dotNET4DB: Un Framework para el Desarrollo Ágil de Aplicaciones Web bajo .NET

Germán Cáseres[1], Nicolas Luna[2], Pablo Thomas[3], Rodolfo Bertone[4], Marcos Boracchia[5], Lisandro Delia[6], Hugo Ramón[7]
Instituto de Investigación en Informática LIDI (III-LIDI)[8]
Facultad de Informática - UNLP

Abstract

The agile development of applications requires a series of elements and organization guidelines. The human factor and its predisposition for team work is one of these elements, nevertheless, having automatized tools that facilitate the process of software development is another important requirement as well. Regarding this purpose, the team work has developed several tools that help the development of applications using PHP as the programming language, with classic BD motors such as MySQL or MSSQL. Accordingly, these results have been published in several research papers.

However, Web technology makes it necessary to have the same type of tool in order to develop using .NET technology. In this regard, dotNET4DB is introduced, a working framework aimed at the support for applications development using .NET technology. This tool provides the possibility of automatically generating the basic functionalities (Additions, Deletions, Modifications and Reports) of a system of computing management as well as facilitating and assisting the development of other specific functional requirements.

Keywords: Agile Development, WEB Aplications, CASE Tools, .NET

Resumen

El desarrollo ágil de aplicaciones requiere disponer de una serie de elementos y pautas en la organización. El factor humano y su predisposición para el trabajo en equipo es uno de ellos, pero disponer además de herramientas automatizadas que agilicen el proceso de construcción del software, es otro requerimiento de vital importancia. En este sentido, el grupo de trabajo ha desarrollado herramientas que asisten en el desarrollo de aplicaciones, utilizando PHP como lenguajes de programación, con motores de BD clásicos como puede ser MySQL o MSSQL. Al respecto, los resultados han sido presentados en diversas publicaciones.

Sin embargo, la tecnología Web hace necesario contar con este mismo tipo de herramientas para desarrollar bajo tecnología .NET. En este sentido, se presenta dotNET4DB, un framework de trabajo para la asistencia en el desarrollo de aplicaciones bajo tecnología .NET. Esta herramienta permite generar de forma automática la funcionalidad elemental (Altas, Bajas, Modificaciones y Listados) de un sistema de gestión informática, así como facilitar y asistir el desarrollo de otros requerimientos funcionales específicos.

Palabras Claves: Desarrollos Ágiles, Aplicaciones WEB, Herramientas CASE, .NET

---

[1] {gcaseres@lidi.info.unlp.edu.ar} Becario III-LIDI. Fac. de Informática. UNLP
[2] {nluna@lidi.info.unlp.edu.ar} Becario III-LIDI UNLP – Fac. de Informática. UNLP
[3] {pthomas@lidi.info.unlp.edu.ar} Profesor Adjunto DE . Fac. de Informática. UNLP
[4] {rbertone@lidi.info.unlp.edu.ar} Profesor Titular DE . Fac. de Informática. UNLP
[5] {marcosb@lidi.info.unlp.edu.ar} Profesor Adjunto DE. Fac. de Informática. UNLP
[6] {ldelia@lidi.info.unlp.edu.ar} Becario III-LIDI. Fac. de Informática. UNLP
[7] {hramon@lidi.info.unlp.edu.ar} Profesor Adjunto DE. Fac. de Informática. UNLP
[8] Instituto de Investigación en Informática. Fac. de Informática. UNLP. La Plata. http://www.lidi.info.unlp.edu.ar

1. Introducción

Los métodos ágiles de desarrollo de software nacieron como una variante de las metodologías de desarrollo tradicionales, dado que es prácticamente imposible evitar los cambios en los requerimientos de los clientes/usuarios, en la tecnología a usar, en la disponibilidad del personal, en el conocimiento disponible y la experiencia y talento en la organización. Así, en el desarrollo de software de pequeña o mediana envergadura muchas veces es deseable avanzar en forma gradual, iterativa, e interactiva con el usuario, utilizando un ciclo de vida de desarrollo de software incremental [1].

Entonces, como no es factible evitar los cambios en las necesidades del usuario, la solución propuesta consiste en evitar esa "rigidez" de los grandes planes, grandes diseños y procesos y reemplazarlos por otro tipo de ciclo de vida, los métodos ágiles, cuya adaptabilidad dan respuesta inmediata y eficaz a los cambios inevitables [12].

Los métodos ágiles necesitan de la participación cotidiana del usuario final en los grupos de trabajo. El objetivo es satisfacer al cliente a través de la entrega continua de software. Para ello, los grupos de trabajo tienen un amplio poder de decisión, lo que les permite tener un alto grado de confianza en sus resultados, teniendo proyectos construidos por personal altamente motivado [3].

El propósito de los métodos ágiles tienen que ver con un cambio importante en el razonamiento de los sistemas [2]. Estos cambios estan orientados a:

- Trabajo individual con alta interacción sobre procesos y herramientas.
- Software que funcione rápidamente como algo más prioritario que una documentación exhaustiva.
- Colaboración total del cliente y participación abierta del mismo, como elemento más importante del contrato que rige del desarrollo.
- Alta respuesta al cambio y adaptación al cambio, más que seguir un plan específico

Los métodos ágiles no sacrifican la calidad en pos de velocidad u otros intereses; buscan la excelencia técnica aun después de cada entrega parcial. La estrategia de estos métodos se fundamenta en iteraciones rápidas, entregando parcialmente software que funciona, incorporando los aspectos faltantes en iteraciones sucesivas posteriores[4] [5].

Así, utilizando métodos ágiles es posible alcanzar niveles de calidad comparables con los obtenidos con CMMI nivel 3. Se debe tener en cuenta que para el tipo de desarrollo resoluble con esta metodología (considerado de pequeña y mediana envergadura), un nivel 3 de CMMI resulta por demás adecuado [8].

2. El problema original

Ante la necesidad de implementar un sistema WEB bajo la plataforma .NET donde gran parte de los requerimientos funcionales poseen características similares, tanto en su aspecto visual como en su funcionalidad, resulta útil estudiar la manera de minimizar los tiempos de desarrollo y optimizar las técnicas utilizadas.

Trabajar sobre cada requerimiento funcional como un problema diferente, tratándolo aisladamente y sin reutilizar el análisis ya realizado, no resulta en una buena opción, básicamente por dos motivos:

- Excesivo tiempo de análisis e implementación.
- Heterogeneidad visual y funcional entre requerimientos similares.

Para mantener la homogeneidad a nivel del proyecto y optimizar el reúso, resulta una buena opción realizar un análisis en conjunto los requerimientos. Sin embargo, la evolución de cada proyecto se convierte en un trabajo individual y cualquier mejora aplicable a otro proyecto deberá ser replicada y adaptada en forma manual.

Para salvar este inconveniente, es necesario hacer un análisis mayor, que permita al usuario programador abstraerse aún más del diseño de interfaz y comportamiento, de manera de mantener el enfoque centralizado solo en aquellos casos particulares donde no sea posible aplicar una solución genérica. Este análisis debe realizarse no solo a nivel de proyecto, sino a nivel de la arquitectura que engloba a cada proyecto, de manera de proveer una solución que no solo se limite a una aplicación en particular.

Una de las arquitecturas mas utilizadas en sistemas WEB es la arquitectura de tres capas. Este estilo de programación se corresponde con la separación física que se presenta en los sistemas de tipo Cliente-Servidor siendo dicha correspondencia la siguiente:

- Cliente: Capa de presentación
- Servidor WEB: Capa de lógica de negocios
- Servidor de datos: Capa de acceso a datos

Este tipo de arquitectura permite al mismo tiempo llevar a cabo la implementación de cada capa de forma separada, optimizando los tiempos de desarrollo y el futuro costo de mantenimiento.

A partir de la experiencia de los autores en su contexto de trabajo, se ha identificado que la mayor cantidad de similitudes entre los diferentes requerimientos funcionales se encuentran en dos puntos completamente distinguibles: La interfaz de usuario y la manera en que se accede a los datos subyacentes.

Ambos puntos, se encuentran directamente relacionados con dos de las capas que conforman la arquitectura de tres capas: la capa de presentación y la capa de acceso a datos.

Mantener la homogeneidad en la implementación de estos dos puntos es importante en el momento de realizar mantenimiento sobre sistemas en producción, al punto que es favorable depender de un núcleo actualizable que se encargue de brindar funcionalidad a cada sistema. De esta manera, la tarea de propagar cualquier cambio positivo a cada uno de los proyectos en producción se facilita notablemente.

Este núcleo, además de ser actualizable, debe proveer las funcionalidades comunes generalmente presentes en diferentes aplicaciones, y al mismo tiempo, ser lo suficientemente abstracto para poder cubrir la mayoría de los dominios posibles.

3. Arquitectura y descripción

3.1 Funcionalidad

dotNET4DB fue generado a partir de un desarrollo netamente evolutivo: se plantearon una serie de objetivos básicos los cuales, una vez alcanzados, permitieron evolucionar tanto en complejidad como en completitud.

dotNET4DB es una plataforma para el desarrollo ágil de aplicaciones WEB implementada sobre ASP.NET en lenguaje C#, cuyo objetivo es presentar al usuario programador un conjunto de clases configurables capaces de resolver los problemas típicos que se presentan en el momento de implementar una aplicación WEB de tres capas.

Se le atribuye el nombre de framework o plataforma, dado que, si bien las clases que lo conforman son capaces de interactuar entre sí, esta plataforma no tiene funcionalidad propia, sino que necesita de otras clases con comportamiento específico para un proyecto, y constituir de esa manera un sistema ad-hoc a un propósito particular.

Una ventaja que ofrece .NET consiste en que las nuevas clases definidas, pueden ser implementadas en otros lenguajes alternativos (Visual Basic, Java, entre otros).

3.2 Estructura

dotNET4DB permite abstraer al usuario de la necesidad de implementar las operaciones básicas de un sistema con características de un ECM (*Enterprise Content Management*). Para ello, cuenta con módulos y clases destinadas a solucionar el problema en cada una de las capas:

- Capa de presentación (Interfaz de usuario)
- Capa de lógica de negocios
- Capa de acceso a datos

Aprovechando la naturaleza orientada a objetos de la plataforma .NET, este framework fue desarrollado utilizando patrones de diseño que permiten una correcta separación entre cada una de las capas, de manera tal de optimizar el trabajo en equipo y lograr una modularización eficiente del sistema en general. Para tal fin, se tomaron como base diferentes librerías y plantillas, entre otros elementos, que permitieron incorporar rápidamente a una aplicación WEB, las prácticas y patrones de diseño comunes en el momento de desarrollar este tipo de aplicaciones.

La separación de la capa de presentación se llevó a cabo utilizando el patrón *Model View Presenter* (gráfico 1), el cual es una adaptación orientada a eventos del ya conocido *Model View Controller* [9][10]. Este patrón permite que el usuario programador pueda realizar cambios en la interfaz de la aplicación sin que dichos cambios alteren la lógica del sistema y viceversa.

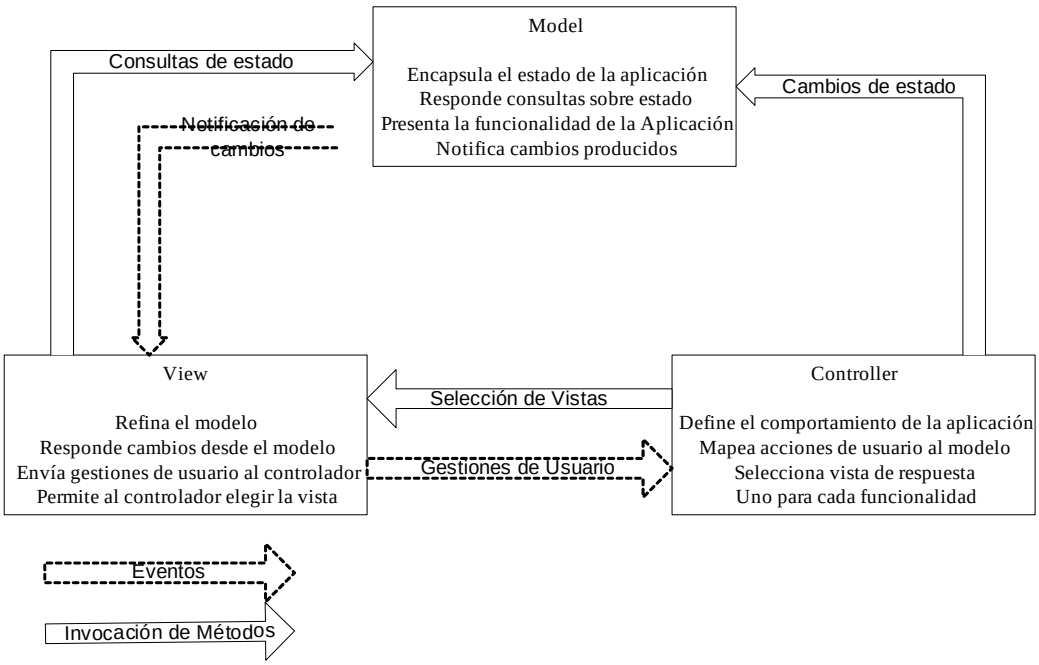

Gráfico 1

De esta manera, mediante la utilización del *Model View Presenter*, la aplicación queda dividida en una capa de presentación (View) y una capa de lógica de negocios, más una capa de acceso a datos (Model) [11]. El *Presenter* es una capa adicional del *Model View Presenter* que se encarga de administrar la interacción del usuario con la interfaz, generando también cambios en el *Model* si es necesario.

La persistencia de los objetos del sistema en un repositorio de datos no es responsabilidad de la capa de lógica de negocios, sino que es necesaria otra capa que se encargue de administrar la persistencia, denominada “Capa de acceso a datos”.

El acceso al sistema específico de almacenamiento de datos lo realiza DAO (*Data Access Object*) quien conoce el modo de conectarse con este sistema (gestor de base de datos, administrador de archivos, entre otros) y acceder a sus estructuras internas.

La capa de acceso a datos se comunica con DAO para realizar la asociación de cada una de las clases y atributos de los objetos de la capa de lógica de negocios, con las estructuras de datos internas respectivas. Generalmente se utiliza como medio de almacenamiento una base de datos relacional donde las tablas se corresponden con las clases de objetos y los campos con los atributos.

Esta separación se implementó con el patrón de diseño *Repository*. Este patrón permite que la lógica de negocios de una aplicación no necesite conocer un lenguaje SQL en particular, ni los datos de conexión a una o más bases de datos.

Con esta arquitectura de capas separadas es posible entonces aislar la implementación de cada una de ellas. dotNET4DB implementa en cada capa una serie de características comunes en sistemas WEB:

- Capa de presentación
- Capa de lógica de negocios
- Capa de acceso a datos

3.3 Capa de presentación

La capa de presentación está formada por una página principal con tres secciones:

- Encabezado: esta es una sección estática que muestra información acerca del sistema. Esta sección puede ser modificada por el usuario programador utilizando código HTML estándar.

- Barra de herramientas: es generada de manera automática por el framework. Las opciones disponibles son definidas por el usuario programador en cada uno de los módulos que conforman el sistema.
  La generación de cada módulo con interfaz visible agrega a la barra de herramientas un nuevo ítem a elegir. El usuario programador tiene la posibilidad de agregar un sub-ítem para cada sub-módulo. Esto convierte a la barra de herramientas en una lista jerárquica en forma de árbol que permite al usuario final navegar libremente por el sistema.

- Cuerpo: esta sección es la encargada de presentar el contenido de cada uno de los módulos seleccionados mediante la barra de herramientas.
  En la parte superior de cada módulo se puede encontrar un mapa navegacional, cuyo contenido se corresponde con el módulo en donde se encuentra el usuario, y el nivel de anidamiento de ese módulo en relación a los demás según se define en la barra de herramientas.

Si bien dotNET4DB permite visualizar cualquier contenido de tipo ASP en esta sección, el framework se especializa en la generación de módulos de tipo Altas, Bajas y Modificaciones (ABM). La interfaz de un módulo ABM generado por el framework consta de dos vistas principales (figura 1):

- *BrowseView* (Vista de búsqueda). Es la primera vista disponible en el momento de entrar al módulo. Está compuesta por secciones, siendo las principales la Sección de Encabezado, Filtrado de Datos y Datos en sí.
- *ActionView* (Vista de acción). La vista de acción, a diferencia de la vista de búsqueda, muestra la información de todos los campos de un solo registro.

Las principales secciones de la vista de búsqueda son:

- Encabezado
- Filtro de datos
- Datos

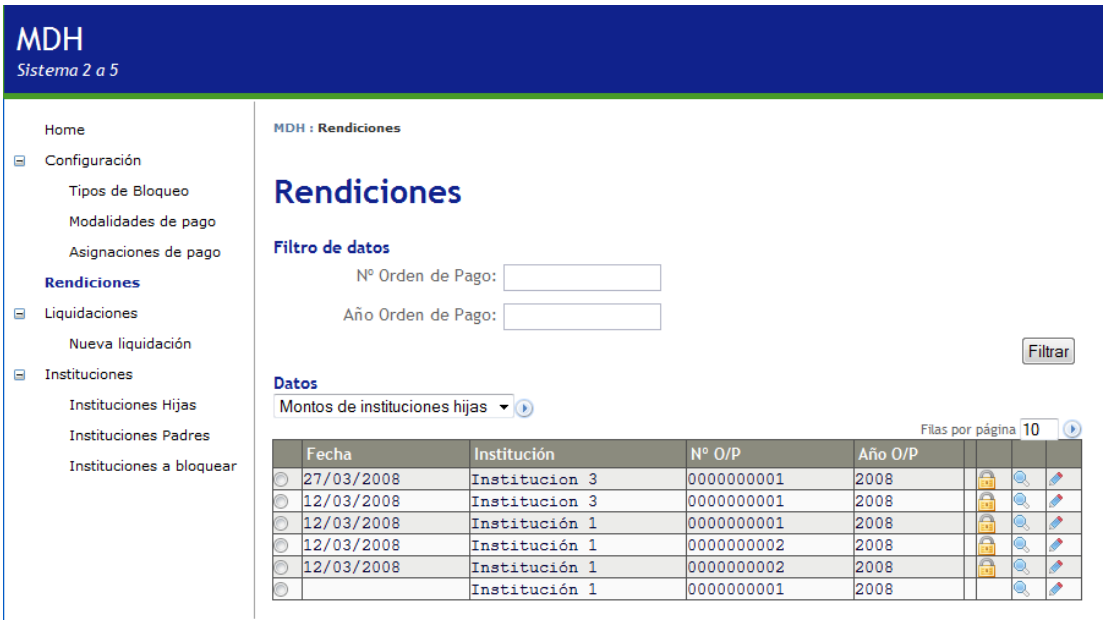

Figura 1

En la sección encabezado se encuentra el nombre o título del módulo en cuestión. Dentro del filtro de datos, el usuario programador puede especificar los campos de filtrado colocando los controles ASP.NET necesarios. Por último, la sección de datos se genera automáticamente y está formada por:

- Información de entidades padre: Utilizada generalmente en funcionalidades de tipo maestro-detalle, donde es necesario conocer en todo momento cual es la entidad padre sobre la que se está trabajando.
- Funcionalidades aplicables a una o más filas seleccionadas: Aplicable a casos donde es necesario realizar una operación sobre un registro en particular o sobre un conjunto de registros seleccionados por el usuario. Cada funcionalidad es especificada por el usuario programador como así también la URL a la que se hace referencia.

- Funcionalidades de paginación: Se genera automáticamente por el framework y permite la navegación a través de grandes volúmenes de datos. Las operaciones disponibles son: ir a la primer página, página anterior, página siguiente, última página, una página en particular (identificada por un número), y además permite modificar la cantidad de registros visibles por página.
- Grilla de datos: Se genera automáticamente por el framework. La configuración de la misma la realiza el usuario programador especificando los campos que se visualizarán junto con el tipo de datos y el formato de visualización.

La vista de acción posee diferentes estados los cuales son activados en el momento de realizar una acción en la vista de búsqueda. Los estados posibles son:

- Actualizar: muestra los campos del registro con la posibilidad de ser modificados (figura 2).
- Insertar: muestra los campos vacíos, con la posibilidad de completarlos con información.
- Eliminar: muestra los campos en modo de sólo lectura previo a la confirmación de la eliminación.
- Examinar: muestra los campos en modo de sólo lectura.

Los campos que se presentan en pantalla deben ser especificados por el usuario programador utilizando los controles ASP.NET estándar.

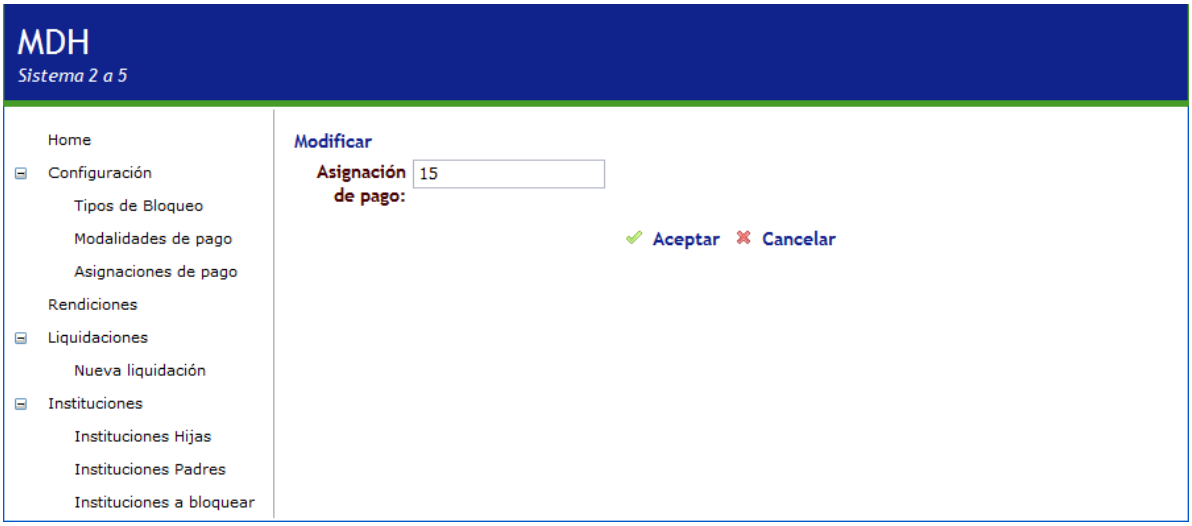

Figura 2

La figura 3 presenta la interfaz de los listados que genera automáticamente dotNET4DB.

3.4 Capa de lógica de negocios

La capa de lógica de negocios es la que contiene la lógica general del sistema. Si bien el framework se encarga de realizar una separación total entre capas, la implementación de la misma está a cargo del usuario desarrollador.

Cada entidad en la base de datos que necesita un módulo ABM, deber ser una clase en la capa de lógica de negocios.

3.5 Capa de acceso a datos

dotNET4DB logra una separación de la capa de acceso a datos a través del patrón de diseño denominado Repository. Este patrón genera una capa intermedia capaz de lograr una traducción entre las entidades de negocios (Personas, Instituciones, Usuarios) y las tablas en la base de datos subyacente.

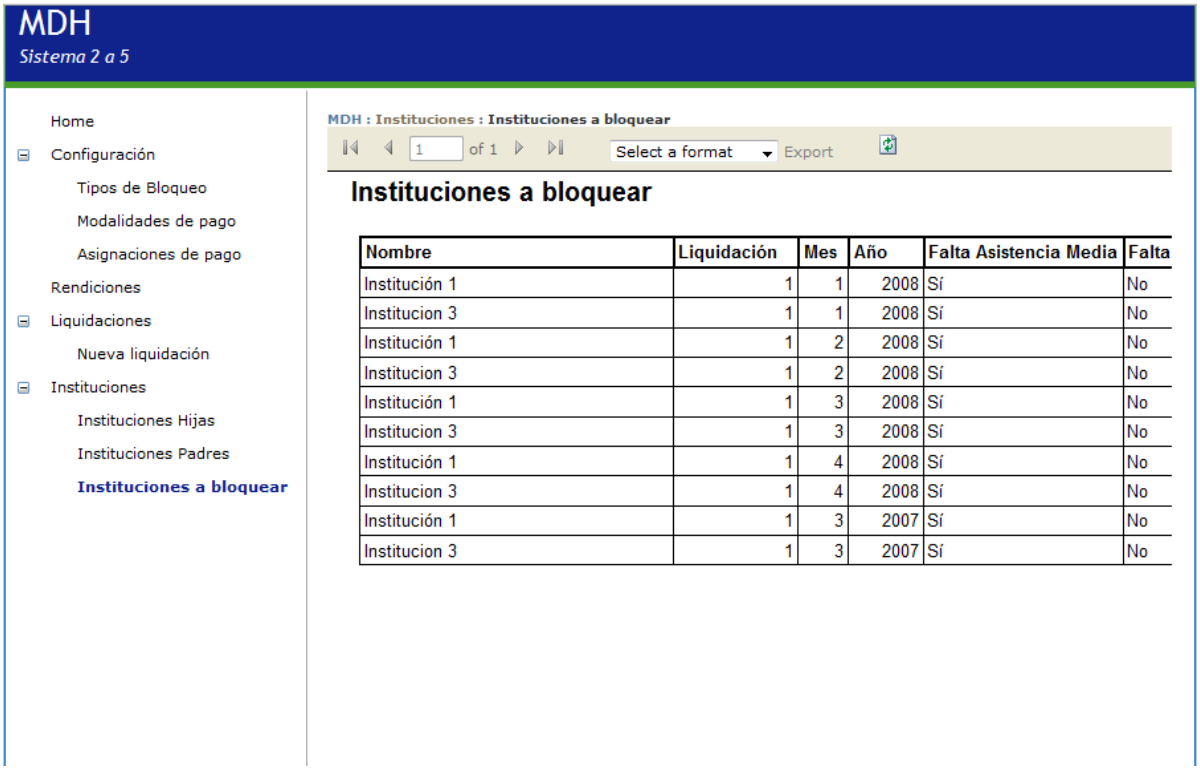

Figura 3

Los repositorios generalmente representan una tabla sobre la que se realizarán operaciones de consulta, inserción, modificación y eliminación. Cada repositorio se comunica con una capa intermedia que permite la abstracción del motor de base de datos subyacente y, de este modo, esta capa se encarga de gestionar las peticiones según el motor de base de datos con el que se está trabajando.

dotNET4DB provee al usuario programador un modelo estándar de repositorio, el cual cumple con los requisitos mínimos para implementar el acceso a datos de un módulo ABM. Las funcionalidades de un Repositorio son:

- Insertar una entidad de negocios.
- Modificar una entidad de negocios.
- Eliminar una entidad de negocios.
- Devolver una lista de entidades que respetan un criterio establecido.
- Devolver la cantidad de entidades que respetan un criterio establecido.

El usuario programador será el encargado de implementar cada una de las funcionalidades según el repositorio donde se está trabajando. Esta implementación consiste en realizar el mapeo

correspondiente entre los atributos de las entidades de negocios y los campos de las tablas de las bases de datos, como así también especificar qué acción o conjunto de acciones se llevarán a cabo sobre el repositorio de datos.

4. Conclusiones

El framework dotNET4DB fue desarrollado para facilitar y asistir la construcción de sistemas con tecnología ASP.NET. En este sentido dentro del III-LIDI fueron desarrollados sistemas experimentales para evaluar la aplicabilidad de esta herramienta. Además, una vez verificadas las ventajas obtenidas con la utilización del framework, el mismo fue puesto en producción en sistemas desarrollados dentro de convenios con terceros por parte del Instituto, siendo destacable el Sistema para el plan de Asistencia Social a Menores de 2 a 5 años, del Ministerio de Desarrollo Humano de la Provincia de Buenos Aires[7].

En concordancia con otra herramienta desarrollada por los autores, un framework de trabajo que utiliza PHP [6], se logró automatizar un gran porcentaje de los casos de uso clásicos relacionados con sistemas de gestión. En general, tanto en los sistemas experimentales como en aquellos que se llevaron a producción, el 60% de los casos de uso definidos fueron implementados de manera automática por dotNET4DB. Para el 40% restante, en tanto, el nivel de codificación particular fue variado. Sólo una pequeña porción requirió un trabajo pormenorizado por parte del analista programador y en situaciones muy puntuales del problema.

Los resultados obtenidos fueron altamente satisfactorios respecto del esfuerzo necesario para la implantación del sistema de gestión, que actualmente se encuentra en producción. Efectos secundarios como homogeneización de la interfaz de usuario y tiempos de respuesta para el mantenimiento, también fueron valorados y sobre ellos también se encontraron importantes mejoras.

Por último, la herramienta fue probada en un caso testigo como elemento de prototipación, resultanto su comportamiento muy ventajoso. Si bien el entorno fue simulado, los tiempos requeridos para generar una vista de usuario de los requerimientos generales del problema fueron muy adecuados, permitiendo obtener una presentación rápidamente. Por lo tanto, dotNET4DB siendo utilizada por usuarios experimentados, puede generar un prototipo desechable en los tiempos adecuados, por ejemplo, de una sesión de JAD.

Además de las ventajas obtenidas para el desarrollo de sistemas de software concretos, la utilidad de dotNET4DB para prototipar la convierte en un elemento de gran importancia, no sólo para el desarrollo de software propiamente dicho, sino como una herramienta de elicitación y validación de requerimientos en etapas tempranas del desarrollo.

5. Trabajos futuros

La evolución de dotNET4DB dependerá de su utilización y las necesidades específicas de los sistemas donde sea utilizado. Al igual que su homólogo PHP4DB, el crecimiento está ligado a los contextos donde el framework sirva como base para el desarrollo.

Entre los trabajos previstos a mediano plazo, están aquellos vinculado con la mejora en las grillas. Aquí se busca incorporar mayor riqueza funcional en el procesamiento, y mejorar la presentación de la información involucrada.

La personalización de interfaz de usuario así como su generación multi-idioma son las evoluciones propuestas a largo plazo.

Bibliografía

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

[2] http://agilemanifesto.org/sign/display.cgi?ms=all

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

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

[5] Bertone, Pasini, Ramón. Programación Extrema y Moprosoft: Son compatibles? Informe técnico III- LIDI. Octubre 2006.

[6] L. Delía, M. Iglesias, G. Cáseres , H. Ramón, P. Thomas, R. Bertone, Framework for Web Application Agile Development, JCS&T (Journal on Computer Science & Technology) supported by ISTEC, Special Issue on Selected Papers from CACIC 2006, Vol. 7 - No. 1 - March 2007 - ISSN 1666-6038, pág. 86-90, http://journal.info.unlp.edu.ar.

[7] Cáseres, Luna, Thomas, Bertone, Boracchia. Informe Técnico . Ministerio de Desarrollo Humano de la Provincia de Buenos Aires. Sistema de Asistencia Social para niños de 2 a 5 años. Marzo 2008.

[8] Bertone, Pasini, Ramón. Programación Extrema y Calidad. Estudio de Compatibilidad XP – CMM. Cacic 2005 (Congreso Argentino de Ciencias de la Computación) Concordia, Entre Rios. Argentina. Octubre 2005

[9] Masoud, Halabi, ASP.NET and JSP Frameworks in Model View Controller Implementation, Information and Communication Technologies, 2006. ICTTA '06. 2nd Volume 2, 24-28 April 2006 Page(s):3593 – 3598

[10] Dezhgosha, Angara; Web services for designing small-scale Web applications, Electro Information Technology, 2005 IEEE International Conference on 22-25 May 2005 Page(s):4 pp. Digital Object Identifier 10.1109/EIT.2005.1627034

[11] Leff, Rayfield,;Web-application development using the Model/View/Controller design pattern. Enterprise Distributed Object Computing Conference, 2001. EDOC '01, Proceedings. Fifht IEEE International 4-7 Sept. 2001 Page(s):118 - 127 Digital Object Identifier 10.1109/EDOC.2001.950428

[12] M. Torchiano, et. al.,Overlooked Aspects of COTS-Bases Development, IEEE Software, 2004.