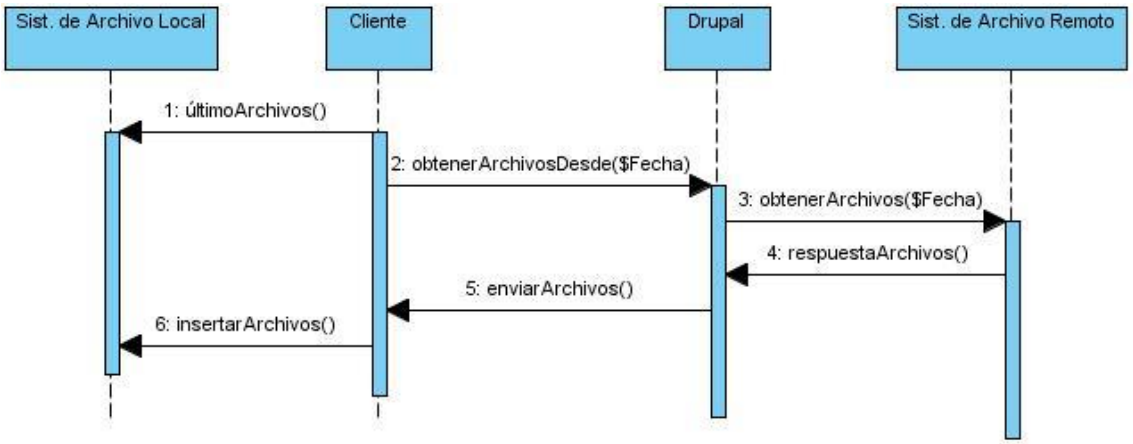

Fig. 2.20 Diagrama de secuencia. Proceso de obtención de ficheros en una aplicación previamente utilizada.

2.4.2.3 Situación general

La situación general es muy parecida a la de la situación de Aplicación previamente utilizada, solo que en esta se incluye el caso de la situación de Recién instalada. Estas dos situaciones anteriormente mencionadas son similares a la de gestionar los contenidos. Independientemente de que sea una situación o la otra el cliente le pide a su sistema de archivos local cuál es su último archivo. En el caso de que sea la situación de Recién instalada el sistema de archivos local estará vacío, por lo que devuelve la *fecha cero*. Esta fecha se utilizará para hacerle la petición al servidor central, de todos los ficheros que se encuentran en el sistema de archivos remotos, los cuales están referenciados desde los contenidos de la base de datos central, para después ser descargados en el sistema de archivos local.

2.4.3 Sincronización de trazas

El proceso de sincronización de las trazas, al igual que el de contenidos y ficheros, se basa en los pasos previamente definidos al inicio del capítulo. Este nos es más que cuando el usuario interactúa con el cliente, se va generando un listado con información, la cual puede ser: desde qué computadora accedió el usuario, qué operaciones hizo en el sistema, a qué hora, en qué fecha y otra información que pueda ser necesaria para la plataforma. Estas trazas pueden ser almacenadas directamente en el servidor central, pero se puede dar el caso que no haya conectividad y no se puedan almacenar. Entonces es aquí donde se tiene que tener la posibilidad de guardarlas localmente en la base de datos SQLite, y una vez que se recupere la conexión entre el cliente y el servidor central, se envíen a este último todas las trazas desde la última sincronización, las cuales son almacenadas en la base de datos MySQL

Para un mejor entendimiento, a continuación se muestra una descripción de los pasos:

Paso 1: El usuario al interactuar con el cliente va generando trazas que inicialmente se almacenan en la base de datos local SQLite.

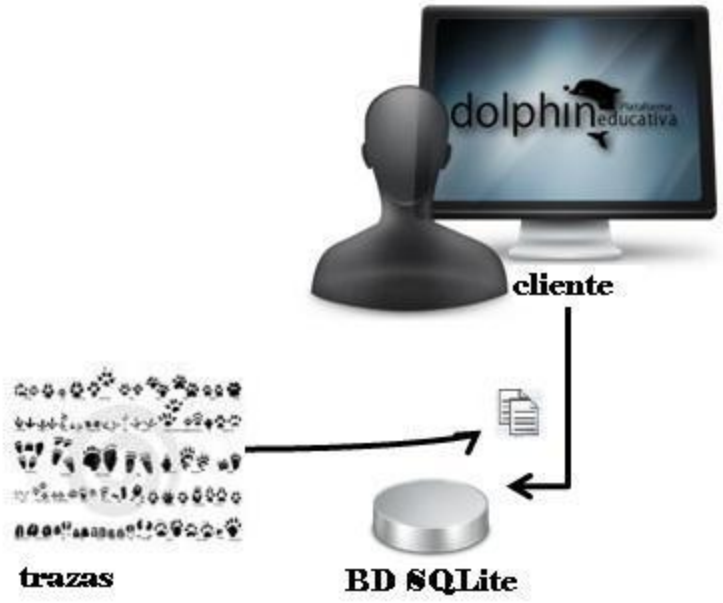

Fig. 2.21 El cliente almacena las trazas en la base de datos SQLite.

Paso 2: El cliente le pide a la base de datos SQLite que le envíe las trazas.

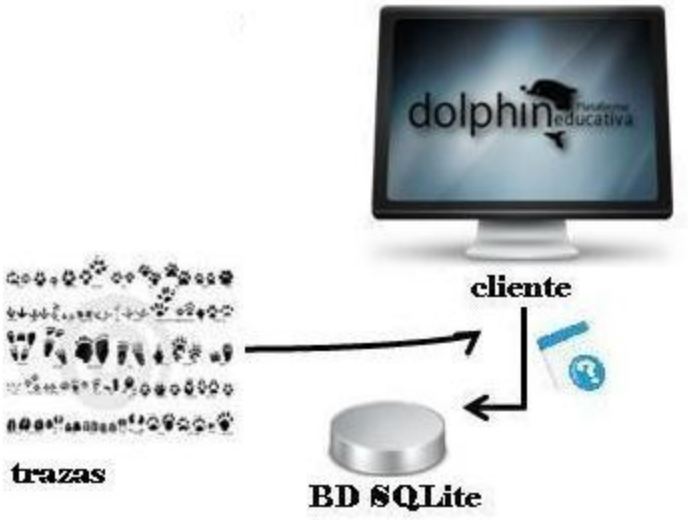

Fig. 2.22 Petición de trazas del cliente a la base de datos SQLite.

Paso 3: La base de datos local SQLite le envía las trazas al cliente.

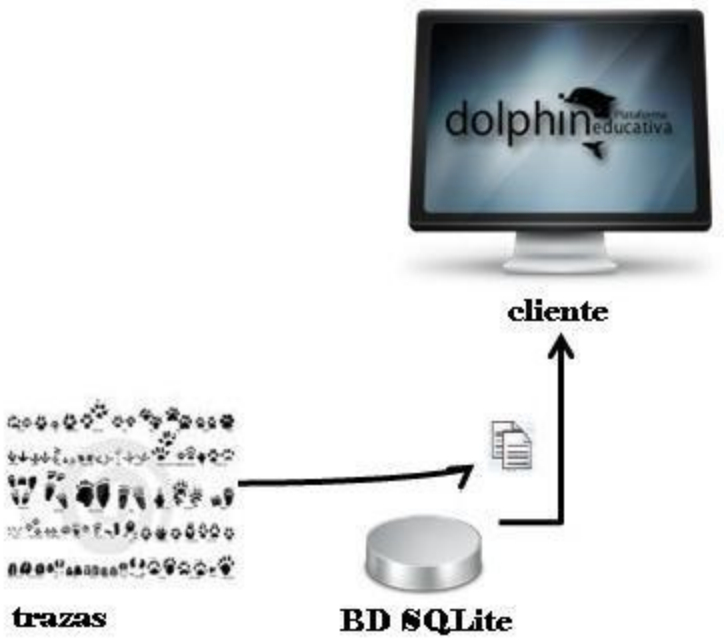

Fig. 2.23 Respuesta de trazas de la base de datos SQLite al cliente.

Paso 4: El cliente verifica si hay conectividad con el servidor central, y en el caso de existir ésta, le envía las trazas.

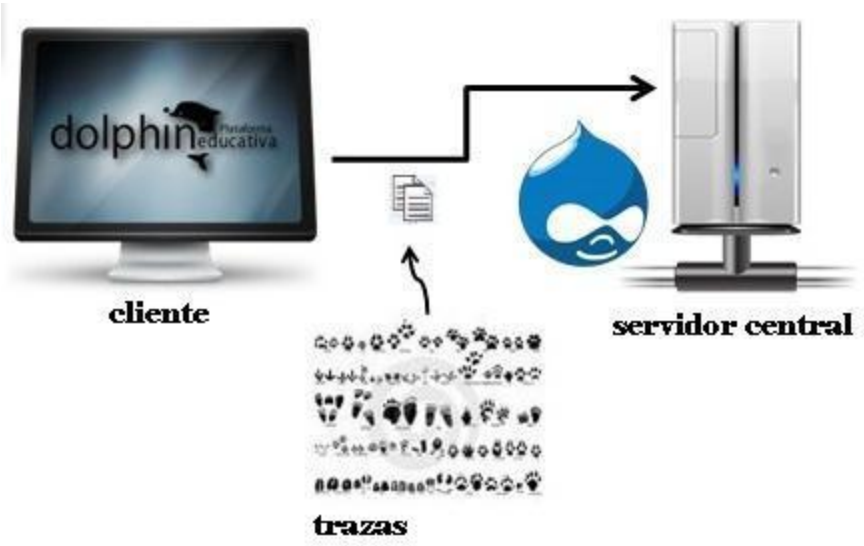

Fig. 2.24 Envío de trazas del cliente al servidor.

Paso 5: El servidor central almacena las trazas en la base de datos MySQL.

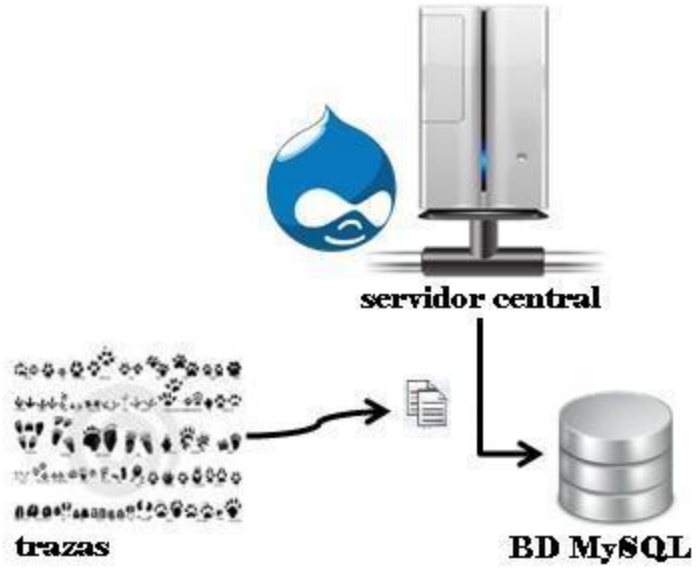

Fig. 2.25 El servidor almacena las trazas en la base de datos MySQL.

2.4.3.1 Diagrama Entidad Relación

Para el proceso de sincronización de las trazas se va a utilizar para la base de datos de MySQL el Diagrama Entidad Relación con que cuenta Drupal que ha sido referenciado en el anexo 1.

Teniendo en cuenta el diagrama anteriormente mencionado, se va a utilizar para gestionar las trazas entre la base de datos local y la base de datos central, las siguientes tablas representadas en el Diagrama Entidad Relación que se muestra en la fig. 2.26; en esencia, estas constituyen un extracto del DER de Drupal anteriormente referenciado.

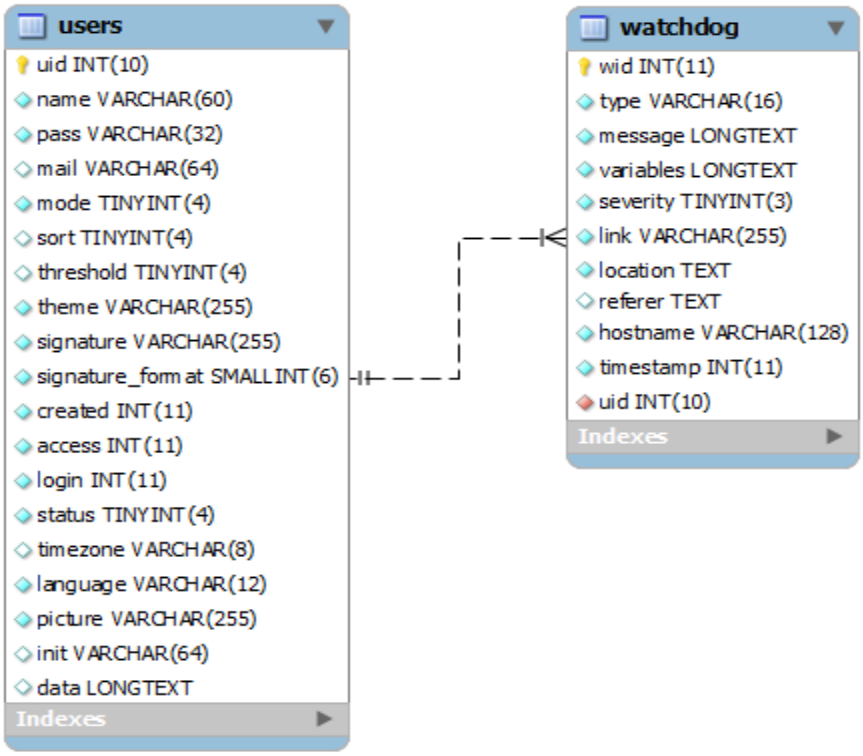

Fig. 2.26 Diagrama Entidad Relación para gestionar las trazas.

La tabla watchdog es la encargada de almacenar toda la información referente a la traza que el usuario va generando a partir de la interacción con el cliente. En ésta se almacena la información, desde qué computadora se accedió, qué hizo el usuario en el sistema, a qué hora y en qué fecha. Adicionalmente, se tiene en cuenta la severidad de la acción realizada.

En la tabla users van a estar recogidos los datos esenciales del usuario que accede al sistema, como por el ejemplo su nombre y contraseña.

Para la realización de la sincronización de las trazas, se tuvo en cuenta los pasos anteriormente descritos y el DER que se propone para gestionar las trazas ya antes referenciado. Por lo que tras un análisis se definen dos situaciones en las que puede estar presente este proceso, llegando a unirse ambos casos para dar una solución, en una situación general.

2.4.3.2 Situación de Conectividad con el servidor

Al usuario interactuar con el cliente se van generando las trazas, este comprueba que haya conectividad con el servidor central y en caso de que exista le envía las trazas a este último, para que después puedan ser insertado en la base de datos central

MySQL. A continuación, en la fig. 2.27 se muestra el diagrama de secuencia del envío de trazas hacia el servidor

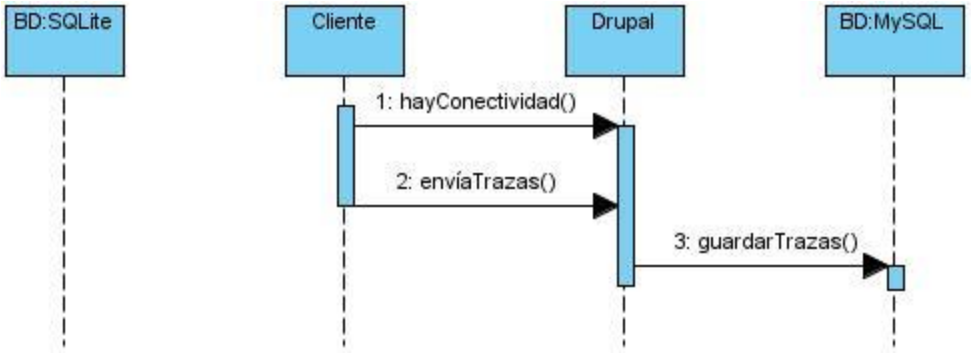

Fig. 2.27 Diagrama de secuencia. Proceso de envío de trazas cuando hay conectividad con el servidor.

2.4.3.3 Situación de Conectividad nula

Cuando el usuario interactúa con el cliente estas trazas que se van generando se van insertando en la base de datos local SQLite. A continuación, en la fig. 2.28 se describe el diagrama de secuencias de almacenamiento de las trazas en situación de conectividad nula.

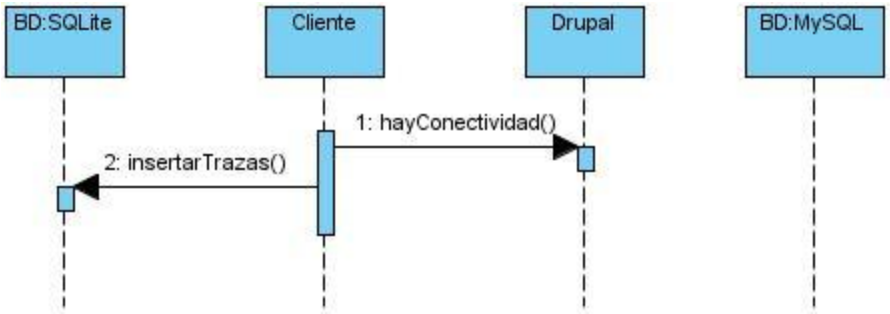

Fig. 2.28 Diagrama de secuencia. Proceso de envío de trazas cuando hay conectividad nula.

2.4.3.4 Situación general

La situación general une los dos casos, ya antes mencionados, el de Conectividad con el servidor y Conectividad nula. Para ello independientemente de que exista o no exista conectividad entre el cliente y el servidor central, las trazas inicialmente se guardan en la base de datos local, después el cliente verifica si existe o no conexión

con el servidor central, para preguntarle a este último cual es la fecha de la última traza, seguidamente se la envía al cliente y este le hace la petición a la base de datos SQLite, después estas son enviadas al servidor, para ser almacenadas en la base de datos MySQL. Al insertarse en esta última, el servidor envía un mensaje de notificación de inserción y son borradas de la base de datos local. A continuación se muestra un diagrama de secuencia para la situación general del envío de trazas hacia el servidor.

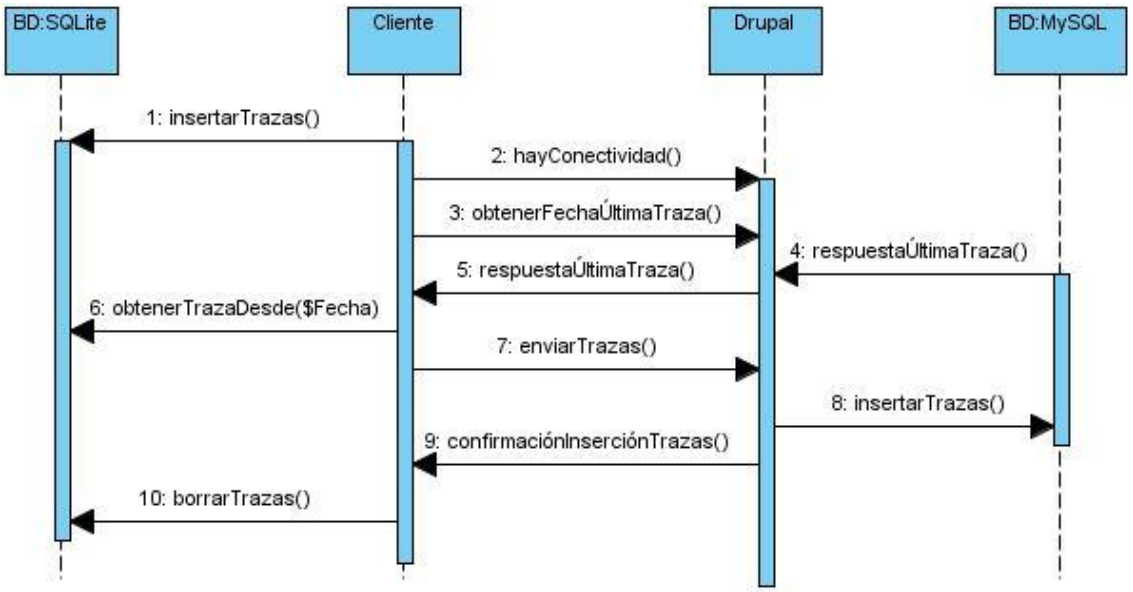

Fig. 2.29 Diagrama de secuencia. Proceso de envío de trazas para la situación general.

2.5 Conclusiones

Con la realización de este capítulo se describieron los procesos de sincronización de contenidos, ficheros y trazas. Para cada uno se tuvo en cuenta dos situaciones, para el caso de los procesos de sincronización de contenidos y ficheros se definieron las situaciones: Recién instalada y Aplicación previamente utilizada y para el de trazas se definió: Conectividad con el servidor y Conectividad nula, apoyándose para su descripción los diagramas de secuencias y englobándose todas estas en una situación general que le diera solución a cada uno de estos procesos. Además, se definieron las tablas que se utilizarán en la base de datos SQLite para la gestión de los contenidos y de las trazas, para el caso de la gestión de los contenidos se definieron las tablas

node, node_type, node_revisions y para la gestión de trazas se definieron las tablas users y watchdog.

Capítulo 3

Descripción de la arquitectura

3.1 Introducción

En el presente capítulo se hace referencia a la propuesta de arquitectura que se va a tener en cuenta para la solución. En esta se hace alusión a como se pueden integrar Drupal y Flex, los servicios que brinda el primero y son utilizados por esta última. Además, se definirá el DER que se utilizará en la base de datos local SQLite y quedará definida en la arquitectura de comunicación, sincronización y almacenamiento del historial en trazas.

3.2 Configuración de Drupal. Instalación

Para la instalación de Drupal se requiere en el sistema un servidor web, el intérprete del lenguaje PHP y un servidor de base de datos soportado por PHP, en este caso MySQL. Este proceso también se puede acelerar a través de una la instalación de un paquete que los contiene todos como: AppServ, XAMPP, WAMP, EasyPHP.

3.2.1 Módulos de Drupal necesarios

Una vez que es instalado Drupal se necesita instalar los siguientes módulos:

- Services.
- AMFPHP.
- Views.

A continuación se da una breve explicación de los mismos:

Services: Este tiene como objetivos crear una API que unifique los servicios web de Drupal para ser expuestos en diferentes formatos. Proporciona un explorador de servicios para poder probar los métodos que permitirán acceder a los nodos de la base de datos.

AMFPHP: Proporciona soporte AMFPHP al módulo de servicios antes mencionado, y juntos permiten comunicar información entre Drupal y Flex.

Views: Permite a los desarrolladores, sin tener conocimiento de SQL, realizar consultas personalizadas a la base de datos, pues su principal finalidad es crear contenidos personalizados, a partir de consultas donde se guarda la información de los nodos. Por ejemplo en una vista se puede hacer un filtro por el tipo de nodo, así como también se puede escoger los campos que se quieran mostrar o se le puede pasar un argumento para que te devuelva todos los nodos que coincidan con el mismo.

Integrando los módulos de services y views, se pueden realizar consultas personalizadas a la base de datos, las cuales puedan ser utilizadas desde aplicaciones externas.

3.2.1 Vistas necesarias

Para poder realizar el proceso de sincronización, tanto de los contenidos, ficheros y trazas, se definieron vistas para acceder a los contenidos de la base de datos, y para que después desde Flex se pueda utilizar las mismas. A continuación se explican las vistas que se definieron en ambos casos:

3.2.1.1 Proceso de sincronización de contenidos

Vista definidas:

- Contenidos_Nuevos_Desde (fecha)

El método consiste en obtener todos los contenidos de la base de datos desde una fecha determinada, para que después estos puedan ser insertados en la base de datos local.

3.2.1.2 Proceso de sincronización de ficheros

Vista definidas:

- Archivos_Nuevos_Desde_Fecha(Fecha)

Este método permite que a través del servicio directory.getFileList, retornar un XML con todos los archivos de una fecha especificada.

3.2.1.3 Proceso de sincronización de trazas

Vista definidas:

- Devolver_Fecha_Última_Trazas()

Este método devuelve la fecha de la última traza almacenada en la base de datos, para que después el cliente le pida a su base de datos local SQLite, todas las trazas a partir de esa fecha y se le puedan enviar a Drupal para que las almacene en la base de datos central MySQL.

- Trazas.save()

El método este permitirá guardar las trazas en la base de datos MySQL.

3.3 Flex

Flex es un framework, que posibilita facilidad en el desarrollo del software, debido a que abstrae a los programadores de definir una estructura global. Esta proporciona un esqueleto que da soporte a otros proyectos de software, permitiéndosele llegar a una solución de manera rápida y sencilla.

Para utilizar Flex con AMFPHP, se hace uso de la clase RemoteObject que se encuentra en el paquete mx.rpc.remoting. Esto permite la comunicación de Flex con Drupal, facilitándole al primero que pueda obtener información de las vistas a través de la interacción con los servicios que ofrece el último.

3.3.1 Diseño de clases de comunicación con Drupal

Para construir el diseño de clases de comunicación con Drupal se utilizó el diagrama de clases, pues permite describir la arquitectura y la relación que se establece entre cada una de ellas, además de obtener una vista más conceptual y detallada de la solución. (Ver anexo 2).

Como se muestra en el anexo 2. El diagrama está estructurado por paquetes, lo que permite una mejor organización del código. Esta estructura es muy utilizada en el caso del lenguaje ActionScript, y es inversa a la que es asignada a los nombres de dominio de los sitios web. Por ejemplo, si se crea un proyecto para www.nombreSitio.cu, la estructura de los paquetes es **cu** como paquete principal, **nombreSitio** como

secundario e internamente se organizaría el proyecto. Finalmente, quedaría **cu.nombreSitio** como el paquete raíz.

Siguiendo la estructura antes mencionada, el paquete de clases de la arquitectura de la solución quedaría de la forma siguiente: cu.dolphin.drupal. A continuación se explican de forma breve la utilización de las clases presente en el diagrama anterior.

DrupalInterface: Es la clase principal. Esta va a ser utilizada por los programadores que quieran usar el framework de comunicación con Drupal. Además, ella hereda de la clase EventDispatcher para poder lanzar eventos.

Paquete node: Se encuentra dentro del paquete Drupal. En él están todas las clases que se utilizan desde la clase DrupalInterface. A continuación se muestra una explicación de cada una de las clases que están contenidas en él.

ContentView: Es la clase encargada de acceder remotamente al servidor y acceder a las vistas.

ContentNode: Es la clase encargada de acceder remotamente al servidor y acceder a los nodos y modificarlos.

Content: Esta le proporciona un arreglo de contenidos tanto a las clases ContentView como ContentNode.

IContent: Es una interfaz que contiene métodos a implementar por las clases ContentView y ContentNode.

Paquete utils: Se encuentra dentro del paquete Drupal. En él se encuentran todas las clases van a ser utilizadas desde las clases que están contenidas en el paquete node.

Remote: Es una clase que proporciona la comunicación entre Flex y Drupal utilizando AMFPHP.

Clases de constantes: se utilizan para evitar errores de comparación de cadenas en funciones literales de los frameworks. Dentro de estas están las clases servicios de Drupal, métodos de los servicios y eventos.

DrupalServices: Esta clase te muestra los servicios con que cuenta Drupal, como por ejemplo: node, system, user, views.

ServicesMethods: Es clase te muestra los métodos de los servicios antes mencionados, como por ejemplo NODE_GET, NODE_SAVE, NODE_DELETE, VIEWS_GET, VIEWS_EXPORT, VIEWS_IMPORT.

Events: Esta clase los programadores la utilizan para obtener respuestas de la petición que hagan a través de DrupalInterface.

Las siguientes clases son propias del framework de Flex a utilizar, pero se importan para lograr la integración de la arquitectura con el mismo.

flash.events.EventDispatcher

flash.events.Event

mx.rpc.remoting.RemoteObject

mx.rpc.events.FaultEvent

mx.rpc.events.ResultEvent

3.4 Diseño de la base de datos

Como se explicó anteriormente, tanto para el proceso de sincronización de contenidos como para el de trazas se va hacer uso de una base de datos central y una local. La central va a estar soportada por MySQL y utilizará el DER de la base de datos del CMS Drupal, como se muestra en el anexo 1; y en la local con SQLite se usarán algunas de las tablas del DER de Drupal, en esencia constituyen un extracto de la base de datos de Drupal. A continuación se muestra el DER que se va a utilizar en la base de datos de SQLite.

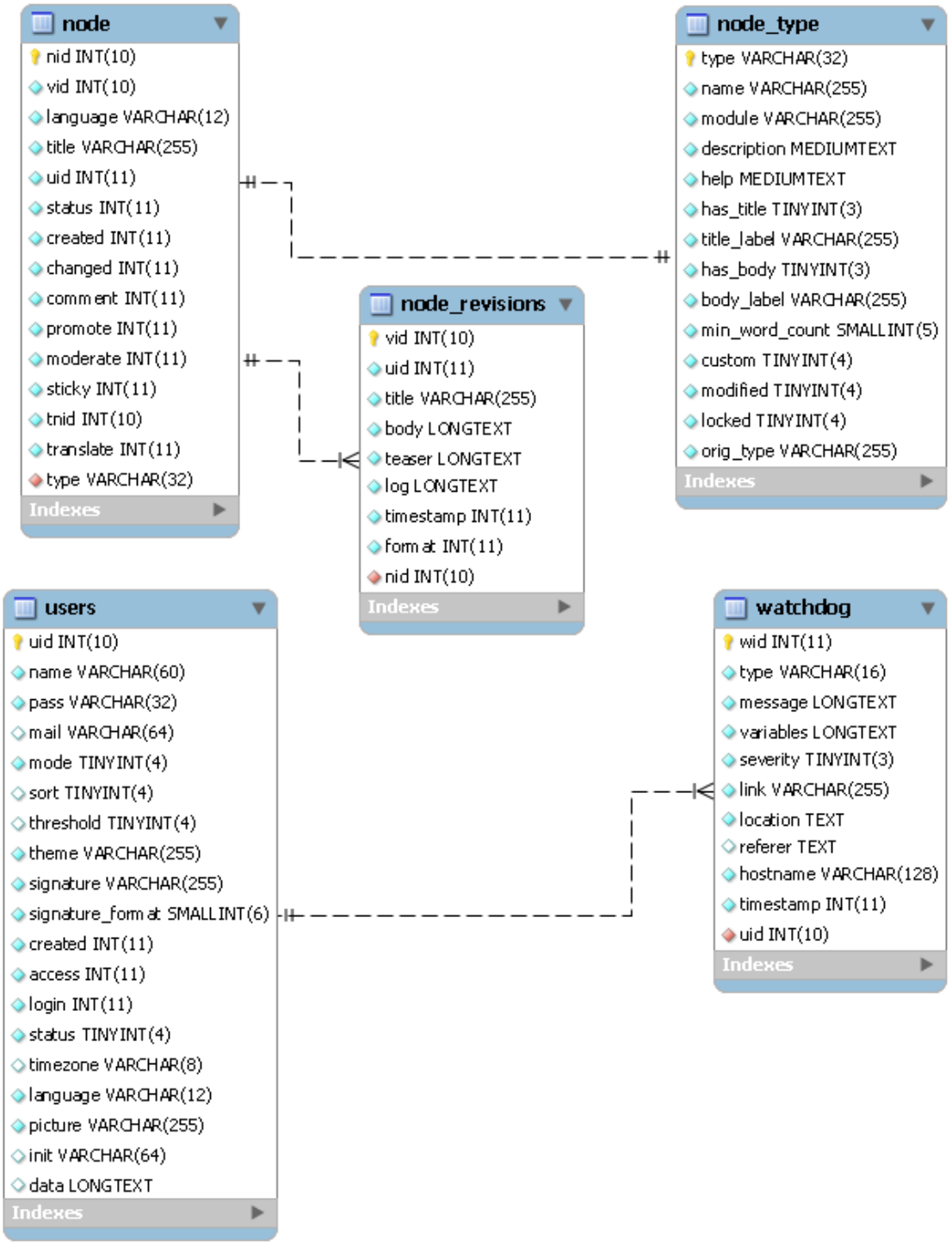

Fig. 3.2 Diagrama Entidad Relación de SQLite.

3.5 Conclusiones

En el presente capítulo se abordaron elementos relacionados con la arquitectura de la solución. Se definieron cuáles son los módulos para la configuración de Drupal, así como las vistas que se necesitan para acceder a los datos de la base de datos y después sean utilizadas a través de las clases que se han definido en Flex. Las clases están definidas mediante un diagrama de clases el cual está estructurado por paquetes. Además, se definió cual es el DER de la base de datos de SQLite, este consta de las tablas node, node_type, node_revisions, users y watchdog.

Capítulo 4

Exploración y planificación

4.1 Introducción

En este capítulo se hace alusión a las fases de exploración y planificación, propias de la metodología de desarrollo seleccionada para la construcción del sistema. El objetivo principal de estas es conocer el alcance del producto a desarrollar y estimar los tiempos de implementación de cada historia de usuario. Se exponen además los artefactos generados durante el transcurso de las mismas.

4.2 Exploración

La metodología de desarrollo XP comienza con su fase de exploración. Es la etapa del proceso de desarrollo de software que propone XP para comenzar la construcción de un producto. Sobre todo los clientes definen historias de usuario, que son la forma de definir los requisitos del sistema a implementar, así como la familiarización de los equipos de trabajo con las tecnologías y herramientas seleccionadas para la construcción del sistema.

4.2.1 Historias de usuario

Las historias de usuario (HU) son la forma en que la metodología de desarrollo ágil XP, especifica los requisitos del sistema. Estas son una forma muy sencilla de administrar las necesidades que tiene el usuario, redactadas brevemente por el mismo, aunque los desarrolladores pueden brindar también su ayuda en la identificación de las mismas. Estas son la base para realizar la estimación, así como la planificación del proyecto.

Para la construcción de la presente solución se definieron tres historias de usuario. Cada una de ellas tiene como objetivo principal ofrecer una funcionalidad al sistema basada en la arquitectura de comunicación del mismo con el servidor de Drupal. La construcción de cada una de estas historias de usuario lleva, de manera paralela, el desarrollo de la funcionalidad al cliente y la construcción de la solución en cuestión.

A continuación se muestra la plantilla utilizada para recoger la información relacionada con las historias de usuario determinadas:

Plantilla de Historias de Usuarios:

Historia de Usuario	
No.: Número sucesivo a partir de 1.	**Nombre:** Identifica la historia de usuario en cuestión.
Usuario: Quien ejecuta la historia de usuario.	
Prioridad en el Negocio: Define la relevancia e impacto de la historia de usuario para el negocio de acuerdo con las necesidades del usuario.	**Nivel de Complejidad:** Define la dificultad técnica que supone desarrollar la historia de usuario desde el punto de vista del programador.
Puntos de Estimación: Permiten estimar duración de implementación.	**Iteración Asignada:** Precisa la iteración a la que pertenece la historia de usuario.
Descripción: Explica en qué consiste la historia de usuario, teniendo en cuenta las acciones realizadas por el usuario y la respuesta brindada por el sistema.	
Información adicional (Observaciones): Información extra que se estime agregar para hacer más comprensible la historia de usuario. Por ejemplo: conceptos, post-condiciones, relación con otros requisitos, etc.	

Durante este proceso se identifican 3 historias de usuarios que se detallan a continuación.

Historia de Usuario	
No.: 1	**Nombre**: Configurar cliente de escritorio
Usuario: Todos	
Prioridad en el Negocio: 1	**Nivel de Complejidad: alta**
Puntos de Estimación: 3	**Iteración Asignada: 1**
Descripción: El objetivo es brindar al usuario una breve introducción al proceso que continúa (la configuración de la aplicación) Es comparable con los Asistentes de instalación tradicionales para aplicaciones de escritorio. Se le ofrece la opción de	

continuar o cancelar el proceso de configuración. Si no se configura la aplicación, se interrumpe el inicio y cuando se ejecute nuevamente, el proceso de configuración se vuelve a presentar. Este es un aspecto obligatorio a especificar en los primeros momentos de uso del sistema.

Información adicional (Observaciones):

Historia de Usuario

No.: 2.	**Nombre**: Iniciar sesión
Usuario: Todos	
Prioridad en el Negocio: 1	**Nivel de Complejidad: alta**
Puntos de Estimación: 3	**Iteración Asignada: 2**

Descripción: El objetivo es brindar al usuario una presentación de la página inicio en la cual va a seleccionar la forma de registro con la que desee entrar, ya sea creando su cuenta, como usuario autentificado, invitado o pedir ayuda en caso de olvidar la contraseña. Este es un aspecto obligatorio a especificar en los primeros momentos de uso del sistema.

Información adicional (Observaciones):

Historia de Usuario

No.: 3.	**Nombre**: Generar Trazas
Usuario: Todos	
Prioridad en el Negocio: 1	**Nivel de Complejidad: alta**

Puntos de Estimación: 3 **Iteración Asignada: 3**

Descripción: El objetivo de estas es que, como resultado de la interacción del usuario con el cliente se vayan generando trazas, estas se vayan guardando localmente en la base de datos SQLite, para después solicitarle al servidor central la fecha de la última que este tiene almacenada en su base de datos MySQL y con esta respuesta el cliente hacerle la petición a su base de datos local, y seguidamente enviarle las trazas al servidor.

Información adicional (Observaciones):

Teniendo en cuenta, que en las tablas que recogen la información de las historias de usuarios, uno de los parámetros que se usó fue el punto de estimación, el cual equivale a una semana. Es necesario aclarar que 1 semana laboral equivale a los 5 días laborales de la misma, pues generalmente se estiman los tiempos de desarrollo en base a los 7 días de la semana, trayendo consigo cálculos erróneos a la hora de estimar.

Mes de ejemplo						
domingo	**lunes**	**martes**	**miércoles**	**jueves**	**viernes**	**sábado**
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Fin de semana

Próximo mes

Días laborales

4.3 Planificación

Durante esta fase se realiza una estimación del esfuerzo que costará implementar cada una de las historias de usuario por parte del equipo, teniéndose en cuenta la prioridad de cada una de ella que establece el cliente. Seguidamente, se procede a organizarlas en las iteraciones correspondientes, en dependencia de la prioridad especificada por el cliente y del tiempo de desarrollo de cada una.

En XP se realizan dos actividades de planificación fundamentales, la planificación de liberaciones, donde participa el cliente de forma decisiva, y la planificación de la iteración, donde participan los desarrolladores definiendo las tareas, estimándolas, comprometiéndose e implementándolas.

4.3.1 Estimación de esfuerzos por historias de usuario

Para el buen desarrollo del sistema propuesto, se realizó una estimación para cada una de las historias de usuario identificadas, llegando a los resultados que se muestran a continuación:

Historia de usuario	**Puntos de estimación**
Configurar cliente de escritorio	2 semanas
Iniciar sesión	2 semanas
Generar Trazas	3 semanas

4.3.2 Iteraciones

Después de ser descritas e identificadas las historias de usuario y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a realizar la planificación de la etapa de implementación del sistema. Este plan especifica exactamente cuáles historias de usuario serán implementadas para cada iteración del

sistema y cuántas iteraciones serán necesarias realizar sobre el sistema para su terminación.

Para organizar las iteraciones no es recomendable extenderlas en más de 1 mes laboral, lo que sería generalmente 20 ó 22 días, según los días laborables que tiene una semana y como se expuso anteriormente. Los plazos de entrega y retroalimentación largos atentan contra el cumplimiento de los objetivos del cliente, sometidos a pequeños cambios de manera constante. En resumen, alargar una iteración por más de 1 mes, es una práctica muy negativa para el desarrollo de un producto. A continuación se muestra el plan de iteraciones a tener en cuenta para el desarrollo del proyecto.

Iteraciones	**Orden de las Historias de usuario a implementar**	**Cantidad de tiempo de trabajo**
Iteración 1	Configurar cliente de escritorio	2 semanas
Iteración 2	Iniciar sesión	2 semanas
Iteración 3	Generar Trazas	3 semanas

4.3.3 Plan de entregas

El plan de entregas (liberaciones) es el compromiso final del equipo de desarrollo con los clientes. Este tiene como objetivo definir el número de entregables que se realizarán en el transcurso del proyecto y las iteraciones que se requieren para desarrollar cada uno.

4.4 Conclusiones

Durante el desarrollo de este capítulo se hace referencia a las dos primeras fases de la metodología de desarrollo de software a utilizar, fase de exploración y planificación del sistema. En este se realizó la documentación de la primera etapa del ciclo de vida de la solución propuesta como los artefactos siguientes: primeramente se definieron tres historias de usuario y el plan de iteraciones, los cuales fueron detallados de forma clara.

Capítulo 5

Tareas de implementación

5.1 Introducción

En el presente capítulo tiene como objetivo definir las tareas de ingenierías, que darán cumplimiento a la implementación de cada una de las historias de usuario de cada iteración, correspondiente con el plan definido anteriormente. Además, se realizará una breve descripción de cada una de ellas, teniendo en cuenta los elementos que se definen en la arquitectura del sistema.

5.2 Desarrollo de las iteraciones

Durante la Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar. Para su cumplimiento, inicialmente se realiza una revisión del plan de iteraciones, en caso de que haya que realizar algún cambio. En el transcurso de las iteraciones se realiza la implementación de las HU seleccionadas para ser realizadas en cada una de ellas. Estas se descomponen en tareas de desarrollo, asignando a un grupo de desarrollo (o una persona), responsable de su implementación. Las mismas son usadas únicamente por los programadores, pueden ser escritas en lenguaje técnico y no necesariamente entendible por el cliente.

A continuación se describen las tareas detalladas de cada una de las tres iteraciones definidas en el plan de iteraciones.

5.2.1 Iteración 1

Con el objetivo de desarrollar la primera iteración, la cual consta de la HU 1 Configurar cliente de escritorio, se trazaron 3 tareas, que se describen a continuación:

Tarea 1: Diseñar Arquitectura.

Tarea 2: Descargar contenidos de la base de datos.

Tarea 3: Descargar sistema de archivos.

Tarea

Número de tarea: 1	**Número de HU**: 1
Nombre de la tarea: Diseñar Arquitectura.	
Tipo de tarea: Desarrollo	**Estimación**: Cinco días
Fecha inicio: Semana 1 día 1	**Fecha fin**: Semana 1 día 5
Programador responsable: Dairelys Enriquez García	

Descripción: Desarrollar la arquitectura que se va a utilizar según la propuesta de solución.

Tarea

Número de tarea: 2	**Número de HU**: 1
Nombre de la tarea: Descargar contenidos de la base de datos.	
Tipo de tarea: Desarrollo	**Estimación**: Tres días
Fecha inicio: Semana 2 día 1	**Fecha fin**: Semana 2 día 3
Programador responsable: Programador 1	

Descripción: Para descargar los contenidos de la base de datos de MySQL se hace uso de la clase DrupalInterface, la cual accede a la clase ContentView, esta última llama a la vista Contenidos_Nuevos_Desde (0), para obtener todos los contenidos y que luego sean almacenados en la base de datos SQLite.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentViews

Tarea

Número de tarea: 3	**Número de HU**: 1
Nombre de la tarea: Descargar sistema de archivos.	
Tipo de tarea: Desarrollo	**Estimación**: Dos días
Fecha inicio: Semana 2 día 4	**Fecha fin**: Semana 2 día 5
Programador responsable: Programador 1	

Descripción: Para descargar el sistema de archivos, se hace uso de la clase DrupalInterface, la cual accede a la clase ContentNode, esta última llama a la vista Archivos_Nuevos_Desde_Fecha(0), para obtener todos los ficheros y que luego sean almacenados en el sistema de archivos local.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentNode

5.2.2 Iteración 2

Con el objetivo de desarrollar la segunda iteración, la cual consta de la HU 2 Iniciar sesión, se trazaron 4 tareas, que se describen a continuación:

Tarea 1: Verificar contenidos nuevos.

Tarea 2: Verificar ficheros nuevos.

Tarea 3: Actualizar contenidos nuevos.

Tarea 4: Actualizar ficheros nuevos.

Tarea

Número de tarea: 1	**Número de HU**: 2
Nombre de la tarea: Verificar contenidos nuevos.	
Tipo de tarea: Desarrollo	**Estimación**: Dos días
Fecha inicio: Semana 1 día 1	**Fecha fin**: Semana 1 día 2
Programador responsable: Programador 1	

Descripción: Para verificar contenidos nuevos se hace uso de la clase DrupalInterface, la cual accede a la clase ContentNode, esta última llama a la vista Contenidos_Nuevos_Desde (fecha_actual), para obtener todos los contenidos desde esa fecha y luego la respuesta sea almacenada en la base de datos SQLite.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentViews

Tarea

Número de tarea: 2	**Número de HU**: 2
Nombre de la tarea: Verificar ficheros nuevos.	
Tipo de tarea: Desarrollo	**Estimación**: Dos días
Fecha inicio: Semana 1 día 3	**Fecha fin**: Semana 1 día 4
Programador responsable: Programador 1	

Descripción: Para verificar ficheros nuevos, se hace uso de la clase DrupalInterface, la cual accede a la clase ContentNode, esta última llama a la vista Archivos_Nuevos_Desde_Fecha(fecha_actual), para obtener todos los ficheros desde esa fecha y que luego sean almacenados en el sistema de archivos local.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentNode

Tarea

Número de tarea: 3	**Número de HU**: 2
Nombre de la tarea: Actualizar contenidos nuevos.	

Tipo de tarea: Desarrollo	**Estimación**: Tres días
Fecha inicio: Semana 2 día 1	**Fecha fin**: Semana 2 día 3
Programador responsable: Programador 1	

Descripción: Desarrollar las clases encargadas de que el cliente en Flex le haga la petición de los contenidos que fueron modificados o agregados en el servidor a partir de una fecha.

Tarea

Número de tarea: 4	**Número de HU**: 2
Nombre de la tarea: Actualizar ficheros nuevos.	
Tipo de tarea: Desarrollo	**Estimación**: Dos días
Fecha inicio: Semana 2 día 4	**Fecha fin**: Semana 2 día 5
Programador responsable: Programador 1	

Descripción: Desarrollar las clases encargadas de que el cliente en Flex le haga la petición de los ficheros que fueron modificados o agregados en el servidor a partir de una fecha.

5.2.3 Iteración 3

Con el objetivo de desarrollar la tercera iteración, la cual consta de la HU 3 Generar trazas, se trazaron 3 tareas, que se describen a continuación:

Tarea 1: Almacenar trazas localmente.

Tarea 2: Enviar trazas al servidor.

Tarea 3: Solicitar fecha de última traza.

Tarea

Número de tarea: 1	**Número de HU**: 3
Nombre de la tarea: Almacenar trazas localmente.	
Tipo de tarea: Desarrollo	**Estimación**: Cinco días
Fecha inicio: Semana 1 día 1	**Fecha fin**: Semana 1 día 5
Programador responsable: Programador 1	

Descripción: Desarrollar las clases encargadas de que el cliente en Flex almacene en la base de datos local SQLite las trazas generadas por el usuario.

Tarea

Número de tarea: 2	**Número de HU**: 3
Nombre de la tarea: Enviar trazas al servidor.	
Tipo de tarea: Desarrollo	**Estimación**: Cuatro días
Fecha inicio: Semana 2 día 1	**Fecha fin**: Semana 2 día 4
Programador responsable: Programador 1	

Descripción: Para enviar trazas al servidor, se hace uso de la clase DrupalInterface, la cual accede a la clase ContentNode, esta última las envía al servidor a través de la vista Trazas.save(), para que puedan ser almacenadas en la base de datos MySQL.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentNode

Tarea

Número de tarea: 3	**Número de HU**: 3
Nombre de la tarea: Solicitar fecha de última traza.	

Tipo de tarea: Desarrollo	**Estimación**: Tres días
Fecha inicio: Semana 3 día 1	**Fecha fin**: Semana 3 día 3
Programador responsable: Programador 1	

Descripción: Para solicitar fecha de la última traza, se hace uso de la clase DrupalInterface, la cual accede a la clase ContentNode, esta última llama a la vista Devolver_Fecha_Última_Trazas(), para obtener de la base de datos MySQL esta fecha.

Paquete: cu.dolphin.drupal

Clases: DrupalInterface, ContentNode

5.3 Conclusiones

En el presente capítulo se definieron las tareas correspondientes a cada iteración, proporcionando una descripción detallada de las mismas y teniendo en cuenta las clases de Flex que se definieron para la comunicación con Drupal, así como las vistas necesarias para la configuración de este último, permitiéndole a los desarrolladores un mejor entendimiento a la hora de realizar la implementación de dichas tareas.

Conclusiones

Luego de la realización del estudio del estado del arte de los principales enfoques acerca de los procesos de sincronización y las metodologías y herramientas existentes, se desarrolló la descripción de la propuesta de solución y la definición de la arquitectura a utilizar. Se describieron detalladamente los procesos de sincronización, comunicación y trazas entre el cliente y el servidor para una futura implementación.

A su vez, el uso de la metodología de desarrollo XP, permitió desarrollar a través de sus fases iniciales los artefactos requeridos para su posterior implementación en un plazo corto de tiempo. Dándose así cumplimiento al objetivo definido inicialmente, el cual es, realizar el análisis y diseño del módulo encargado de la comunicación, sincronización e historial de forma dinámica de las interacciones realizadas con los contenidos de la plataforma educativa Dolphin.

La presente investigación permitirá centralizar y actualizar la información generada por las interacciones realizadas por los estudiantes con los contenidos de la plataforma educativa Dolphin y que el profesor pueda darle seguimiento a cada una de ellas.

Recomendaciones

Una vez concluido el desarrollo de este trabajo se recomienda:

- Realizar la implementación del módulo, a partir de los artefactos generados.
- Integrar el módulo desarrollado con los demás que forman parte de la plataforma educativa Dolphin.

Referencias Bibliográficas

Adisley Flores Coronado, Imirys Rovira Prieto. 2007. Sistema control de acceso a comedores en la UCI. Ciudad de La Habana : s.n., 2007.

Corporation, Microsoft. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165711%28SQL.90%29.aspx.

—. 2010. Microsoft Developer Network. [En línea] 2010. http://msdn.microsoft.com/es-es/library/ms151198.aspx.

DBBalance. 2010. DBBalance. [En línea] febrero de 2010. http://www.dbbalance.com/contents/page.asp?contentPageID=10.

Educativas, Instituto de Tecnologías. Instituto de Tecnologías Educativas. [En línea] http://usuarios.pntic.mec.es/sqlite.php.

González, Francisco Ruiz. 2000. Arquitecturas de sistemas de bases de datos. [En línea] 2000. http://148.208.239.12/dep/sada/carreras/Ingenieria%20en%20Sistemas%20Computacionales/5to%20Semestre/Taller%20de%20Base%20de%20Datos/taller_bd/ARKI_SBD.pdf.

Grajeda, Marcos Alejandro. 2009. Adobe Air, Adobe Flex. ISSUU. [En línea] 2009. http://issuu.com/mgrajeda/docs/adobe_flex_-_air.

Inc, Adobe Systems. 2009. Adobe Systems Inc. Adobe Systems Inc. [En línea] 2009. http://www.adobe.com/products/flex/.

Incorporated, Adobe Systems. 2009. Adobe Systems Incorporated. Adobe Systems Incorporated. [En línea] 2009. http://www.adobe.com/products/air/.

J. Rumbaugh, G. Booch, 1. Jacobson. 1998. El Lenguaje de Modelado Unificado. Manual de Referencia. Cupertino, California : Addison Wesley, 1998.

—. 2000. El Proceso Unificado de desarrollo de Software. Madrid : Addison Wesley, 2000.

Microsoft Corporation. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165742%28SQL.90%29.aspx.

—. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165720%28SQL.90%29.aspx.

Oracle. 2001. Oracle Corporation. Oracle Corporation. [En línea] 2001. http://www.macs.hw.ac.uk/dept/facil/guides/oracle_9i/doc/server.901/a87499/repoverv.htm.

Ossa, Luis de la. 2009. Departamento de Sistemas Informáticos. Departamento de Sistemas Informáticos. [En línea] 2009. http://www.dsi.uclm.es/asignaturas/42548/trasp/Tema9Pr-2.pdf.

Pecos, Daniel. 2010. danielpecos.com. danielpecos.com. [En línea] 2010. http://danielpecos.com/docs/mysql_postgres/x15.html#AEN30.

Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. 2008. Adonisnet's Weblog. [En línea] junio de 2008. http://adonisnet.files.wordpress.com/2008/06/articulo-metodologia-de-sw-formato.doc..

Talend. 2008. Talend. [En línea] 2008. http://es.talend.com/solutions-data-integration/data-synchronization.php.

Bibliografía

Corporation, Microsoft. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165711%28SQL.90%29.aspx.

Educativas, Instituto de Tecnologías. Instituto de Tecnologías Educativas. [En línea] http://usuarios.pntic.mec.es/sqlite.php.

J. Rumbaugh, G. Booch, 1. Jacobson. 1998. El Lenguaje de Modelado Unificado. Manual de Referencia. Cupertino, California : Addison Wesley, 1998.

—. 2000. El Proceso Unificado de desarrollo de Software. Madrid : Addison Wesley, 2000.

Microsoft Corporation. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165742%28SQL.90%29.aspx.

—. 2008. Microsoft Developer Network. [En línea] noviembre de 2008. http://msdn.microsoft.com/es-es/library/ms165720%28SQL.90%29.aspx.

Glosario de Términos

Base de datos: Conjunto no redundante de información almacenada en memoria organizada independientemente de su utilización y su implementación en máquinas accesibles en tiempo real, y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo.

Framework: Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías de código y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Licencia GNU/GPL: Licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Metodología: Métodos de investigación que se siguen para alcanzar una gama de objetivos en una ciencia.

Multiplataforma: puede ser ejecutado en diferentes Sistemas Operativos (SO).

Publicador: Es una instancia de base de datos que permite que los datos estén disponibles para otras ubicaciones a través de la replicación. El publicador puede tener una o más publicaciones, cada una de las cuales representa un conjunto de objetos y datos relacionados lógicamente para replicar.

RIA: Aplicaciones Enriquecidas de Internet (del inglés Rich Internet Applications) son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.

Software Educativo: Software destinando a la enseñanza y el auto aprendizaje y que además permite el desarrollo de ciertas habilidades cognitivas.

Suscriptores: Es una instancia de base de datos que se encarga de recibir los datos replicados, de los publicadores y publicaciones. Además, posibilita devolver los datos con modificaciones al publicador, o volver a publicarlos en otro suscriptor.