Framework para el Desarrollo Ágil de Aplicaciones Web

Lisandro Delía [1], Germán Cáseres [2], Hugo Ramón [3], Pablo Thomas [4], Rodolfo Bertone [5]

Instituto de Investigación en Informática LIDI (III-LIDI) [6]
Facultad de Informática - UNLP

Abstract

Any system interacting with a data base requires modules capable of operating data stored in it. Its development times generally range from a 50 to 60 % of the time used for the application life cycle. The present paper describes the architecture and characteristics of a Framework called PHP4DB to agilely generate Web Systems. Its main objectives are to significantly reduce the working time, minimize errors and tuning, as well as respect a homogeneous interface between each module. These characteristics allow the development team to focus and make emphasis on the specific tasks of the application domain. In order to see its advantages more clearly, this paper presents some of the projects in which this Framework has been used and the corresponding analysis of the results obtained.

Key Words

Software Engineering, Agile Developments, WEB Applications.

Resumen

Todo sistema que interactúe con una base de datos requiere de módulos que sean capaces de operar los datos almacenados en ella. Sus tiempos de desarrollo generalmente oscilan entre un 50 y 60% del tiempo utilizado para el ciclo de vida de la aplicación. El presente trabajo describe la arquitectura y características de un Framework para la generación ágil de Aplicaciones Web, denominado PHP4DB. Sus objetivos principales son reducir drásticamente el tiempo de trabajo, minimizar los errores y la puesta a punto, como así también respetar una interfaz homogénea entre cada uno de los módulos. Estas características permiten al equipo de desarrollo concentrarse y poner énfasis en las tareas especificas del dominio de la aplicación. Para una mejor apreciación de sus ventajas, se presentan algunos de los proyectos donde se utilizó el Framework con el análisis respectivo de los resultados obtenidos.

Palabras Claves

Ingeniería de Software, Desarrollos Ágiles, Aplicaciones WEB.

---

[1] Becario III-LIDI UNLP - Fac. de Informática, ldelia@lidi.info.unlp.edu.ar

[2] Becario III-LIDI UNLP - Fac. de Informática, gcaseres@lidi.info.unlp.edu.ar

[3] Profesor Adjunto DE, UNLP - Fac. de Informática, hramon@lidi.info.unlp.edu.ar

[4] Profesor Adjunto DE, UNLP - Fac. de Informática, pthomas@lidi.info.unlp.edu.ar

[5] Profesor Adjunto DE, UNLP - Fac. de Informática, rbertone@lidi.info.unlp.edu.ar

[6] III-LIDI UNLP - Fac. Informática, calle 50 y 115, La Plata (1900), Buenos Aires, Argentina, Tel/Fax +54 221 4337707 http://www.lidi.info.unlp.edu.ar

1 Introducción

El fundamento de la IS (Ingeniería de Software) es tener un proceso establecido para el desarrollo de Sistemas de Software. Un proceso define un marco de trabajo para un conjunto de áreas clave, conocidas como KPA (Key Process Area), que se debe establecer para la entrega efectiva de un producto de software. [1] [2]

Para poder generar una capa de proceso estable y de calidad se debe partir de una especificación de requerimientos consistente del problema [17]. Estos requerimientos deben; (1) representar y entender el dominio de información de un problema, (2) definir las funciones que realizará el software, (3) representar el comportamiento deseado del software (como consecuencia de eventos externos), (4) dividir los modelos que representan información, función y comportamiento de manera que se descubran los detalles por capas jerárquicas. [3]

Para mantener dentro de la planificación el desarrollo de un SI (Sistema de Información) se puede minimizar, entre otros, el tiempo necesario para realizar la codificación. Si bien este tiempo es mínimo dentro del ciclo de desarrollo de un sistema, las tareas repetitivas no específicas del dominio de aplicación ocupan generalmente entre un 50 y un 60% del tiempo total asignado. Además, la puesta a punto y depuración de la funcionalidad, y la generación de interfaz de usuario resulta en valores temporales que no pueden considerarse despreciables. [4] [5]

Una vez estabilizados los requerimientos es posible desarrollar un modelo de datos completo, ágil y dinámico que los represente de la forma más adecuada. [6]

A partir de aquí, un sistema que implante la funcionalidad requerida necesitará de módulos básicos que administren la información contenida en la BD (Base de Datos). El desarrollo y mantenimiento de cada uno de estos módulos requiere dedicar tiempo a tareas rutinarias manteniendo la consistencia de interfaz y correctitud [7].

El equipo de desarrollo debe concentrarse en la programación de las funcionalidades mínimas (utilizando un lenguaje de programación), actualizar la BD (generalmente con otro lenguaje específico), armar los formularios de carga de datos, grillas, combinando componentes visuales, entre otras actividades [8]. Además, un aspecto importante de toda aplicación, en particular cuando se trata de un SI, es la coherencia en el desarrollo de la interfaz, debiendo presentar la información e interactuar con el usuario en forma homogénea y consistente.

Estos objetivos resultan normalmente tediosos para los desarrolladores. Es aquí donde resultan particularmente útiles los lenguajes de programación de cuarta generación (4GL) (como por ejemplo Clarion [9]) y las herramientas CASE de generación automática de código.

Un generador de código automático es una herramienta que deriva, a partir de determinados patrones, el código fuente de una aplicación. El uso de estas herramientas reduce el tiempo necesario para el desarrollo del software, minimiza los errores, reduciendo consecuentemente los tiempos de depuración y puesta a punto. Los 4GL constan de procedimientos que generan el código fuente en función de lo expresado en el diseño de la aplicación o modelo de datos. Para esto, el usuario especifica la funcionalidad del programa, o parte del mismo, y la herramienta determina cómo realizar dicha tarea. [1]

Sin embargo, la generación automática de código en muchos casos no es suficiente, dado que las aplicaciones obtenidas a partir de un 4GL cuentan con un nivel de *estaticidad* tal, que cualquier cambio en el modelo de datos produce un alto impacto en el mantenimiento.

El contexto de trabajo, básicamente cercano a metodologías ágiles tipo XP[18], lleva a un entorno donde el modelo de datos sufre dinámicamente conversiones y/o adaptaciones. Disponer de una herramienta CASE estática no resuelve efectivamente el problema.

Es necesario, entonces, pensar en el desarrollo de un CASE que pueda adaptar dinámicamente una aplicación a los cambios continuos producidos sobre el modelo de datos, manteniendo la regularidad en las interfaces de usuarios producidas.

2 Presentación del problema

Dentro del contexto de trabajo de los autores, el Instituto de Investigación de Informática III-LIDI, se han desarrollado proyectos con características similares, que requieren de un alto porcentaje de tablas con las operaciones clásicas tales como listados, filtros, reportes, ABM (Altas, Bajas y Modificaciones) de datos. A cada tabla junto a sus operaciones clásicas asociadas, de aquí en mas las denominaremos *repositorios*.

Estos proyectos son, básicamente, Sistemas Web [10], dada la necesidad de acceder a la información desde lugares físicamente remotos y estan desarrollados con herramientas open-source, en ambientes LAMP (Linux + Apache + MySQL + PHP) [11] e interactuan con BD's integradas por información heterogénea. Esto último provoca un gran esfuerzo para generar la interfaz de cada *repositorio*. Por lo tanto, es fundamental contar con una capa de software genérica que automatice estas tareas.

La complejidad en desarrollar esta capa, depende del tipo de aplicación en cuestión. En aplicaciones tipo RAD (Rapid Application Development) tales como Delphi, PowerBuilder, VisualBasic, es posible parametrizar componentes para lograr *repositorios* con poco esfuerzo [12].

En aplicaciones Web, basadas en tecnología cliente-servidor [13], la solución se presenta un tanto más compleja. Esto es, se debe resolver el proceso tanto del lado del cliente (mediante JavaScript, Java Applets), como del lado del servidor (mediante PHP, ASP, JSP, etc), en conjunto con un modo de mostrar la información (HTML, XML + CSS).

El desarrollo propuesto, un Framework denominado PHP4DB, se diseñó para resolver los problemas presentados. PHP4DB es una herramienta orientada a objetos desarrollada íntegramente en PHP, con el objetivo de generalizar lo más posible la capa de software que permita automatizar tareas rutinarias de codificación en un ambiente LAMP. En los siguientes apartados se presenta PHP4DB y se analiza su comportamiento.

3 Arquitectura y descripción

3.1 Funcionalidad

PHP4DB fue un desarrollo netamente evolutivo: se plantearon una serie de objetivos básicos los cuales, una vez alcanzados, permitieron "evolucionar" tanto en complejidad como en completitud. En la versión actual es posible realizar las siguientes tareas:

- Visualizar los datos de un *repositorio* mediante una grilla paginada.
- Filtrar dinámicamente los datos del *repositorio* según atributos definidos para tal fin
- Obtener una vista rápida de una fila de la grilla
- ABM de datos mediante un formulario pre-establecido
- Generar un reporte formato PDF de todos los datos que se visualiza en la grilla o de un dato en particular

- Relacionar un dato en particular con otra funcionalidad externa al *repositorio*
- Auditar en formato XML cada operación que hace el usuario en el *repositorio*

Para esto, PHP4DB se comunica dinámicamente con la BD del problema a resolver para recuperar o actualizar la información allí contenida.

El desarrollo de esta herramienta estuvo ideado, desde un principio, para llevarse a cabo en productos de licencia libre. Por este motivo, el DBMS utilizado fue MySQL. En versiones posteriores, se observó que las limitaciones implantadas a partir del uso de un DBMS particular no eran adecuadas, y por este motivo el Framework evolucionó para abstraerse del motor de BD particular. Para lograr la abstracción requerida se utilizó la librería DB de PEAR (PHP Extension and Application Repository) [14].

La Figura 1 presenta la ejecución de un *repositorio* como parte de un sistema. Inicialmente, como acceso principal a dicho *repositorio*, se muestra una grilla paginada de datos con un formulario de filtro asociado. A partir de aquí, es posible acceder a todas las demás funcionalidades del Framework. En la grilla se describen los datos listados, los cuales pueden ser derivados, como se dijo anteriormente, a un reporte en formato PDF.

A los datos mostrados se le pueden aplicar acciones, algunas básicas, como la modificación o baja, y otras específicas al *repositorio* vinculadas con el comportamiento definido por los requerimientos del sistema. Toda la funcionalidad específica se presenta asociada a cada fila de la grilla y se aplica sobre ella. La presentación puede ser mediante una lista desplegable o una barra de herramientas.

Además, como funcionalidad básica, es posible insertar nuevos elementos. La figura 2 presenta un ejemplo de este formulario, mientras que la figura 3 presenta la vista de un registro particular ante una consulta.

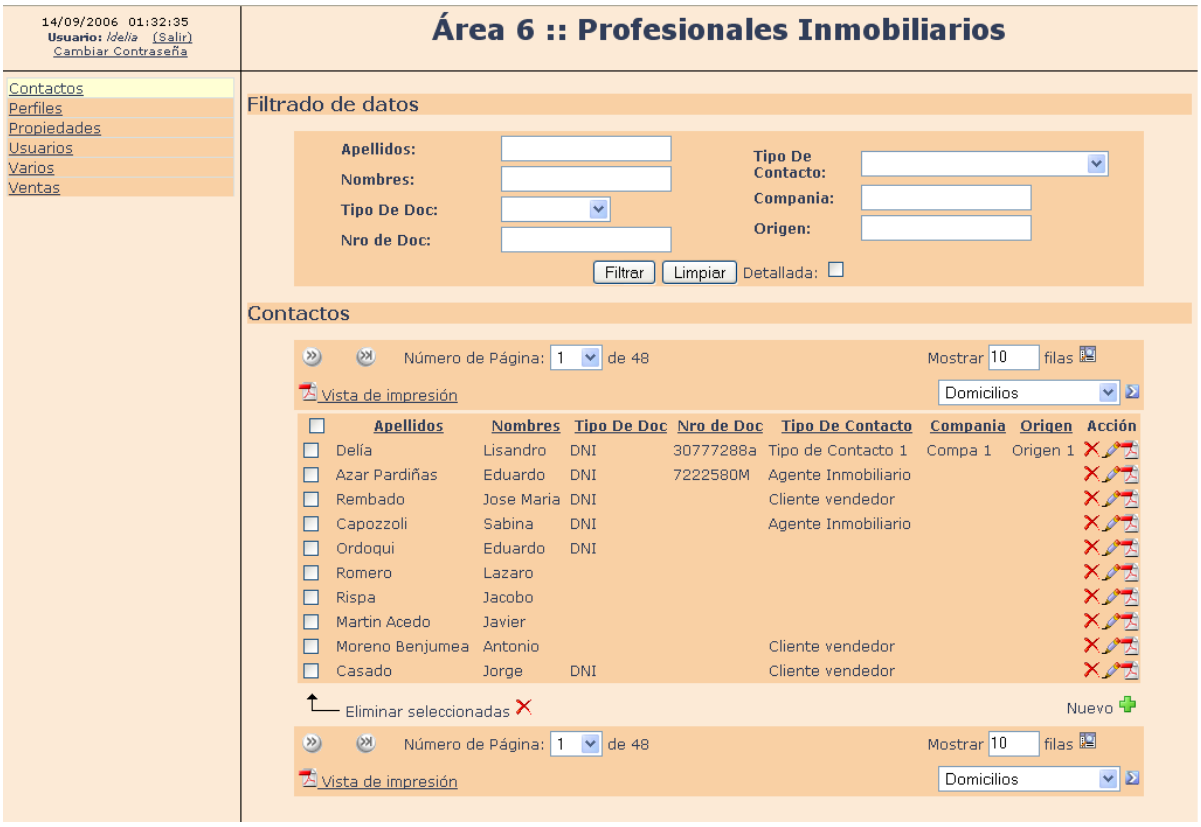

Figura 1: Ejemplo de repositorio

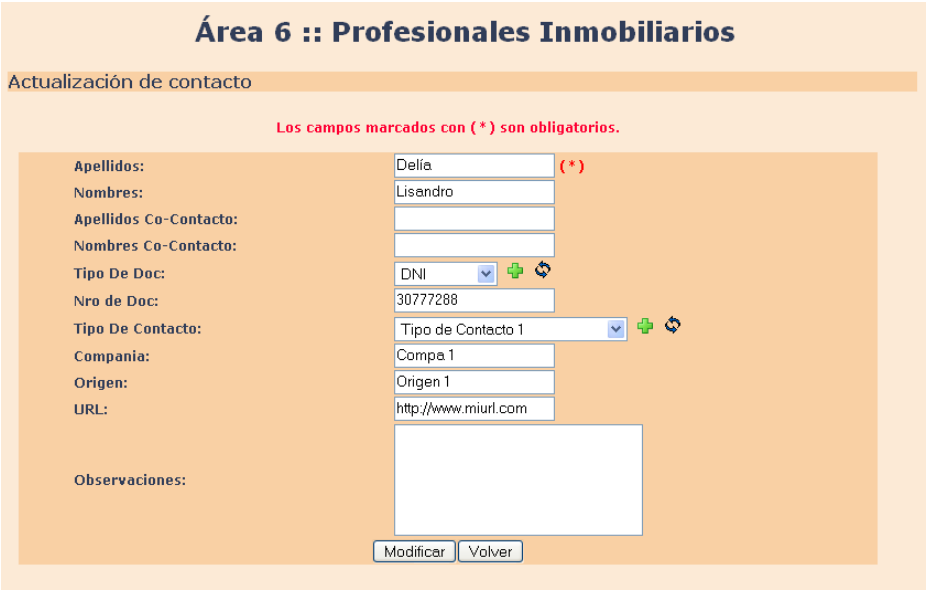

Figura 2: Formulario Alta/Modificación

Figura 3: Vista rápida de un registro

3.2 Estructura

PHP4DB esta diseñado como un núcleo centralizado encargado de crear toda la funcionalidad mencionada en la sección 3.1. Para ello, cada *repositorio* definido utiliza la funcionalidad del núcleo para la presentación de la información.

El núcleo necesita ser configurado para cada proyecto específicamente, a fin de responder a cada aplicación desarrollada en el Instituto. Esto da lugar al ProyectDataScript (PDS), archivo de configuración de la aplicación, el cual cuenta con la descripción de la BD del proyecto (servidor, DBMS, usuario, clave), el estilo que tendrán las interfaces (CSS, Iconos), así como información adicional necesaria.

Mediante el PDS el núcleo tiene la configuración del sistema y la información en común de todos los *repositorio*s.

Por su parte, se definió un archivo denominado FDS (FormDataScript), encargado de brindar toda la información específica de cada *repositorio* y de la tabla que hace referencia. Con este descriptor de datos, PHP4DB puede brindar toda la funcionalidad para el *repositorio* asociado. Los FDS describen, entre otras cosas, la siguiente información:

- Títulos para cada operación del *repositorio*
- Nombre de la tabla de la BD, que hace referencia el *repositorio*
- Campos de la tabla de la BD, donde para cada uno de ellos se tiene:
    - Nombre del campo
    - Label significativo a mostrar
    - Visibilidad en grilla/reportes
    - Visibilidad en el filtro
    - Tipo del campo
- Permisos para cada función del *repositorio*, con el objetivo de habilitar/deshabilitar funciones de acuerdo al perfil del usuario

La Figura 4 presenta la estructura del Framework PHP4DB. Cada FDS, contiene la información de un *repositorio* en particular. Cuando se invoca a uno de estos *repositorio*s, el FDS envía toda su información al núcleo PHP4DB, quien lleva a cabo alguna de sus funciones, recuperando la información de la BD.

Cabe destacar que cada nueva funcionalidad que se le incorpore al núcleo del Framework, como puede ser la exportación de datos a un formato particular, es obtenida por cada *repositorio* sin realizar modificación alguna. Lo mismo ocurre con el mantenimiento de cada funcionalidad o corrección de errores, todo *repositorio* recibirá estos beneficios, sin necesidad de alterar su contenido.

Es interesante destacar que cuando se necesita crear un nuevo *repositorio*, no es necesario agregar programación alguna (ya sea código PHP, HTML o SQL). Solo se debe crear el FDS asociado al *repositorio*.

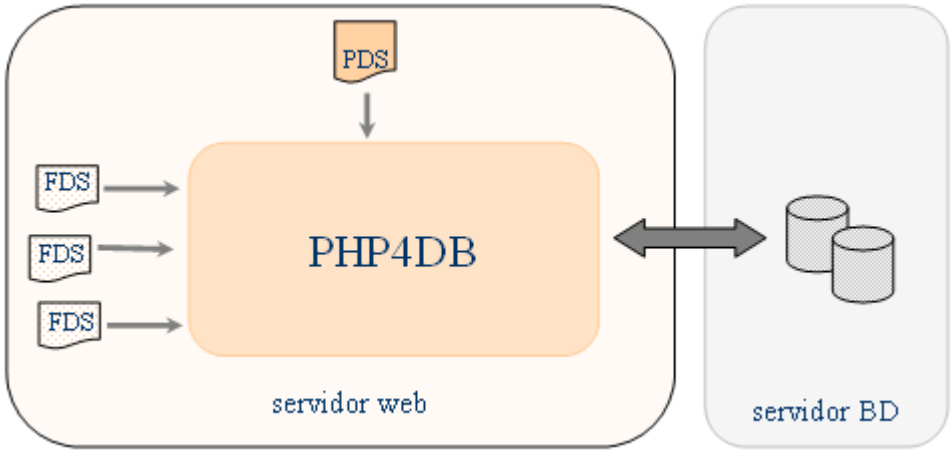

Figura 4: Estructura del Framework PHP4DB

3.3 Asistente para la creación de los FDS

Si bien los desarrolladores pueden generar/modificar los FDS manualmente, se desarrolló una herramienta que automatiza esta tarea, ahorrando tiempo y evitando errores debido a lo engorroso que puede resultar la opción manual.

Esta herramienta es una aplicación de escritorio, denominada PHP4DB Assistant, que en pocos pasos permite crear los FDS para cada *repositorio*. Teniendo en cuenta que PHP4DB es un Framework pensado para que coexista con varios proyectos simultáneamente, la primera tarea consiste en utilizar el PDS asociado al proyecto donde se desea agregar el nuevo *repositorio*. PHP4DB Assistant visualiza las tablas de la BD del proyecto, donde el usuario escoge la tabla que será referenciada por el nuevo *repositorio*.

Seleccionada la tabla, automáticamente se despliega la información de sus campos y resta al usuario configurar ciertos detalles tales como: (1) los *labels* que tendrán los campos, (2) el tipo de dato base de cada campo (texto, número, clave foránea, etc), (3) los títulos de las diferentes acciones, (4) las funcionalidades que estarán activas para el *repositorio*, etc. La figura 5 presenta un resumen de estos detalles.

Una vez configurado, el archivo FDS se almacena en un servidor web, y se está en condiciones de presentar el *repositorio* desde cualquier navegador.

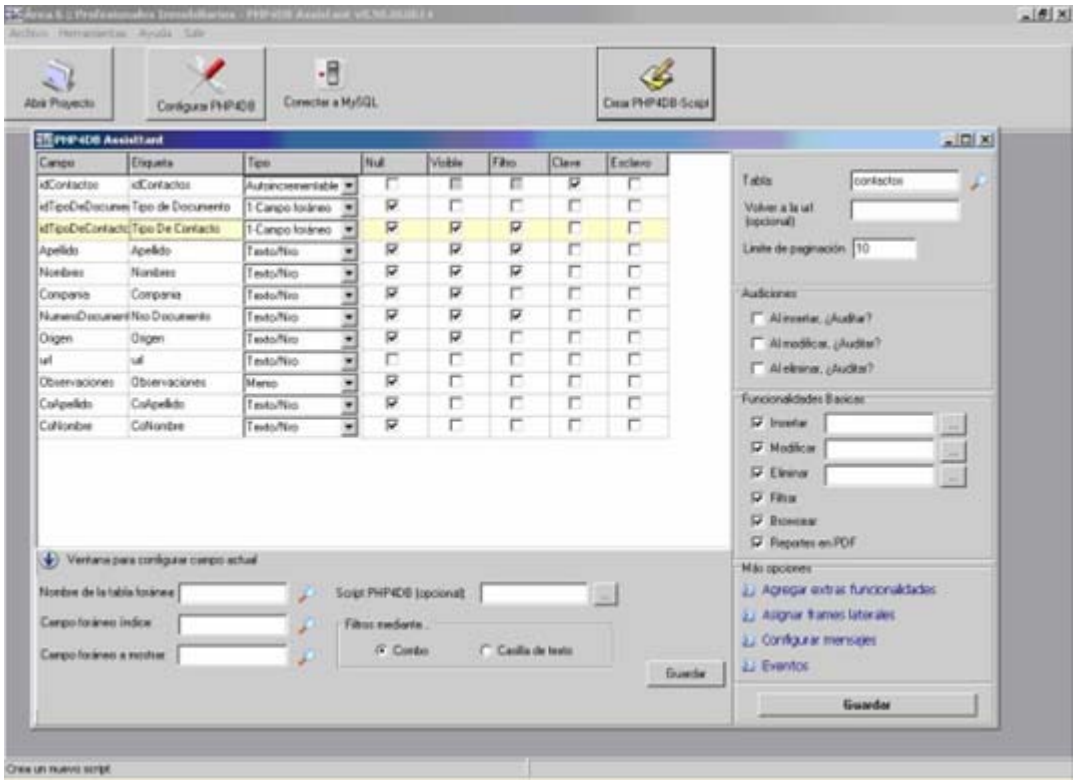

Figura 5: Creación de un repositorio con PHP4DB Assistant

3.4 Características de los repositorios

3.4.1 Tipos de campos

En los archivos FDS se describe la información sobre los campos a presentar en los formularios, grillas, filtros y reportes. Estos campos varían unos a otros. Por ejemplo, la manera de presentación

de una fecha en un formulario, no debería ser igual a mostrar un texto. Es por esto que PHP4DB identifica a cada campo con un tipo en particular.
Actualmente el Framework puede trabajar con los siguientes tipos de campos:

- Textos cortos y extensos
- Números enteros y flotantes
- Fechas
- Imágenes
- Campos foráneos
- Boléanos

Al tener un diseño orientado a objetos, agregar un nuevo tipo de dato al Framework no es una tarea costosa. De esta forma es posible extender el Framework fácilmente ampliando el alcance de los *repositorios*.

3.4.2 Eventos

Si bien todas las funcionalidades básicas de un *repositorio* se pueden resolver automáticamente, existen casos en que algunas de ellas necesitan comportarse de otra forma.

PHP4DB brinda la posibilidad que los *repositorios* tengan orientación a eventos, permitiendo ejecutar acciones en determinados momentos de la ejecución. Para llevarlo a cabo, PHP4DB verifica si el DFS que está siendo ejecutado tiene definido el evento correspondiente al punto de ejecución. De estar definido PHP4DB invoca al evento; caso contrario sigue trabajando normalmente. Eventos como *before_execute_insert()* o *after_execute_insert()* por citar ejemplos, aumentan el nivel de dinamismo del Framework

3.4.3 Relación con otras funcionalidades

Se mencionó que el marco ideal de trabajo seria que el equipo de desarrollo dedique el tiempo en los módulos que requieren una programación específica, sin perderlo con el desarrollo de los *repositorios*.

En caso de contar con módulos específicos, existe la necesidad de poder relacionarlo con los *repositorios*. La Figura 6 muestra como se puede relacionar un registro en particular con otras funciones del sistema. Mediante una lista desplegable, o simples iconos en cada fila de la grilla, se le puede aplicar una acción a un registro seleccionado. Estas acciones implican llamadas a otros módulos que han sido desarrollados específicamente, o bien, otros repositorios creados con PHP4DB Assistant.

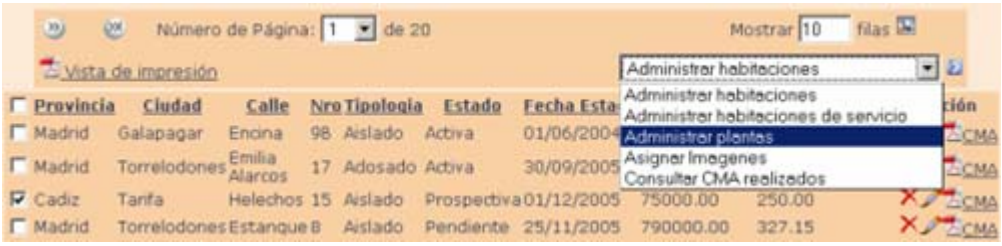

Figura 6: Relación con otras funciones

4 Resultados obtenidos

Como se mencionó previamente, el Instituto ha desarrollado un número importante de sistemas con transferencia. Por lo tanto, disponer de una herramienta como PHP4DB ha minimizado el tiempo de codificación, entre otras ventajas. Las próximas líneas describen el alcance de algunos de estos proyectos.

4.1 Area 6 Profesionales Inmobiliarios

Sistema CRM (Customer Relationship Management) Multi-Inmobiliario orientado a la web, actualmente en producción en España. Su objetivo es administrar las actividades inmobiliarias inherentes al ciclo de vida de una propiedad, desde su ingreso al mercado hasta la venta. Además, brinda el servicio de estimar objetivamente el precio de venta que debería tener una propiedad, basándose en otras propiedades con características similares. Este es el proceso central de esta aplicación y se denomina CMA (Comparative Market Analysis)

4.2 Software de Administración Integral Hospitales

El objetivo buscado bajo el proyecto denominado SAIH-LIDI consiste en la informatización total de hospitales tanto de autogestión como aquellos que dependen de un presupuesto preasignado.

Esto se logra integrando las áreas de atención por consultorios externos, internaciones y de servicios externos (que en algunos casos puede consistir en derivación de pacientes); generando una Historia Clínica básica para cada paciente. Además permite administrar el cobro a Obras Sociales. Conjuntamente a lo mencionado se integran actividades de acción social del Municipio. Todo esto permite generar un marco informativo de calidad hacia el paciente y consultas externas, resolviendo la administración interna del hospital.

4.3 DPIC (Dirección Provincial de Informática y Comunicaciones de la Provincia de Buenos Aires)

La Provincia de Buenos Aires está gestionando un proceso de licitación pública para proveer el "Servicio de Transmisión de Datos y Canales de Ordenes" para la Red Única Provincial de Comunicación de Datos. La RedPIBA (Red provincial de grupos de Investigación y desarrollo en áreas de Ciencia de la Computación e Informática) tuvo a su cargo la definición del manual de procedimientos para unificar los criterios y el mecanismo de obtener la aceptación de nodos, definir el marco para la capacitación de los mismos en las tareas especificas, y hacer control del seguimiento del proyecto [15].

Esta red provincial consta de 1300 nodos, sobre cada uno de los cuales se realiza auditoria de obra y el pasaje a producción a la nueva red. Para esto, se dividió la provincia en 6 zonas, cada una con cabecera de Universidades pertenecientes a la RedPIBA**:** UNLP (Universidad Nacional de La Plata), UNLM (Universidad Nacional de La Matanza), UTN (Universidad Tecnológica Nacional), UNLu (Universidad Nacional de Luján), UNC (Universidad Nacional del Centro), UNS (Universidad Nacional del Sur). Cada zona posee un coordinador y dos equipos de técnicos especialistas. Además existe un equipo central de coordinación.

Las tareas de coordinación para la certificación se realizaron mediante una Aplicación WEB desarrollada con PHP4DB.

5 Conclusiones

La utilización del Framework en los proyectos fue esencial ya que logró automatizar un gran porcentaje de CU (casos de uso).

En el proyecto Area6, de 50 CU solo 1 fue programado específicamente (CMA) [16], siendo la función más compleja que requiere estadísticas y formas de uso particulares. SAIH-LIDI, por su parte, cuenta con 30 CU implementados hasta la fecha. De estos 30 CU, sólo 12 (Turnos y Farmacia) fueron implementados específicamente. DPIC cuenta con 30 CU y solo 4 recibieron una programación particular.

Es claramente visible el beneficio obtenido con PHP4DB. El tiempo de desarrollo se redujo drásticamente con la utilización de la herramienta, con la consecuente satisfacción del usuario dada la temprana disponibilidad de los productos requeridos. Además la centralización provista por PHP4DB ha permitido lograr interfaces homogéneas, facilitando el mantenimiento posterior.

Por último, es de notar la expectativa que genera disponer de este framework para aplicaciones futuras que requieran ser orientadas a la web.

6 Trabajo futuro

Si bien actualmente se dispone con una madurez en el Framework que hace posible la operabilidad con cualquier tabla de cualquier motor de BD, la vorágine tecnológica conduce a tener en mente otros dominios de información tales como archivos XML o vistas abarcando información de varias tablas. Con estas extensiones se ampliará la usabilidad del Framework permitiendo que este logre convivir con una mayor cantidad de Sistemas .

La incorporación de nuevos tipos de datos para PHP4DB también será beneficioso. Son, ejemplos de estos beneficios, la posibilidad de almacenar cualquier tipo de archivo en las tablas (.doc,.mp3,.mpeg), o de disponer campos cuyos valores sean definidos por el usuario, entre otros. Cualquiera de estos, lograrán incrementar aún más la usabilidad del producto.

Otro tarea no menos importante es la de llevar el asistente mencionado en la sección 3.3, a una plataforma WEB, con el fin de poder crear/modificar los *repositorio*s desde cualquier lugar.

7 Referencias

[1] Ingeniería de Software. Un enfoque práctico. Roger Pressman. Mc Graw Hill. 1998.

[2] Capability Maturity Model for Software, M. Paulk, Software Engineering Institute, Cargenie Mellon University. 1993

[3] Principles of Software Development. A. Davis, Mc Graw Hill. 1995.

[4] Generación Automática de Código a partir del modelo de datos. M. Walsamakis, M. Mansutti, R. Bertone, R. Champredonde . CACIC2004. La Matanza. Octubre 2004

[5] Ingeniería de Software. 6ta Edición. Ian Sommerville. Addison Wesley 2002.

[6] Diseño conceptual de Bases de Datos. Batini, Navathe Cieri. Addison Wesley. 1990

[7] Developing User Interfaces - Dan R. Olsen, Morgan Kaufmann, 1998

[8] Introduction to the Team Software Process - Watts S Humphrey - Addison-Wesley Professional 1999

[9] Clarion 4. Manual de Referencia. Top Speed.

[10] Rodriguez de la Puente Santiago, **Programación de aplicaciones WEB**, Paraninfo, 2003.

[11] M. Torchiano, M. Morisio. Overlooked Aspects of COTS-Bases Development, IEEE Software, 2004.

[12] Software Engineering With Reusable Components - Johannes Sametinger - Springer-2001

[13] Building Application Servers - Rick Leander - Cambridge University Press – 2000

[14] http://pear.php.net/

[15] www.dpic.sg.gba.gov.ar

[16] http://homebuying.about.com/library/glossary/bldef4.htm

[17] Loucopoulos, P., Karakostas, V., *System Requirements Engineering*, McGraw-Hill, 1995, London.

[18] Beck K., *Una explicación de la Programación Extrema*, Addison Wesley, 2002.