Framework para el Desarrollo Ágil de Sistemas Web

Tesina de Grado

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 consisten en reducir drásticamente el tiempo de trabajo, minimizar los errores y la puesta a punto de los módulos generados, como así también respetar una interfaz homogénea entre cada uno de ellos. Estas características permiten al equipo de desarrollo concentrarse y poner énfasis en las tareas específicas 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.

Lisandro Delía

Dirección: Mg Hugo Ramón

Co-Dirección: Mg Rodolfo Bertone

Quiero dedicar esta tesina a mi familia, mi *gran* familia,

a Lau y a nuestro hijo, la *hermosa* esperanza que está por venir

Índice

1. Introducción

1. Metodologías Agiles

La evolución de la Informática y de las comunicaciones en las últimas dos décadas ha situado a los sistemas informáticos en un rol preponderante dentro de las organizaciones, llegando incluso, en algunos casos, a producir cambios radicales en la política de negocios de las mismas. Esta situación se ha visto potenciada por el auge de Internet, dando origen a nuevos modelos de negocio, donde los sistemas informáticos forman parte fundamental de los mismos.

En este contexto, los departamentos de sistemas se han visto forzados a responder de forma inmediata a los cambios del negocio, lo cual ha planteado un interesante desafío a quienes deben diseñar las aplicaciones, ya que decisiones erróneas en el diseño, pueden resultar muy costosas en el mediano y largo plazo cuando la aplicación deba evolucionar [Pae07].

Como respuesta a este nuevo escenario, es que en los últimos años han surgido nuevas tendencias en Ingeniería de Software, como los métodos de desarrollo ágiles, que facilitan el desarrollo rápido de aplicaciones.

El término “Métodos Ágiles” nace para definir a los métodos que estaban surgiendo como alternativa a las metodologías formales o clásicas (cascada, espiral), algunas de las cuales eran consideras excesivamente “pesadas” y/o “rígidas” por su carácter sistemático y fuerte dependencia de planificaciones detalladas, previas al desarrollo.

El Manifiesto Ágil, creado por críticos de los modelos de mejora del desarrollo de software basado en procesos, resume los principios de los métodos agiles [Web1]:

- La mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software con valor.
- Aceptar requisitos cambiantes, incluso en etapas avanzadas.
- Entregar software frecuentemente, con una periodicidad desde un par de semanas a un par de meses
- Los responsables de negocio y los desarrolladores deben trabajar juntos diariamente a lo largo del proyecto.
- Construir proyectos con profesionales motivados.
- El método más eficiente y efectivo de comunicar la información a un equipo de desarrollo y entre los miembros del mismo es la conversación cara a cara.
- Software que funciona es la principal medida de progreso.
- La atención continua a la excelencia técnica y los buenos diseños mejoran la agilidad.
- Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de equipos que se auto organizan.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo, entonces mejora y ajusta su comportamiento de acuerdo a sus conclusiones.

2. Aplicaciones Web

El fuerte crecimiento de Internet en la última década, ha generado nuevas necesidades en los usuarios. El acceso a internet vía teléfonos móviles, monitoreo mediante cámaras IP's, conectividad inalámbrica, comercio electrónico, entre otras, eran impensadas poco tiempo atrás. El auge de Internet también marcó un punto de inflexión en el desarrollo de aplicaciones. Las tradicionales aplicaciones de escritorio empezaron a perder terreno debido a la necesidad de poder operar desde distintas partes del mundo, de independizarse de una plataforma específica, y por el costo que tiene hacer una puesta en producción entre los clientes. Por esto, las aplicaciones web comenzaron a suplantarlas.

La situación actual del desarrollo de aplicaciones web guarda semejanza con los comienzos del desarrollo del software, cuando la calidad de este dependía totalmente de ciertas habilidades individuales. De hecho, las aplicaciones Web comúnmente son desarrolladas sin seguir un modelo de proceso formalizado: los requerimientos no son debidamente capturados y el diseño no es tenido en cuenta; los desarrolladores deben, rápidamente, atacar la etapa de implementación y poner en producción la aplicación sin un testeo correcto y completo. La calidad de las aplicaciones Web es un atributo complejo, multidimensional que involucra varios aspectos, incluyendo: corrección, fiabilidad, mantenimiento, usabilidad, accesibilidad, performance y la compatibilidad con los estándares [Ric04].

Desde el punto de vista de la arquitectura de las aplicaciones Web se distinguen dos posiciones: el cliente, en donde se encuentra el usuario final utilizando la aplicación por intermedio de un navegador, y el servidor, en donde residen los datos, reglas y lógica de la aplicación. El objetivo es permitir al cliente realizar un conjunto limitado de tareas, solo lo necesario para poder realizar su trabajo y, en cambio, hacer que el servidor realice las operaciones importantes: almacenamiento de datos, transacciones, reglas del negocio y la lógica del programa [Web2].

Con la división del problema en dos capas, se logra centralizar la administración a un solo lado, del lado del servidor, obteniendo importantes ventajas como:

- No se requiere instalación. El software basado en navegadores nunca requiere un proceso de instalación ni espacio de disco rígido.
- Actualizaciones. En lugar de actualizar cada aplicación de usuario individual, las actualizaciones se ejecutan en el servidor
- Disponibilidad a toda hora y en todo lugar
- Independencia de plataforma de SO y de cliente Web.
- La lógica de negocio no se encuentra diseminada a lo largo de la red

En el año 2005 las aplicaciones Web logran igualar o incluso superar a las aplicaciones de escritorio: el acrónimo Ajax (Asynchronous JavaScript And XML) aparece por primera vez en un artículo titulado "Ajax: A New Approach to Web Applications" [Gar05].

Ajax es un término que engloba todo lo que rodea el uso de peticiones HTTP asincrónicas iniciadas por JavaScript con el propósito de recuperar información del servidor sin recargar la página. Proporciona a los desarrolladores la capacidad de crear interfaces de usuario más

sofisticadas y con mejor respuesta, rompiendo así con el paradigma de "hacer clic y esperar" que hasta ahora había dominado la Web desde su creación.

3. PHP4DB

Inspirado en las características provistas por las metodologías ágiles en el desarrollo de software, surge la idea de implementar un Framework para aplicaciones Web capaz de dar soporte a cada una de las actividades del proceso del software [Som05]:

1. *Especificación del software* donde los clientes e ingenieros definen y enmarcar el problema y las restricciones sobre su operación, tanto para requerimientos funcionales como no funcionales.
2. *Desarrollo del software* donde el software se diseña y programa.
3. *Validación del software* donde el software se válida para asegurar que lo resuelto es lo acordado.
4. *Evolución del software* donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

El Framework propuesto tiene por objetivo minimizar los tiempos de desarrollo y testeo del software, automatizando tareas rutinarias; permitiendo implementar un proceso iterativo, donde el tiempo de entrega de productos funcionales sea mínimo y, a su vez, permita ser utilizado como herramienta de prototipación si el proyecto a desarrollar necesitase de dicha metodología.

En el próximo capítulo se detallan las motivaciones que llevan a desarrollar el Framework. En el capítulo 3 se mencionan investigaciones precedentes al trabajo. En el capítulo 4 se describe la arquitectura del framework, como está compuesto y sus características principales. Los diseños, clases principales y otras cuestiones de implementación aparecen en el capítulo 5, mientras que en el capítulo 6 se muestran ejemplos completos de cómo utilizar el framework. En el capítulo 7 se lo compara con otros Frameworks existentes, destacando ventajas y desventajas. En el capítulo 8 se analizan los resultados obtenidos, dando lugar a posibles trabajos futuros, mencionados en el capítulo 9. Finalmente, en el capítulo 10, se concluye sobre el trabajo en general.

2. Motivación

El fundamento de la IS (Ingeniería de Software) es tener un proceso establecido para el desarrollo de Sistemas de Software.

Para poder generar una capa de proceso estable y de calidad se debe partir de una especificación de requerimientos consistente del problema [Lou95]. Estos requerimientos deben [Dav95]:

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.

Una vez estabilizados los requerimientos es posible desarrollar un modelo de datos completo, ágil y dinámico que los represente de la forma más adecuada [Bat90].

A partir de este punto, un sistema que implante la funcionalidad requerida necesitará de módulos 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 [Ols98].

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. Al mismo tiempo, 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 [Som05] [Wal04].

Otra actividad frecuente de la etapa de codificación, es la de aplicar principios de seguridad, tales como la protección de los SI contra el acceso desautorizado o la modificación de información, contra la negación de servicios a usuarios desautorizados y la provisión de servicio a usuarios autorizados, incluyendo las medidas necesarias para detectar, documentar y contrariar las amenazas.

El equipo de desarrollo debe, entre otras actividades [Hum99]:

- concentrarse en la programación de las funcionalidades mínimas (utilizando un lenguaje de programación)
- actualizar la BD (generalmente con otro lenguaje específico)
- definir los formularios de carga de datos y grillas (combinando componentes visuales)
- mantener una coherencia en el desarrollo de la interfaz (aspecto importante de toda aplicación, en particular cuando se trata de un SI) presentando la información y permitiendo la interacción con el usuario de 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) y las herramientas CASE (Computer Aided Software Engineering).

Por ejemplo, un generador de código automático es un CASE que deriva, a partir de determinados patrones, el código fuente de una aplicación. La utilización 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 [Pre98]. 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.

Cuando los requerimientos cambian en el tiempo, y por consiguiente, el modelo de datos o los módulos sufren conversiones y/o adaptaciones, las herramientas CASE estáticas presentan sus defectos.

Es necesario, entonces, disponer de una herramienta 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. La complejidad en desarrollar una solución al problema, 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 [Sam01]. En aplicaciones Web, basadas en tecnología cliente-servidor [Lea00], la solución se presenta un tanto más compleja ya que se debe resolver el proceso tanto del lado del cliente (mediante JavaScript, Java Applets, etc), como del lado del servidor (mediante PHP, ASP, JSP, etc), en conjunto con un modo de mostrar la información (HTML, XML + CSS).

Esta propuesta de CASE genera una ayuda significativa en las etapas de desarrollo y mantenimiento, pero los beneficios no son aplicables únicamente en estas etapas. En la especificación del software, los prototipos ayudan a mostrar los conceptos, probar las opciones de diseño y entender mejor el problema y su solución [Som05]. Con poco esfuerzo es posible testear, comentar y rechazar ideas, fomentando la libertad creativa. Crear módulos, con funcionalidades básicas, de forma instantánea ayudaría notablemente la generación de prototipos, y en consecuencia, la especificación de software. De esta manera, los usuarios experimentarían la utilidad que tendrá el sistema y se podrían revelar errores u omisiones en los requerimientos propuestos a tiempo.

Por último, resta analizar su utilidad en la etapa de pruebas. Realizar las pruebas de una aplicación Web es más complicado que evaluar un programa tradicional debido a su naturaleza heterogénea, distribuida, concurrente y plataforma-independiente [Liu00]. La prueba de unidad resulta muy útil en este punto, la cual consiste en evaluar los componentes de un determinado módulo. Estos componentes pueden ser: funciones individuales o métodos dentro de un objeto, clases de objetos, o componentes compuestos por diferentes objetos o funciones [Som05].

La motivación final para el desarrollo de una herramienta CASE que asista al desarrollo de aplicaciones Web surgió en el Instituto de Investigación de Informática III-LIDI, donde surgieron una serie de proyectos con netas características de desarrollo web [Rod03] dada la necesidad de acceder a la información desde lugares físicamente remotos. Estos proyectos web debían ser desarrollados mediante herramientas open-source, motivo por el cual se opto por ambientes LAMP (Linux + Apache + MySQL + PHP) [Ram05] [Tor04] y debían interactuar con BD's integradas por información heterogénea. Estas aplicaciones Web requerían de una alta cantidad de repositorios de tablas con las operaciones clásicas como listados, filtros, reportes y ABM (Altas, Bajas y Modificaciones) de datos.

Por este motivo, era de gran importancia contar con una capa de software genérica que automatice el desarrollo de los repositorios y otra capa encargada de llevar el control del acceso al sistema y sus funcionalidades, actuando como un molde para todos los proyectos de similares características.

El desarrollo propuesto es el framework denominado PHP4DB [Del06a] [Del06b] [Del07] [Del08a] [Del08b], sigla que hace referencia a "PHP para base de datos" y fue diseñado para resolver los problemas presentados. PHP4DB es una herramienta orientada a objetos (OO) desarrollada íntegramente en PHP, con el objetivo propuesto de generalizar la capa de software para permitir automatizar tareas rutinarias de codificación en un ambiente LAMP y brindar un esquema de seguridad ágil, dinámico y confiable.

3. Investigación previa

1. La evolución de las aplicaciones

Junto al progreso de los procesos de computación, desde tareas batch a programas interactivos mono usuarios y a sistemas de sistemas distribuidos, los requerimientos de los lenguajes de programación y de los entornos asociados han cambiado con ellos.

Los programas antiguos podían ser escritos en lenguajes que manipulaban directamente el hardware de la computadora, maximizando la eficiencia del equipo.

Con el incremento de la capacidad de cómputo, la tendencia apuntó a la abstracción, alejando al programador del hardware, y llevando a que los programas resultantes utilicen la computadora de manera menos eficientemente.

Existen varias formas de clasificar los lenguajes de programación, pero hay dos que son especialmente relevantes en cuanto a la productividad. La primera característica es el número de instrucciones máquinas generadas por sentencia. Esta es una señal del nivel de abstracción del lenguaje de programación. Por un lado, el lenguaje Assembler tiene una instrucción por sentencia y representa el menor nivel de abstracción. Por otro lado, los lenguajes ágiles como Python o Tcl pueden generar cientos o miles de instrucciones por sentencia. Estos lenguajes de alta abstracción permiten al programador generar más funcionalidades en menor tiempo.

El segundo aspecto de interés es el grado de tipado de un lenguaje, el cual afecta el nivel de esfuerzo requerido por el programador para convertir datos de un tipo a otro. En el lenguaje Assembler, no hay conversión requerida ya que no hay otros tipos que aquellos definidos por la arquitectura de hardware. Lenguajes de sistemas como C/C++ y Java requieren que los datos tengan un tipo y sean explícitamente convertidos si existiera alguna ambigüedad en la conversión. Los lenguajes ágiles toman un enfoque diferente y llevan a cabo la mayoría de las conversiones automáticamente. De esta forma, por ejemplo, si una variable es usada en el contexto de los strings, es convertida automáticamente a string; en tanto, si está siendo usada en una expresión numérica, es convertida a un número. Las preocupaciones sobre los tamaños de las palabras o del valor del signo están ocultas en los lenguajes ágiles a menos que sean explícitamente invocadas por el programador.

Los sistemas de la actualidad requieren un rápido desarrollo de módulos con una fácil integración con otros sistemas, y es aquí donde los lenguajes de alto nivel sacan ventaja [Wri06].

2. La seguridad en las aplicaciones

Existen estándares a la hora de hablar de la seguridad de un sistema. El estándar de niveles de seguridad más utilizado internacionalmente es el TCSEC Orange Book [Doe85], desarrollado en 1983 de acuerdo a las normas de seguridad en computadoras del Departamento de Defensa de los Estados Unidos.

Los niveles describen diferentes tipos de seguridad y se enumeran desde el mínimo grado de seguridad al máximo. Estos niveles han sido la base de desarrollo de estándares europeos (ITSEC/ITSEM) y luego internacionales (ISO/IEC). Cabe aclarar que cada nivel requiere todos los niveles definidos anteriormente: así el subnivel B2 abarca los subniveles B1, C2, C1 y el D.

Nivel D

Este nivel contiene sólo una división y está reservada para sistemas que han sido evaluados y no cumplen con ninguna especificación de seguridad. Sin sistemas no confiables, no hay protección para el hardware, el sistema operativo es inestable y no hay autentificación con respecto a los usuarios y sus derechos en el acceso a la información. Los sistemas operativos que responden a este nivel son MS-DOS y System 7.0 de Macintosh.

Nivel C1: Protección Discrecional

Se requiere identificación de usuarios que permite el acceso a distinta información. Cada usuario puede manejar su información privada y se hace la distinción entre los usuarios y el administrador del sistema, quien tiene control total de acceso.

Muchas de las tareas cotidianas de administración del sistema sólo pueden ser realizadas por este "super usuario" quien tiene gran responsabilidad en la seguridad del mismo. Con la actual descentralización de los sistemas de cómputos, no es raro que en una organización encontremos dos o tres personas cumpliendo este rol. Esto es un problema, pues no hay forma de distinguir entre los cambios que hizo cada usuario.

Los requerimientos mínimos que debe cumplir la clase C1 son:

- Acceso de control discrecional: distinción entre usuarios y recursos. Se podrán definir grupos de usuarios (con los mismos privilegios) y grupos de objetos (archivos, directorios, disco) sobre los cuales podrán actuar usuarios o grupos de ellos.
- Identificación y Autentificación: se requiere que un usuario se identifique antes de comenzar a ejecutar acciones sobre el sistema. El dato de un usuario no podrá ser accedido por un usuario sin autorización o identificación.

Nivel C2: Protección de Acceso Controlado

Este subnivel fue diseñado para solucionar las debilidades del C1. Cuenta con características adicionales que crean un ambiente de acceso controlado, y lleva una auditoria de accesos e intentos fallidos de acceso a objetos. Además, tiene la capacidad de:

- restringir aún más a los usuarios la ejecución de ciertos comandos o el acceso a ciertos archivos
- permitir o denegar datos a usuarios en concreto, con base no sólo en los permisos, sino también en los niveles de autorización.

La auditoría es utilizada para llevar registros de todas las acciones relacionadas con la seguridad, como las actividades efectuadas por el administrador del sistema y sus usuarios.

La auditoría requiere de autenticación adicional para estar seguros de que la persona que ejecuta el comando es quien dice ser. Su mayor desventaja reside en los recursos adicionales requeridos por el procesador y el subsistema de discos.

Los usuarios de un sistema C2 tienen la autorización para realizar algunas tareas de administración del sistema sin necesidad de ser administradores.

El nivel C2 permite llevar mejor cuenta de las tareas relacionadas con la administración del sistema, ya que los usuarios son quienes ejecutan el trabajo y no el administrador del sistema.

Nivel B1: Seguridad Etiquetada

Este nivel soporta seguridad multinivel. Se establece que el propietario del archivo no puede modificar los permisos de un objeto que está bajo control de acceso obligatorio. A cada objeto del sistema (usuario, dato, etc.) se le asigna una etiqueta, con un nivel de seguridad jerárquico (alto secreto, secreto, reservado, etc.) y con unas categorías (contabilidad, nóminas, ventas, etc.).

Cada usuario que accede a un objeto debe poseer un permiso expreso para hacerlo y viceversa. Es decir que cada usuario tiene sus objetos asociados.

También se establecen controles para limitar la propagación de derecho de accesos a los distintos objetos.

Nivel B2: Protección Estructurada

Requiere que se etiquete cada objeto de nivel superior por ser padre de un objeto inferior. La Protección Estructurada es la primera que empieza a referirse al problema de un objeto a un nivel más elevado de seguridad en comunicación con otro objeto a un nivel inferior. Así, un disco rígido será etiquetado por almacenar archivos que son accedidos por distintos usuarios.
El sistema es capaz de alertar a los usuarios si sus condiciones de accesibilidad y seguridad son modificadas; y el administrador es el encargado de fijar los canales de almacenamiento y ancho de banda a utilizar por los demás usuarios.

Nivel B3: Dominios de Seguridad

Refuerza a los dominios con la instalación de hardware: por ejemplo el hardware de administración de memoria se usa para proteger el dominio de seguridad de acceso no autorizado a la modificación de objetos de diferentes dominios de seguridad. Existe un monitor de referencia que recibe las peticiones de acceso de cada usuario y las permite o las deniega según las políticas de acceso que se hayan definido. Todas las estructuras de seguridad deben ser lo suficientemente pequeñas como para permitir análisis y testeos ante posibles violaciones.

Este nivel requiere que la terminal del usuario se conecte al sistema por medio de una conexión segura. Además, cada usuario tiene asignado los lugares y objetos a los que puede acceder.

Nivel A: Protección Verificada

Es el nivel más elevado, incluye un proceso de diseño, control y verificación, mediante métodos formales (matemáticos) para asegurar todos los procesos que realiza un usuario sobre el sistema.

El diseño requiere ser verificado de forma matemática y también se deben realizar análisis de canales encubiertos y de distribución confiable. El software y el hardware son protegidos para evitar infiltraciones ante traslados o movimientos del equipamiento.

3. Frameworks

Los frameworks son una técnica de reutilización OO y representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Existe una gran confusión sobre si un framework es un patrón a larga escala o si es otro tipo de componente de Software. Incluso la definición de *Framework* varía, advirtiendo la dificultad de definir claramente su significado. Algunas definiciones que frecuentemente se utilizan son:

- un framework es un diseño reutilizable del todo o de una parte de un sistema que está representado por un conjunto de clases abstractas y la manera en que sus instancias interactúan
- un framework es el esqueleto de una aplicación que puede ser personalizada por los desarrolladores

Ambas definiciones son válidas según el punto desde donde se las mira, ya que la primera describe aspectos estructurales del framework mientras que la segunda describe su propósito.

En [Joh88] se puede encontrar una definición más completa: “Un framework es una aplicación reutilizable, semi-completa que puede ser especializada para producir aplicaciones concretas y específicas. El framework describe los objetos que componen el sistema y cómo éstos interactúan, sus interfaces y el flujo de control entre ellos, y como las responsabilidades de los sistemas mapean en objetos”.

Los frameworks se diseñan con la intención de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Sin embargo, hay quejas comunes acerca que el uso de frameworks añade código innecesario y que la preponderancia de frameworks competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a utilizarlos [Web3].

Una característica de los frameworks es la “inversión del control”. Tradicionalmente, los programadores reutilizaban componentes de librerías, en programas donde se las necesitaban. El programador decidía cuando invocarlas y era responsable de la estructura y el flujo de control del programa. En un framework, lo que se reutiliza es el programa principal, dejando en manos del programador la decisión de qué conectar a él. El código del programador es

invocado por el código del framework, el cual determina la estructura general del programa y su flujo de control.

Típicamente, un framework está implementado mediante un lenguaje OO como C++, Smalltalk o Eiffel. Los objetos del framework generalmente están definidos mediante una clase abstracta. Una clase abstracta es una clase sin instancias, por lo cual es únicamente utilizada como superclase. Este tipo de clase comúnmente tiene, al menos, una operación sin implementación, delegando esa responsabilidad en sus subclases. Al no contar con instancias, las clases abstractas son empleadas como una plantilla para crear subclases. Los frameworks las utilizan como diseño de componentes debido a que definen tanto la interface como el esqueleto del mismo. A partir de su definición pueden ser extendidas de acuerdo a las necesidades y usos puntuales del entorno.

Los frameworks capturan los beneficios de los lenguajes de programación OO: abstracción de datos, polimorfismo y herencia. Como un tipo de dato abstracto, una clase abstracta representa una interface forzando una implementación, la cual podría cambiar para adaptarse a necesidades puntuales. El polimorfismo es la habilidad de una variable o de un parámetro de tomar valores de varios tipos y permite al programador mezclar y unir componentes; haciendo posible la construcción de objetos genéricos, los cuales pueden trabajar con un gran rango de componentes. La herencia, por su parte, facilita la creación de nuevos componentes.

En un framework, las clases significativas generalmente resultan ser abstractas. Asimismo, es frecuente contar con una librería de componentes que contienen subclases concretas de las clases del framework. Esto último resulta ser un buen complemento para todo framework, pero no representa una característica esencial, como sí lo es el modelo de interacción y el flujo de control entre los objetos.

Los frameworks reutilizan código debido a que permiten la construcción sencilla de aplicaciones, gracias a las librerías de componentes. Estos componentes pueden ser fácilmente utilizados en conjunto con otros, debido a que todos comparten la misma interface del framework. Además, los frameworks reutilizan código desde la perspectiva que un nuevo componente puede heredar la mayor parte de su implementación a partir de alguna superclase.

La principal razón de que un framework permita la reutilización de código está en que su diseño es en sí reutilizable. El diseño de alto nivel permite descomponer un sistema en componentes pequeños, describiendo las interfaces internas entre componentes, y en algoritmos abstractos y reutilizables. Estas interfaces hacen posible el intercambio de componentes y la construcción de una gran variedad de sistemas partiendo de un pequeño número de componentes existentes.

Por último, los frameworks reutilizan el análisis. Describen los tipos de objetos que son importantes y proveen un vocabulario técnico y uniforme, el cual enriquece la comunicación de los desarrolladores a la hora de describir los problemas. Un usuario experto en un framework en particular ve a los problemas en términos del framework y naturalmente, lo dividirá en base a sus componentes. Además, fácilmente entenderá el diseño hecho por otro

usuario, ya que ambos pensarán las soluciones con componentes similares y describirán el sistema que se desea construir de manera similar.

Permitir la reutilización del análisis, del diseño y del código es, en sí, una características muy importante, pero si se piensa a largo plazo, la reutilización del análisis y del diseño proveerá mayor rentabilidad en el proceso de construcción de sistemas [Big96] [Fay99].

La Ilustración 1 [Mar00] presenta un esquema de escenario de un Framework genérico. Se dispone de un núcleo que brinda un conjunto de funcionalidades y algunos puntos flexibles (o puntos calientes) que deben ser configurados o conectados para que una aplicación específica pueda aprovechar las funcionalidades.

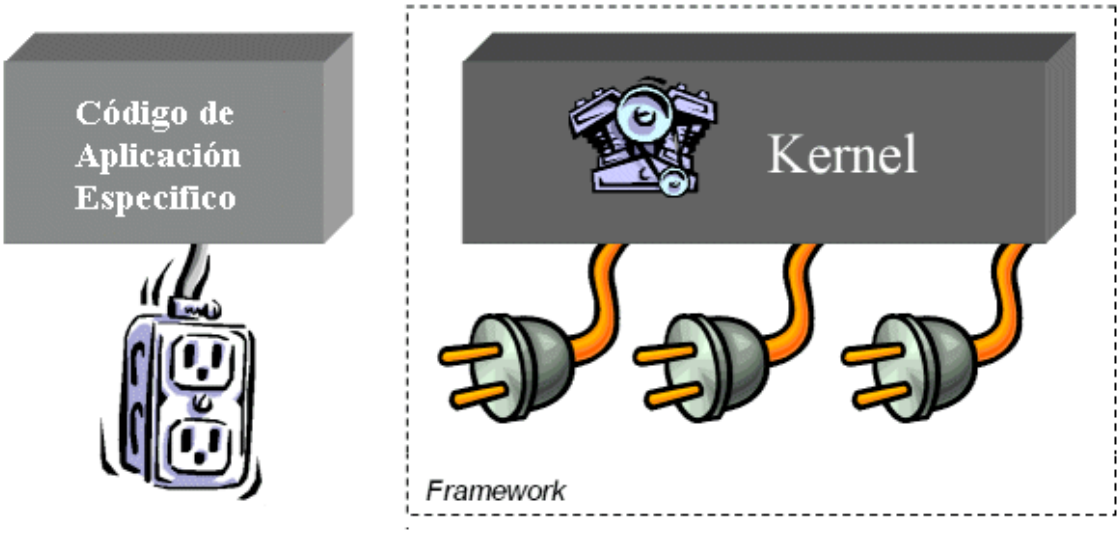


Ilustración 1: Escenario de un Framework

Los frameworks OO son una tecnología prometedora en lo que respecta a factores como la reducción de costo y mejora de la calidad del software. Frameworks como MacApp, ET++, Interviews, Advanced Computing Environment (ACE), Microsoft Foundation Classes (MFCs) y Microsoft´s Distributed Common Object Model (DCOM), Remote Method Invocation (RMI) e implementaciones de Common Object Request Broker (CORBA) juegan un rol de creciente importancia en el desarrollo del software moderno.

4. Clasificación de frameworks

Aunque los beneficios y los principios de diseño subyacentes de los frameworks son independientes del dominio en donde son aplicados, es útil clasificarlos según su alcance, así es posible definir:

Frameworks para la infraestructura de sistemas

Simplifican el desarrollo de infraestructuras de sistemas portables y eficientes como sistemas operativos y frameworks de comunicación, y frameworks para interfaces de usuario y herramientas para el procesamiento de lenguajes. Los frameworks para la infraestructura de

sistemas son usados principalmente internamente dentro de una organización de software y no se encuentran disponibles para los clientes directamente.

Frameworks para la integración del Middleware

Comúnmente utilizados para integrar aplicaciones y componentes distribuidas. Están diseñados para facilitar a los desarrolladores a modularizar, reutilizar y escalar sistemas para trabajar mejor en un entorno distribuido.

Frameworks para empresas

Abarcan amplios dominios de aplicación y son el núcleo de las actividades de negocio de la empresa. Son costosos de desarrollar y/o comprar, pero esenciales para crear rápidamente software de alta calidad.

Además de clasificarlos por el alcance, es posible clasificarlos según las técnicas utilizadas para extenderlos. Así será posible disponer de frameworks de caja blanca, de caja gris, o de caja negra.

Los frameworks de caja blanca emplean características de los lenguajes OO como la herencia y el binding dinámico, para heredar de las clases base del framework y sobrescribir métodos significativos usando patrones como el Templete Method [Gam95].

Los frameworks de caja negra soportan extensibilidad, definiendo interfaces para los componentes que pueden ser conectados al framework usando composición de objetos. La funcionalidad existente se reutiliza definiendo componentes que conforman a una interface particular, e integrando esos componentes al framework, utilizando para ello patrones como Strategy [Gam95] y Functor.

Por último, los frameworks de caja gris son diseñados para enfrentar las desventajas presentes en los frameworks de caja blanca y negra. Un framework de caja gris resulta lo suficientemente flexible y extensible, además de contar con la capacidad de ocultar información innecesaria para los programadores [Fay99].

5. Frameworks para aplicaciones Web

Un Framework para Aplicaciones Web (WAF) consiste de una plataforma modular reusable y semi-completa que puede ser especializada para producir aplicaciones web. Estas aplicaciones comúnmente son utilizadas por navegadores Web vía el protocolo http. Un WAF incluye un conjunto de servicios y componentes esenciales para la construcción de sistemas sofisticados y ricos en funcionalidades [Sha06].

En la actualidad existen varios frameworks para el desarrollo de aplicaciones web, siendo en su mayoría open source. Esta característica permite a los desarrolladores disponer del código fuente, y con esto, aprender de otros desarrolladores y reutilizar algún patrón en su trabajo.

Un patrón de diseño [Gam95] hace referencia a una solución probada en un problema recurrente en la Ingeniería de Software. Model - View - Controller [Gam95] (MVC) es uno de los patrones más populares de la Ingeniería de Software. MVC (ver Ilustración 2) es un concepto introducido por Smalltalk para encapsular datos junto a su procesamiento (el

modelo) y separarlo de la interacción del usuario (el controlador) y de las diferentes formas de presentar la información (vistas). Seguidamente se detallan las tres capas:

- El modelo representa la información de trabajo de la aplicación, es decir, su lógica de negocio.
- La vista transforma el modelo en una página Web con la que el usuario opera.
- El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

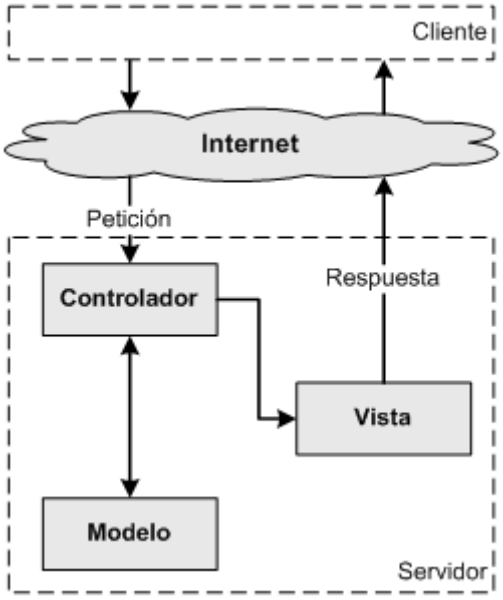


Ilustración 2: Patrón MVC

La mayoría de los frameworks para desarrollo de aplicaciones Web, como Zend [Web4], Symfony [Web5] y Kumbia [Web6], brindan soporte a este patrón de diseño, como así también otras funcionalidades. A continuación, se describen las más comunes:

- Permitir internacionalizar un sitio, es una funcionalidad encontrada en la mayoría de los frameworks. Aunque el inglés se acepte como el idioma oficial de Internet y la mayoría de sus usuarios lo entiendan, es natural que resulte mucho más atractivo y fácil de utilizar un sitio Web en el idioma del visitante.
- Disponer de una capa de abstracción a la BD es una de las características más importantes de los frameworks. La principal ventaja de contar con ella es la portabilidad, ya que permite cambiar la aplicación a otra base de datos, incluso en mitad del desarrollo de un proyecto. Si se debe desarrollar rápidamente un prototipo de una aplicación y el cliente no ha decidido todavía la base de datos que mejor se ajusta a sus necesidades, se puede construir la aplicación utilizando, por ejemplo, SQLite y cuando el cliente haya tomado la decisión, cambiar fácilmente a MySQL, PostgreSQL o Oracle, sin tener que reescribir módulos.
- La administración de la cache de una aplicación web permite reducir el uso del ancho de banda, la sobrecarga del servidor y los retrasos percibidos.

- Las utilidades Ajax son una técnica de desarrollo para crear aplicaciones web interactivas, evitando recargas innecesarias de las páginas. Las mismas aumentan la velocidad y usabilidad de los sitios, haciéndolos más interactivos.
- Generación de URL limpias al estilo http://dominio.com/articulo/mostrar/id/5 en lugar de las tradicionales URL http:// dominio.com/index.php?c=23423&s=a4j3h.
- Algunos frameworks como Symfony, fomentan el uso de entornos de trabajo separados, diferenciando producción, prueba y desarrollo. La definición de varios entornos permite utilizar una base de datos cuando se desarrolla la aplicación y otra base de datos diferente cuando la aplicación se encuentra en el servidor de producción o de prueba.
- El método Scaffolding fue popularizado por el framework Ruby on Rails [Web7] y ahora es utilizado por otros frameworks cómo CakePHP [Web8] y Kumbia. En este método el programador escribe una especificación que describe cómo debe ser usada la base de datos, para que luego el compilador utilice esa especificación para generar el código que la aplicación usará para crear, leer, actualizar y eliminar registros de la base de datos.

Es clara la diversidad de funcionalidades que un framework puede llegar a soportar. En este trabajo, me concentré en aquellas que cumplan un rol fundamental y necesario para obtener los objetivos planteados en el capitulo anterior, y dejé como tareas futuras aquellas menos prioritarias.

6. Ventajas y desventajas de los frameworks

Cuando los frameworks OO son utilizados junto a patrones de diseño, librerías de clases y componentes, se puede incrementar significativamente la calidad del software y reducir el esfuerzo del desarrollo. Sin embargo, se deben abordar ciertos desafíos para que su utilización sea correcta [Fay99]:

- Esfuerzo de desarrollo: Si el desarrollo de software complejo es una tarea difícil, desarrollar frameworks de alta calidad, extensibles, y reutilizables para el desarrollo de aplicaciones complejas resulta ser una tarea aún más difícil. Uno de los objetivos de este punto es el de desmitificar el proceso del software y los principios de diseño asociados con el desarrollo y uso de frameworks.
- Curva de aprendizaje: Aprender a utilizar un framework correctamente requiere invertir esfuerzos considerables. Por ejemplo, poder ser altamente productivo con un framework GUI suele llevar entre 6 y 12 meses de aprendizaje, dependiendo de la experiencia de los programadores. Generalmente, se requiere de cursos de entrenamiento para educar a los programadores respecto de su utilización. A menos que el esfuerzo requerido para aprender a utilizarlo pueda ser amortizado entre varios proyectos, esta inversión podría no ser efectiva en cuanto a costos se refiere
- Integrable: El desarrollo de aplicaciones tiende a ser basado en la integración de múltiples frameworks (GUIs, sistemas de comunicación, base de datos, etc) junto a librerías de clases, sistemas y componentes existentes. Sin embargo, la mayoría de los antiguos frameworks fueron diseñados para ser extendidos internamente en lugar de ser integrados con otros.

- Mantenimiento: Los requerimientos de las aplicaciones cambian frecuentemente. Por lo tanto, los requerimientos de los framework también lo hacen. Como los frameworks evolucionan continuamente, las aplicaciones que los utilizan deben evolucionar junto a ellos. El mantenimiento de los frameworks podría tomar diferentes formas, ya sea agregar nuevas funcionalidades, quitar otras y/o generalizarlas. Comprender profundamente los componentes del framework y sus interrelaciones es esencial para llevar a cabo esta tarea correctamente.
- La validación y la eliminación de defectos: Aunque un framework bien diseñado y modular puede delimitar el impacto de los defectos del software, validar y depurar aplicaciones construidas mediante frameworks puede ser altamente complejo por las siguientes razones:
    - Los componentes genéricos son más difíciles de validar en lo abstracto.
    - Distinguir entre errores del framework de los errores del código de la aplicación es una tarea delicada.
    - Puede ser difícil depurar aplicaciones construidas con frameworks, ya que el flujo de control fluctúa entre la estructura del framework y las llamadas específicas de la aplicación.
- Eficiencia: Los frameworks mejoran la extensibilidad empleando niveles adicionales de indirección. Por ejemplo, el binding dinámico es utilizado comúnmente para permitir a los programadores subclasificar y personalizar interfaces existentes. Sin embargo, estas acciones reducen su eficiencia final.
- Falta de normas: Actualmente, no existen estándares aceptados para el diseño, la implementación, la documentación y la adaptación de los frameworks. Además, los frameworks emergentes en la industria (como CORBA, DCOM y Java RMI) carecen de semántica, características, e interoperabilidad para ser verdaderamente correctos a través de múltiples dominios de aplicación.

A lo largo del capítulo, se han definido algunas de las características y problemas que presentan los frameworks que asisten al desarrollo de sistemas. Debido a su potencia y complejidad, son difíciles de aprender. Esto significa que para trabajar con un framework es necesario contar con documentación de referencia completa y disponer del tiempo necesario para realizar un entrenamiento que instruya efectivamente en su uso.

Además, un framework requiere de un mayor costo para desarrollarlos y distintos perfiles de programadores que en el desarrollo de una aplicación normal. Estas son algunas de las razones que limitan el uso de los frameworks. Aunque la reutilización es ventajosa, no es gratis, por lo que las compañías que aprovechen sus ventajas deberán pagar su precio.

7. Utilización de frameworks

Existen varias maneras de utilizar un framework. Una aplicación desarrollada utilizando un framework tiene tres partes:

- el framework
- las subclases concretas de las clases del framework

- el resto, tal como scripts que especifican qué clases concretas serán empleadas y cómo estas estarán interconectadas, como así también otros objetos en general que no tienen relación alguna con el framework.

La manera más fácil de utilizar un framework es conectando componentes existentes, lo cual no genera cambios en él, ni crea ninguna subclase concreta nueva. Se reutiliza la interface del framework y las reglas para conectar componentes. Los programadores de las aplicaciones solo deben saber que los objetos del tipo A están conectados a objetos del tipo B, sin tener que saber la especificación exacta ni de A ni de B.

No todos los frameworks trabajan de esta manera. En ocasiones, para utilizarlos se requiere crear nuevas subclases. Esto conduce a otra forma de utilizar un framework, definiendo nuevas subclases concretas y usándolas para implementar una aplicación. Como las subclases están fuertemente acopladas a sus superclases, esto implica un mayor conocimiento de las clases abstractas a comparación con el primer modo de utilización de los frameworks.

La manera más compleja de utilizar un framework es cuando se lo extiende modificando las clases abstractas que forman el núcleo del framework, generalmente agregando nuevas operaciones o variables. Esta forma se asemeja a desarmar el esqueleto de una aplicación, y generalmente requiere el código fuente del framework. Aunque es la manera más compleja, es la más potente también. Sin embargo, se debe tener en cuenta que los cambios efectuados en las clases abstractas pueden perjudicar clases concretas existentes, impidiendo que el framework funcione.

Cuando los programadores utilizan un framework conectando componentes sin tener que detenerse en detalles de su implementación, se está ante un framework denominado de "caja negra".

En tanto, aquellos frameworks basados en la herencia, requieren de un mayor conocimiento por parte de los programadores, por lo que son denominados frameworks de "caja blanca".

Los frameworks de caja negra son más simples de utilizar, mientras que los de caja blanca son generalmente más potentes cuando son utilizados por expertos. Es común que se utilice a un framework como una caja negra la mayor parte del tiempo, y sea extendido cuando la ocasión lo demande [Fay99].

8. Estudio de frameworks

Aprender un framework es más difícil que aprender una librería estándar de clases, debido a que no se puede estudiar una clase por vez. Las clases de un framework están diseñadas para trabajar en conjunto, por lo tanto, en conjunto deben ser estudiadas. Además, las clases importantes son abstractas, lo que las hace más difícil de estudiar. Estas clases no implementan todo el comportamiento de los componentes del framework ya que delegan parte del comportamiento a sus subclases concretas. Por lo tanto se debe aprender que cosas nunca cambian en el framework y cuales difieren en base a los componentes.

Es más fácil aprender a utilizar un framework si este tiene buena documentación y ejercitación. La ejercitación es de vital importancia para los frameworks complejos, e incluso

para los simples, ayuda en el aprendizaje. Pero cabe preguntarse, ¿Cuáles son las características de la buena documentación y del buen entrenamiento?

La mejor manera de empezar a aprender un framework es a través de ejemplos. La mayoría de los frameworks disponen de un conjunto de ellos, y los que no, son casi imposibles de aprender. Es más fácil aprender a usarlos mediante algo concreto, como ejemplos, que intentar de aprenderlos como un todo. Sería ideal, contar con un conjunto de ejemplos en el rango de lo simple a lo avanzado, y estos ejerciten las distintas características del framework.

Sin embargo algunos frameworks cuentan con muy poca documentación, además del código fuente y de una serie de ejemplos. Una documentación completa de un framework debería explicar: su propósito, como se lo debe utilizar y como éste trabaja.

La documentación en varios casos simplemente lista las clases del framework y los métodos de cada una de ellas. Esto no es de mucha utilidad para los principiantes, y la complejidad es similar a cuando un programador tiene que aprender un nuevo lenguaje de programación. Los programadores necesitan entender el framework en su forma global. La documentación que describa el funcionamiento interno del framework debe focalizarse en la interacción entre los objetos y como esta particionada la responsabilidad entre ellos.

El primer uso del framework es siempre una experiencia de aprendizaje, y hay que tomarlo de esa manera. Se aconseja elegir una aplicación adaptada al dominio del framework, estudiar ejemplos similares y basarse en ellos para implementar dicha aplicación. Debido a que el primer uso del framework resulta ser el más duro, es buena idea usarlo bajo la dirección de un experto.

Luego de que se haya construido la primera aplicación de prueba, la documentación del framework será más clara [Fay99].

9. Desarrollo de un framework

Una de las observaciones más comunes sobre como diseñar un framework es que se precisa de iteraciones [Joh88]. ¿Porque son necesarias? El diseño de un framework es iterativo debido a que los diseñadores no saben cómo hacerlo correctamente la primera vez. Quizás la falla está en que deberían dedicar más tiempo a analizar el dominio del problema. Sin embargo, hasta los diseñadores calificados iteran cuando diseñan frameworks.

El diseño de un framework se asemeja al diseño del software más reutilizable. Empieza con el análisis de dominio, el cual (entre otras cosas) recolecta un número de ejemplos. La primera versión del framework generalmente es diseñada para ser capaz de implementar los ejemplos y generalmente es de caja blanca. Luego se utiliza el framework para construir aplicaciones. Estas aplicaciones advierten puntos débiles en él, las cuales son partes costosas de cambiar. La experiencia brinda mejoras al framework, haciéndolo a menudo de caja negra. En algún momento, los desarrolladores deben decidir que el framework está terminado y liberarlo.

El análisis del dominio da razones para iterar. A menos que el dominio esté maduro, es difícil definir un análisis correcto desde el inicio. Los errores en el análisis son descubiertos cuando se construye el sistema, conduciendo a una nueva iteración.

Otra razón para iterar es que los frameworks son abstracciones, por lo cual su diseño depende de los ejemplos originales. Cada ejemplo que se considere hace al framework más general y reutilizable.

Un error común es empezar a utilizar el framework cuando su diseño aun está cambiando. Modificarlo genera cambios en las aplicaciones que lo utilizan. Por otro lado, la única manera de descubrir que es lo que está incorrecto en un framework es utilizándolo. El primer uso debería ser en algún proyecto piloto, para estar seguro que es lo suficientemente flexible y general. Si no lo es, esos proyectos serán un buen caso de test para los desarrolladores. El framework no debería ser utilizado en profundo hasta que haya sido probado completamente.

Debido a que los frameworks requieren de iteraciones y un profundo estudio del dominio de la aplicación, es difícil crearlos en tiempo y forma. Por esto, el diseño del framework no debe estar en el camino crítico de un proyecto. Esto sugiere que sean desarrollados por grupos de investigación o de desarrollo avanzado, y no por grupos de producción. Por otro lado, el diseño del framework debe estar asociado estrechamente con los desarrolladores de aplicaciones debido a que los diseñadores de frameworks requieren experiencia en el dominio de la aplicación [Fay99].

Existen muy pocas metodologías documentadas para el desarrollo de un framework. Las mayorías admiten que el objetivo es identificar abstracciones con un enfoque de menor a mayor: empezar examinando soluciones existentes y ser capaz de generalizar a partir de ellas. Cuando se adopta un proceso de desarrollo de un framework es importante tener en cuenta que cualquier framework empezará como una caja blanca e idealmente evolucionará a una caja negra. Además, es importante entender que desarrollar un framework es más difícil que desarrollar una aplicación debido a que los usuarios deben ser capaces de entender varias decisiones de diseño. Por esta razón, es importante seguir buenas prácticas y principios de diseño.

En [Joh93] el autor explica la diferencia entre la *ideal* y la *buena* manera de desarrollar frameworks y software en general. La ideal manera consiste en tres fases básicas. Primero analizar el dominio del problema, identificando las abstracciones y recolectando ejemplos de programas a construir. Luego, diseñar las abstracciones que puedan ser especializadas para cubrir todos los ejemplos y derivar una teoría para explicar estas aplicaciones. Y finalmente testear el framework usándolo para solucionar el problema de los ejemplos. Este ideal es imposible de seguir a fondo por dos razones principales: es muy difícil y requeriría demasiado tiempo llevar a cabo un análisis exhaustivo del dominio, analizando todos los posibles ejemplos existentes; y porque las aplicaciones existentes funcionan correctamente y no hay un incentivo financiero para actualizarlas usando nuevo software. Entonces, una menos ideal pero buena manera de desarrollar un framework es, primero, elegir dos aplicaciones en el dominio que se supone que deban ser resueltas utilizando el framework; asegurarse que alguno de los desarrolladores en el equipo han desarrollado aplicaciones para ese dominio en particular; y dividir el proyecto en tres grupos: un grupo para el desarrollo del framework que considere cómo otras aplicaciones podrían reutilizarlo; y dos grupos de aplicación que intenten reutilizarlo en otros softwares y critiquen sobre cuán difícil es hacerlo.

En la práctica, las fases importantes en el desarrollo son:

1. la fase de desarrollo del framework que usualmente es la fase que más esfuerzo consume y tiene por objeto producir un diseño reutilizable en un dominio en particular
2. la fase de uso o instanciación del framework donde las aplicaciones son desarrolladas
3. la fase de evolución y mantenimiento del framework.

Varias actividades pueden ser identificadas en la primera fase de desarrollo, como el análisis de dominio, diseños arquitecturales, diseños del framework, implementación y testeo del framework, etc. Cuando se determina el alcance del dominio existe un problema en la elección del tamaño correcto: si el framework es muy grande, varios especialistas podrían ser necesitados en el equipo y el proceso puede llegar a ser largo y costoso; si el framework es muy básico, puede tener que adaptarse en cada nueva aplicación que surja. Dado que el framework puede ser utilizado en varias, y a veces desconocidas, maneras puede simplemente no ser factible poner a prueba todos sus aspectos.

Desarrollar un framework representa un proceso iterativo. Las interfaces y las clases abstractas de un framework solo pueden ser creadas gracias a la experiencia previa; es decir, surgen o bien porque se ha llevado a cabo una tarea muchas veces en el pasado, o bien porque se reconoce en el momento de hacerla elementos en común durante una segunda o tercera implementación de una tarea pasada. Es imposible intentar crear un objeto abstracto o genérico sin disponer de alguna forma de repetición.

Mientras que los diseños iniciales pueden ser aceptables para una única aplicación, la habilidad de generalizar un diseño para muchas aplicaciones puede solo surgir construyendo la aplicación y determinando que abstracciones están siendo reutilizadas a través de las aplicaciones. Generalizar desde una única aplicación es poco común. Es mucho más fácil generalizar desde dos aplicaciones, pero aun sigue siendo una tarea compleja. La regla general es: construir una aplicación, construir una segunda aplicación que es ligeramente diferente de la primera y finalmente construir una tercer aplicación que es aun mucho mas diferente que las primeras dos. Siempre que las aplicaciones encajen en el problema de dominio las abstracciones comunes aparecerán [Dze04].

Si bien el desarrollo de un framework, como se mencionó, es algo costoso tanto en tiempo como en recursos, puede reducir el costo de desarrollar futuras aplicaciones ya que permite reutilizar tanto diseño como código, entre otros beneficios.

4. Arquitectura de PHP4DB

El objetivo de este capítulo es describir los requerimientos implementados y explicar cómo está compuesto el Framework PHP4DB.

PHP4DB es un desarrollo evolutivo: inicialmente se plantearon una serie de objetivos básicos, y una vez alcanzados, se evolucionó tanto en complejidad como en completitud.

Es posible dividir a PHP4DB en dos capas funcionales:

- Aquella que provee a las aplicaciones que lo utilicen un modulo robusto de seguridad
- Otra, de mayor complejidad, capaz de agilizar el desarrollo de módulos.

1. La capa de seguridad

PHP4DB cumple un nivel C2 de seguridad, de acuerdo a lo definido en la sección La seguridad en las aplicaciones. Define un esquema de acceso de acuerdo a perfiles de usuarios especificados en el sistema. Cada perfil, admite un conjunto de operaciones (denominadas Funciones) que el usuario puede realizar. El soporte de modelo de datos que presenta la administración de perfiles, funciones y usuarios se muestra en la Ilustración 3.

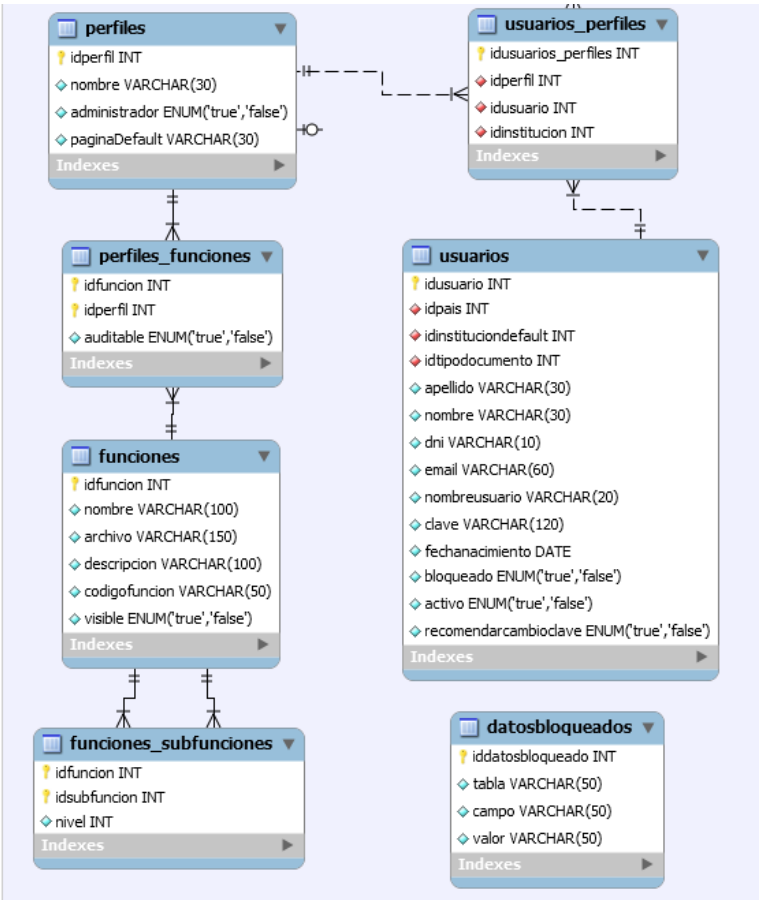


Ilustración 3: Esquema de seguridad

PHP4DB incorpora repositorios para administrar usuarios, perfiles y funciones, y sus asociaciones, permitiendo a un usuario calificado configurar todo el subsistema de seguridad.

Existen casos en que los sistemas son utilizados en organizaciones grandes, las cuales están divididas en diferentes áreas, juridiscciones, delegaciones u otro modo de división. A modo de ejemplo, la Universidad Nacional de La Plata está compuesta por unidades académicas como lo son: Facultad de Informática, Facultad de Ciencias Exactas, etc. Es posible que ciertos usuarios sean miembros en distintas áreas de la misma organización, pero cumpliendo roles diferentes. El framework contempla este caso, por lo tanto los usuarios tienen asociada una institución por defecto la cual indicará los perfiles y operaciones que le corresponden al usuario a la hora de ingresar al sistema. Luego, se debe contar con una operación para poder alternar a otras instituciones que pudiese tener asociada, actualizando los perfiles y funciones que le corresponden a la institución elegida.

El subsistema de seguridad de PHP4DB tambien opera a nivel de tupla de la BD. A menudo, se necesita que ciertos valores de la BD queden bloqueados, y sean de solo lectura. A modo de ejemplo, si se desea evitar que se pueda modificar al usuario 'admin' se debe configurar la tabla *datosbloqueados* indicando: 'usuarios' en la columna 'tabla', 'nombreusuario' en la columna 'campo' y 'admin' en la columna 'valor'.

PHP4DB añade, además, las funcionalidades básicas presentes en todo sistema multiusuario, logrando que el equipo de desarrollo se concentre rápidamente en los requerimientos "específicos" del proyecto. Estas funcionalidades básicas son:

- Loggeo de usuario (Ver Ilustración 4)
- Recuperación de la palabra clave de acceso de usuario
- Desarrollo de la interface principal de trabajo (Ver Ilustración 5) ofreciendo el acceso a las funcionalidades ligadas a los perfiles que tiene asociado el usuario
- Otra característica importante de la capa de seguridad, es aquella que permite registrar una traza de acciones. Todo evento generado por un usuario es guardado, para que luego, un usuario con el perfil adecuado pueda realizar controles, seguimientos y estadísticas respecto de la utilización del sistema por parte de los usuarios (Ilustración 6).
- Se incorpora características típicas de los sistema multiusuario, tales como:
    - bloqueo temporal de un usuario ante reiterados fallos de loggeo
    - la activación/desactivación de usuarios por parte de un administrador
    - recomendación de cambio de clave a todos los miembros de un perfil
- Por último, la capa de seguridad brinda un conjunto de clases y funciones para ser utilizadas en aquellos scripts que implementen los desarrolladores, facilitando:
    - el acceso a la BD
    - manipulación de las variables de sesión
    - creación de interfaces semejantes a las creadas automáticamente por PHP4DB
    - control de los permisos que los usuarios deben tener para acceder a una ventana.

Ilustración 4: La capa de seguridad en funcionamiento – Login del usuario

Ilustración 5: La capa de seguridad en funcionamiento – Entorno de trabajo principal

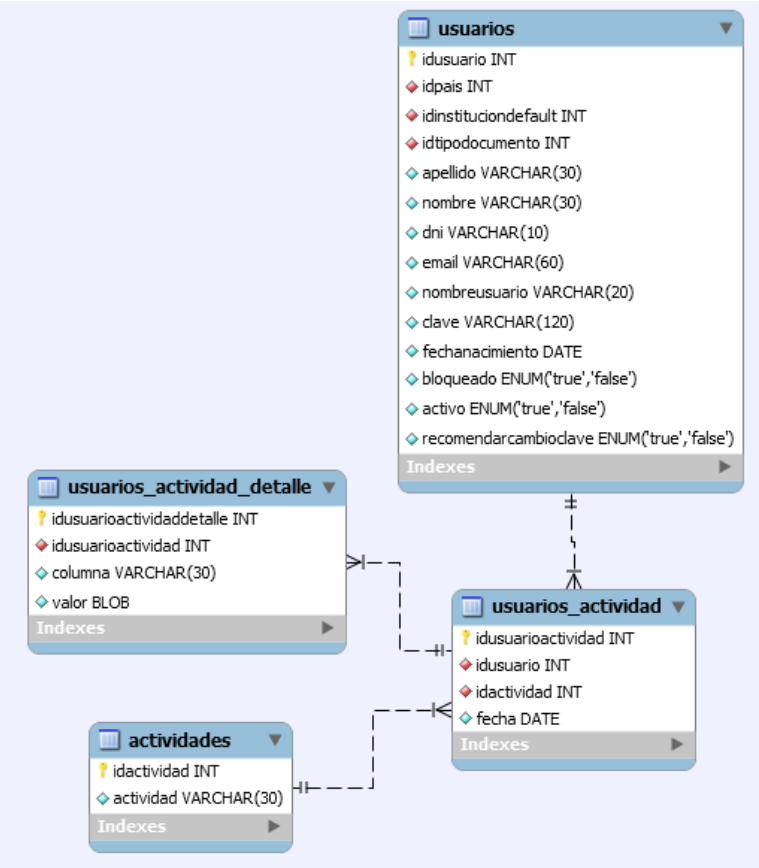


Ilustración 6: Esquema de trazabilidad de acciones

2. La capa para el desarrollo ágil de repositorios

Visualización de datos

Cuando se presenta el concepto de repositorio para una tabla, la funcionalidad inicial tiene que ver con la visualización de datos. PHP4DB presenta el contenido de una tabla de datos a partir de una grilla. Esta grilla se encuentra paginada, por los siguientes motivos:

- Es más sencillo para el usuario ver pocas filas de datos en lugar de cientos de ellas
- Es más eficiente para el DBMS retornar un número pequeño de tuplas
- La visualización de los datos en el navegador es más rápida.

PHP4DB define dos tipos de grillas:

- HTMLGrid, es una grilla nativa del framework. Permite listar los datos, paginarlos, ordenarlos por alguna columna en particular y relacionar una fila de la grilla con alguna funcionalidad especifica del repositorio. Lo destacable de esta grilla es su versatilidad para adaptarse a las necesidades del usuario (Ilustración 7). Es posible personalizarlas tanto visualmente como funcionalmente. En el primer caso, gracias al mecanismo CSS (Cascading Style Sheets) que permite controlar totalmente el estilo y formato de un documento, y funcionalmente gracias al poder de la herencia, ya que es posible crear nuevas grillas derivadas de HTMLGrid, redefiniendo o agregando comportamiento.

- ExtGrid, es una grilla incorporada en el framework ExtJS [Web9] (Ilustración 8). Estas grillas imitan el funcionamiento de las grillas de las aplicaciones de escritorio. Permiten al usuario poder cambiar de página de datos u ordenar por columnas mediante Ajax, evitando tener que recargar la ventana en su totalidad. El usuario puede ocultar o mover de lugar las columnas, como así también cambiar sus tamaños, todo desde la interface. Soportan totales, porcentajes y otros cálculos de grupos, y permiten expandir datos en cada fila, como así también la posibilidad de editar los valores en la misma grilla.

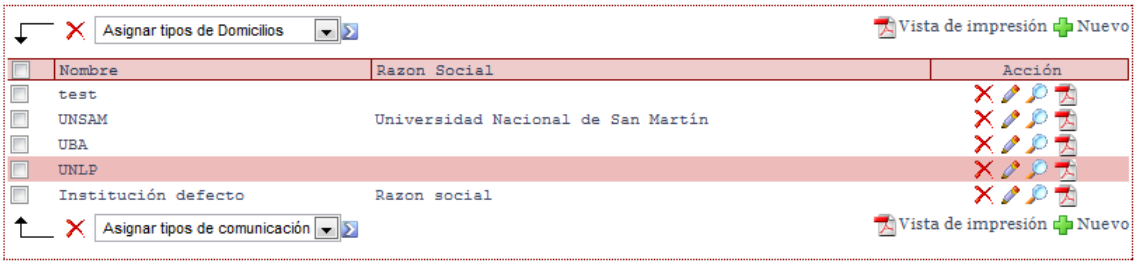


Ilustración 7: HTMLGrid

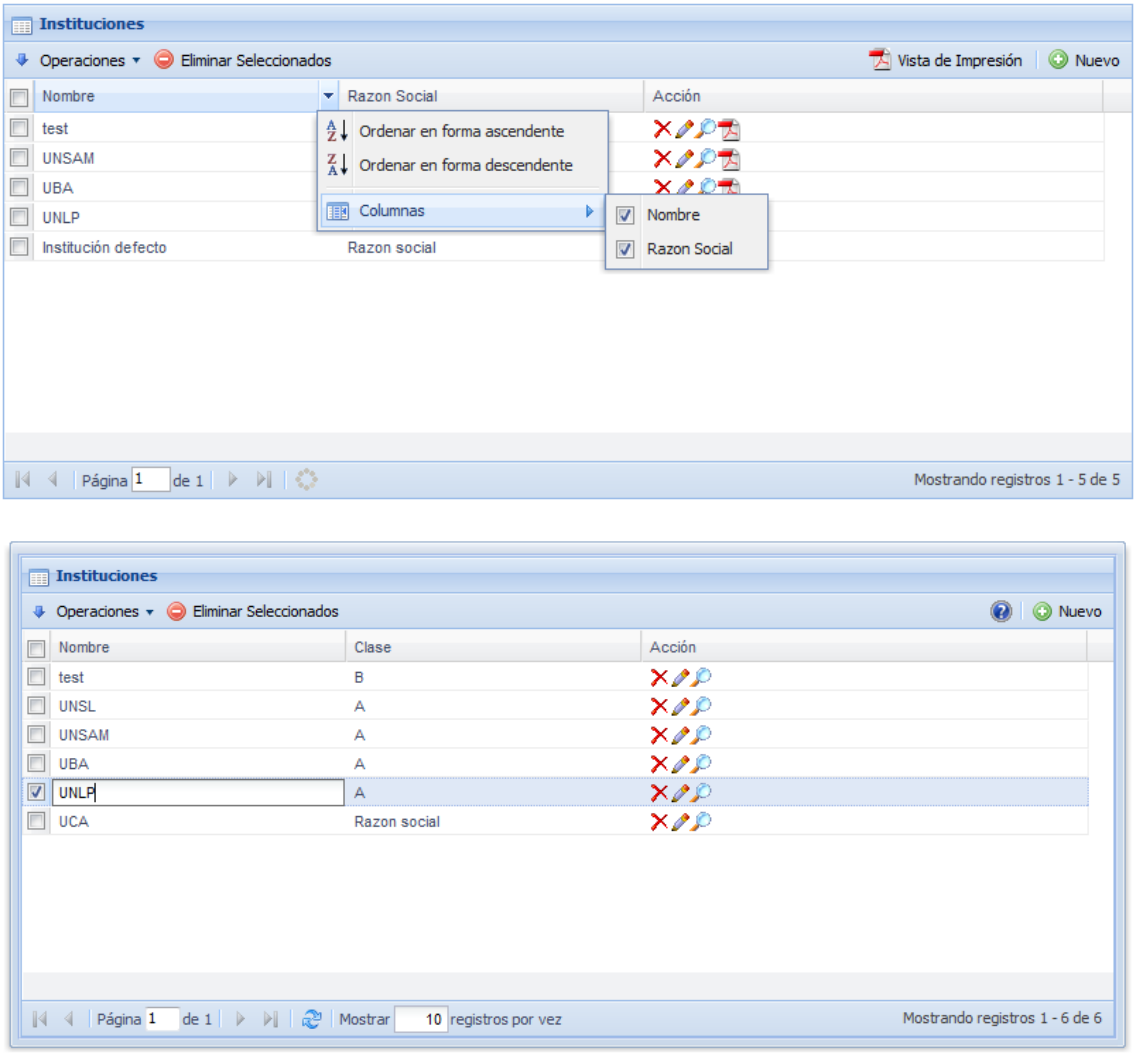


Ilustración 8: ExtGrid

El paginado de datos no es suficiente cuando se manipulan grandes cantidades de registros. El usuario, en general, puede necesitar visualizar aquellos registros de la tabla que cumplan con ciertos criterios de búsqueda. Los filtros permiten generar estas situaciones.

PHP4DB permite que se especifique cuales serán los campos que el usuario dispondrá para filtrar la información (Ilustración 9). Además, para algunos de ellos es posible configurar como será su funcionamiento en el filtro. Por ejemplo, los campos para fechas pueden filtrar las filas que cumplan una fecha exacta o las que pertenezcan a un rango de fechas.

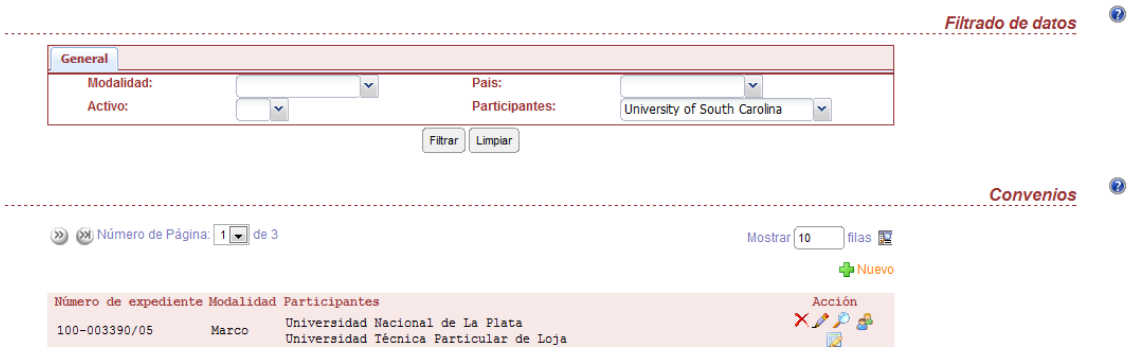


Ilustración 9: Filtros

El desarrollador es quien decide que columnas presentar en la grilla, generalmente las de mayor significancia. No obstante, se debe proveer una manera de visualizar la fila de datos completamente. Para ello, en cada fila de la grilla se dispone de una funcionalidad ( ) que muestra una fila completa (Ilustración 10).

Además, se puede aplicar a los datos listados, las siguientes operaciones nativas de PHP4DB:

- Visualizar el manual de ayuda de un *repositorio* según la acción que se esté ejecutando
- Generar un reporte formato PDF de todos los datos que se visualiza en la grilla o de un dato en particular
- Exportar a archivos CSV los datos visualizados en la grilla
- Relacionar un dato en particular con otra funcionalidad externa al repositorio, la cual puede ser otro repositorio desarrollado mediante PHP4DB, o un script desarrollado por el programador

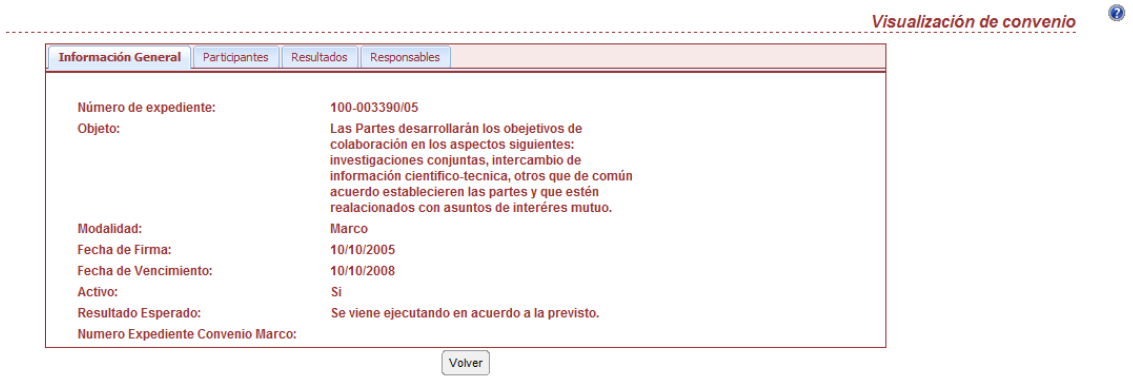


Ilustración 10: Vista completa de una fila

Actualización de datos

La información que administra un contenedor debe cambiar en el tiempo. Para ello se cuenta con las operaciones clásicas de alta, baja y actualización de datos. PHP4DB es capaz de generar los formularios asociados a estas operaciones liberando a los desarrolladores de este trabajo. Lo mismo ocurre con la validación de los datos ingresados y el control de la completitud de aquellos campos requeridos. La Ilustración 11 intenta ejemplificar posibles formularios.

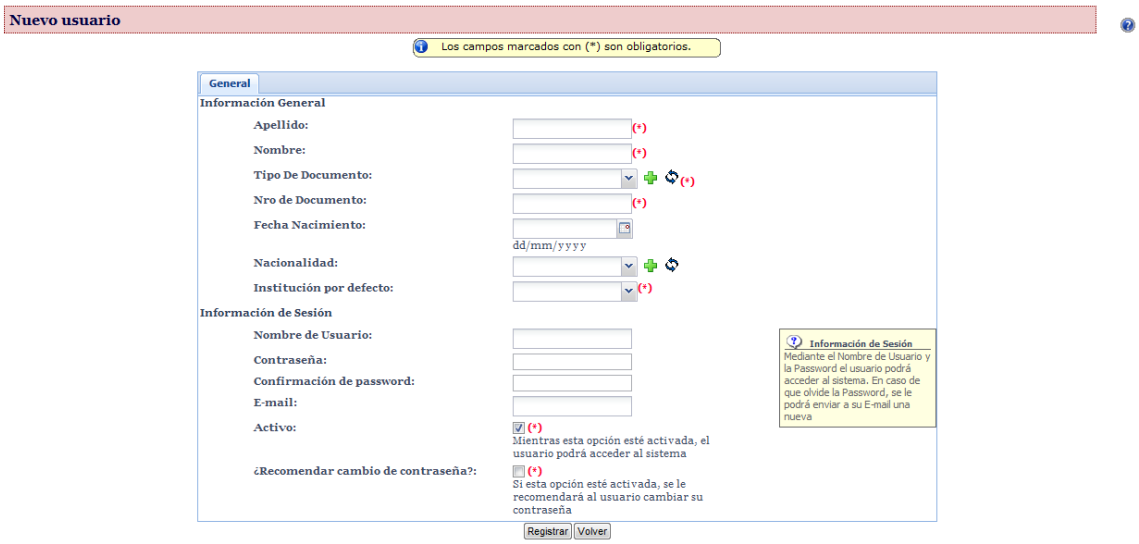


Ilustración 11: Ejemplo de formulario

El procesamiento de estos formularios también es realizado por PHP4DB, quien recibe los datos ingresados por el usuario y se encarga de armar las sentencias de inserción, eliminación o actualización y luego, ejecutarlas.

En la sección La capa de seguridad de este capítulo se mencionó que toda interacción por parte del usuario con el sistema es registrada en la BD. En algunas situaciones, las operaciones de alta, baja y modificación pueden requerir una auditoria más compleja. Para ello, es posible indicar que se desea realizar una auditoría específica en el contenedor. Esta auditoría es en formato xml y registra qué usuario realizó la operación, qué datos introdujo (para el caso de alta y modificación), cuándo (fecha y hora) y qué datos habían previamente a que se ejecute la operación (para el caso de baja y modificación).

La Ilustración 12 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 repositorio. En la grilla se listan los datos, los cuales pueden ser derivados, por ejemplo, a un reporte en formato PDF.

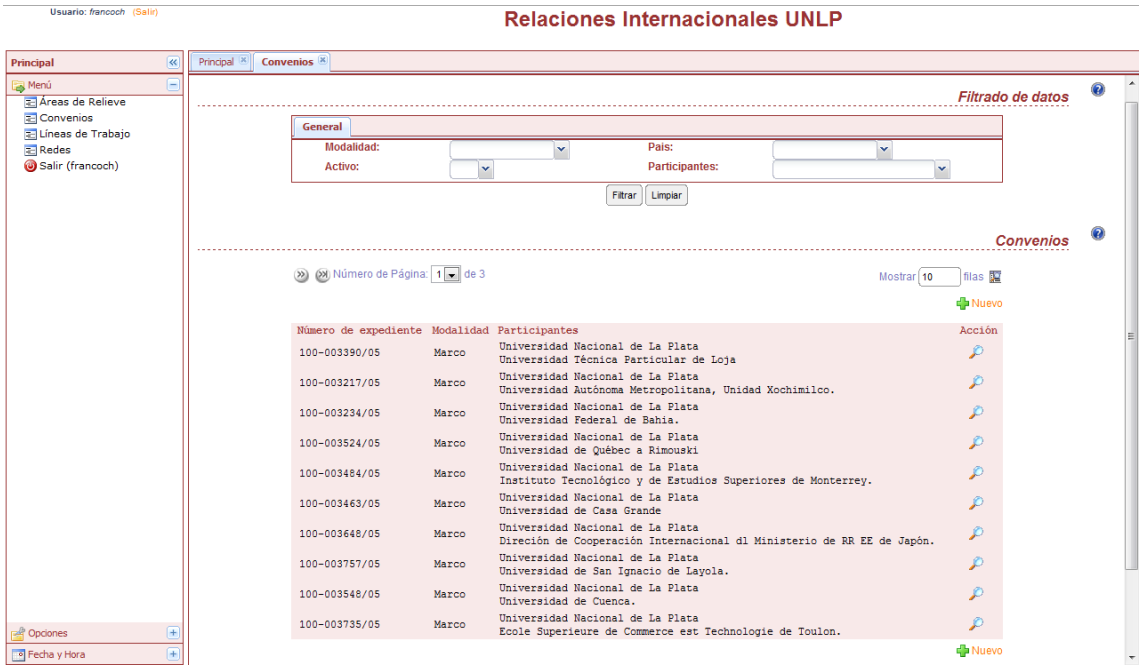


Ilustración 12: Ejecución de un repositorio

3. Estructura

PHP4DB está diseñado como un núcleo centralizado. Este núcleo precisa ser configurado para cada proyecto específicamente, a fin de responder a cada aplicación desarrollada. Esto se consigue mediante la definición de un ProyectDataScript (PDS), archivo de configuración de la aplicación, el cual describe la BD del proyecto (servidor, DBMS, usuario, clave), las interfaces adaptadas al usuario (hojas de estilo CSS) e información adicional necesaria, común a los repositorios de la aplicación.

Cada repositorio tiene asociado un archivo denominado FDS (FormDataScript), encargado de brindar la información específica de un repositorio. Los FDS describen:

- Títulos para cada operación básica del repositorio
- Permisos para cada función del repositorio, con el objetivo de habilitar/deshabilitar funciones de acuerdo al perfil del usuario
- Nombre de la tabla de la BD que hace referencia el repositorio
- Campos de la tabla de la BD. Algunos de los atributos que se necesitan para cada campo son:
    - Nombre que lo identifica
    - Etiqueta significativa a mostrar
    - Visibilidad en grilla y filtro
    - Obligatoriedad
    - Tipo del campo

La Ilustración 13 presenta la estructura externa del Framework PHP4DB desde una aplicación específica. Como se mencionó líneas atrás, cada FDS contiene la información de un repositorio en particular. El núcleo PHP4DB lleva a cabo sus funciones automáticamente, cuando un FDS es invocado.

Cada funcionalidad que se incorpore al Framework estará presente para cada repositorio sin realizar modificaciones. Esto ocurre, además, con el mantenimiento de cada funcionalidad o la corrección de errores: todo repositorio recibirá estos beneficios, sin necesidad de alterar su contenido.

Es interesante destacar que un FDS no necesita programación (ya sea código PHP, HTML o SQL) para poder crear las interfaces, presentar los datos, ejecutar las funciones, etc. El Framework se encargará de realizar estas operaciones en base a la configuración del FDS.

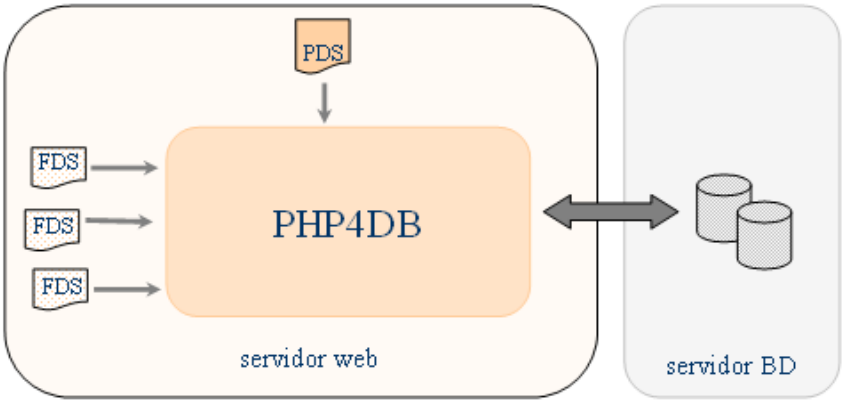


Ilustración 13: Estructura externa del Framework PHP4DB

Internamente, PHP4DB utiliza un Framework JavaScript llamado ExtJS el cual permite crear aplicaciones ricas de Internet (Rich Internet Applications), combinando las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. ExtJS incluye una gran cantidad de widgets para crear interfaces complejas, que son utilizadas por PHP4DB.

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. Para lograr la abstracción requerida se utilizó la librería MDB2 [Web10] de PEAR (PHP Extension and Application Repository) [Web11].

La Ilustración 14 presenta la estructura interna del Framework PHP4DB.

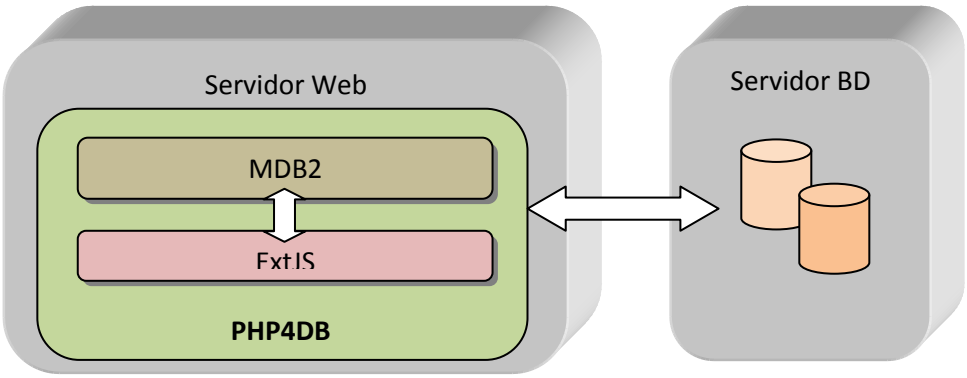


Ilustración 14: Estructura Interna del Framework PHP4DB

5. Implementación de PHP4DB

1. Repositorios (Clase Form)

En la sección Estructura del capítulo Arquitectura de PHP4DB se mencionó que cada repositorio tiene asociado un archivo denominado FDS, encargado de brindar la información específica de un repositorio. En el FDS se especifica y se instancia la clase que describe el repositorio, la cual es una especialización de la clase *Form* de PHP4DB. La Ilustración 15 grafica la clase Form y sus relaciones. Un objeto de la clase Form está compuesto por componentes (clase Type) quienes representan los campos de la BD (más adelante se estudiará en profundo las distintas opciones que se tiene para representar campos), los cuales están ubicados dentro de pestañas (clase Tab) y, a su vez, dentro de paneles de agrupamientos de campos (clase Fieldset). En cuanto a las operaciones del repositorio, además de las clásicas operaciones de alta, baja y modificación, se puede contar con funcionalidades específicas. Por esto, la clase Form cuenta con un conjunto de funcionalidades extra (clase ExtraFunctionality). Un repositorio puede ejecutar distintas acciones (clase Action) de la cual se explicará en detalle más adelante.

2. Relación entre repositorios

En el capítulo anterior se mencionaron las acciones básicas que pueden ser aplicadas a los repositorios. Existen casos en que se necesita aplicar operaciones inherentes al repositorio vinculadas con el comportamiento definido por los requerimientos del sistema.

PHP4DB cuenta con la clase *ExtraFunctionality* la cual representa una operación específica para un repositorio (de aquí en más ExtraFunctionality).

PHP4DB puede presentar las ExtraFunctionality de distintas maneras:

- Barra de herramientas. La grilla dispone de una barra de funcionalidades para aplicar a una o varias filas (Ilustración 16)
- Iconos de acceso directo en cada fila de la grilla (Ilustración 17)
- Lista desplegable de funciones para aplicar a una o varias filas (Ilustración 18)

Una ExtraFunctionality permite relacionar una tupla del repositorio con otro módulo, el cual puede ser desarrollado también mediante PHP4DB. Para dicho caso es posible indicar que la ExtraFunctionality sea “expandible” y así, poder visualizar los datos relacionados con una fila de la grilla del repositorio (Ilustración 19).

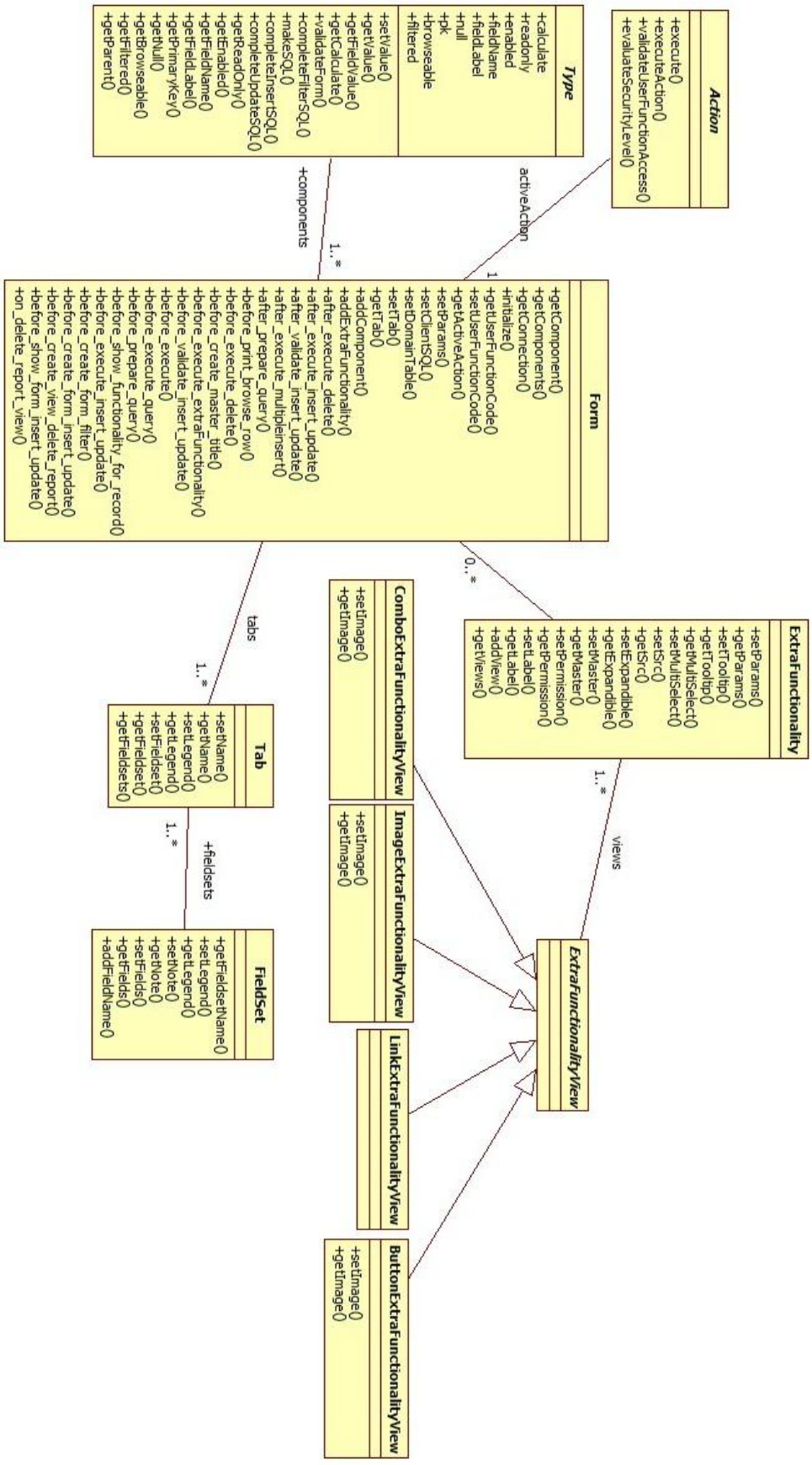


Ilustración 15: Clase Form y sus relaciones

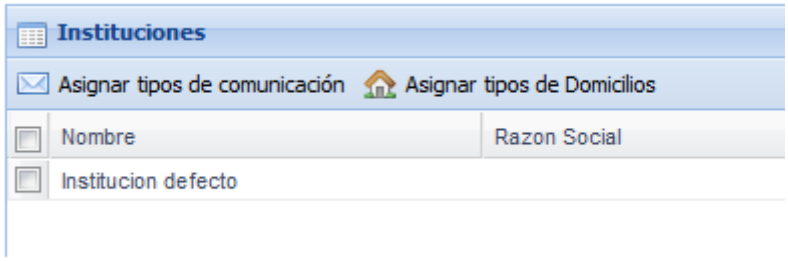


Ilustración 16: Barra de herramientas

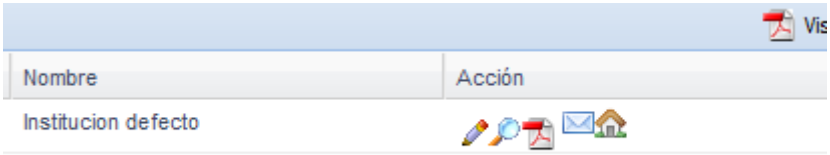


Ilustración 17: Iconos de acceso directo

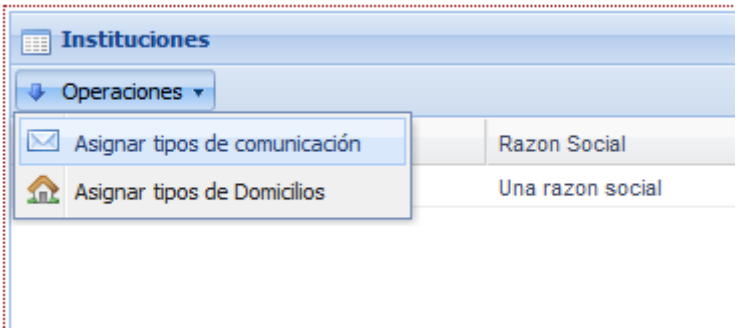


Ilustración 18: Lista desplegable de funciones

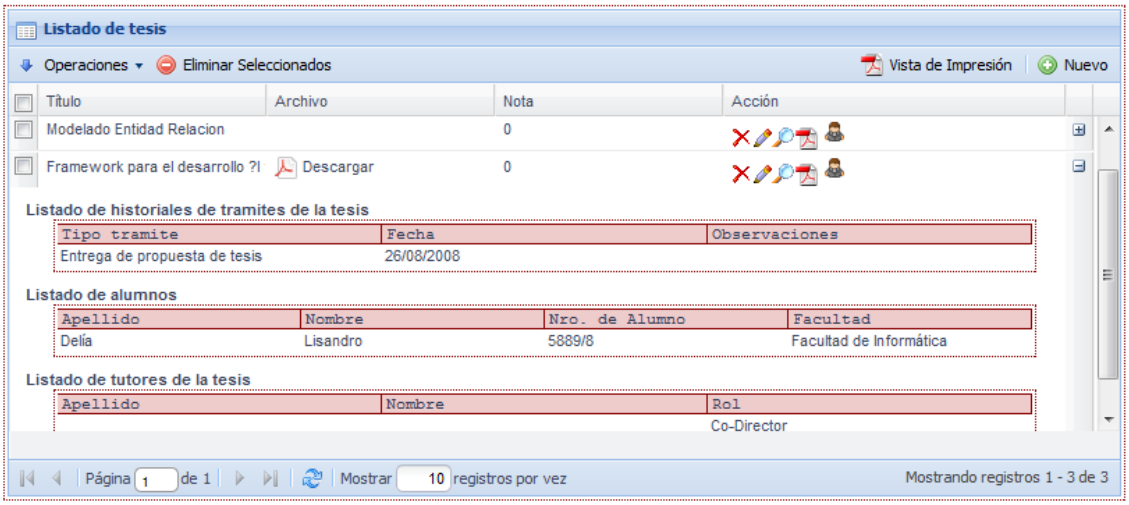


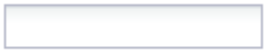
Ilustración 19: Funciones expandibles

3. 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 presentar 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.
A continuación se describen los distintos tipos de campos que provee PHP4DB.

Campos de texto

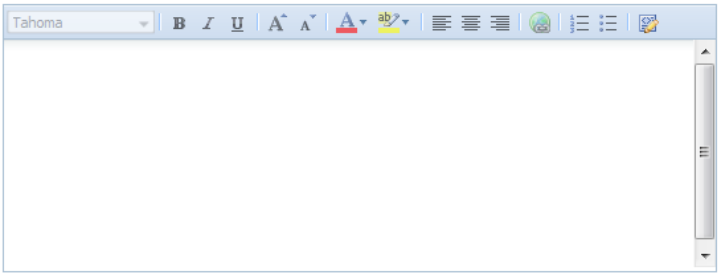
La clase *TextType* representa la tradicional casilla para escribir una línea de texto. El tamaño de la casilla puede definirse mediante el método *setSize* y la extensión máxima del texto con el método *setLength*



Campos de texto con múltiples líneas

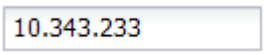
La clase *MemoType* se utiliza para definir un cuadro de texto más grande que la línea simple propuesta por la clase TextType. La cantidad de columnas puede definirse mediante el método *setCols* y la cantidad de filas mediante el método *setRows.*

Existen casos en que se desea poder escribir texto con formato. Para esto PHP4DB cuenta con la clase *RichMemoType*, actuando como un editor de texto más.



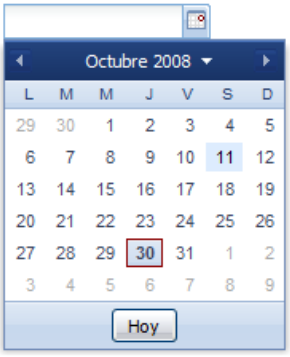
Campos de enteros y flotantes

La clase *IntegerType* se utiliza para definir un cuadro de texto limitado a números. Mediante el método *setAllowThousandSeparator* es posible enmascarar el número con separadores de miles. La clase FloatType permite además escribir números con decimales, y mediante el método *getDecimalPrecision* se puede configurar la cantidad de decimales a mostrar.



Campos de fechas

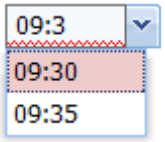
Es posible trabajar con fechas mediante la clase *DateType*. Los campos para fecha generados permiten al usuario escribir una fecha valida o seleccionarla de un calendario desplegable.



Campos para horarios

Los horarios son representados por la clase *TimeField* mediante una lista desplegable y filtrable para horas y minutos. Es posible configurar las horas mínimas y máximas que aceptará el campo, como

así también el incremento entre un horario y el siguiente.

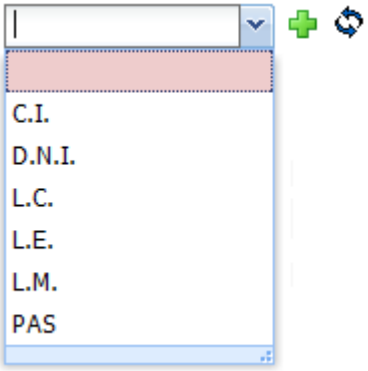


Campos para contraseñas

Los campos para la registración o actualización de contraseñas se crean mediante la clase *PsswrdType.*

Campos para claves foráneas

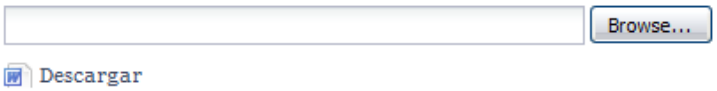
Cuando un repositorio referencia una tabla que se relaciona con otra por medio de una clave foránea es deseable mostrar los datos foráneos en lugar del valor de la relación. Para esto se cuenta con dos maneras de resolverlo. La primera, mediante la clase *FrKeyType*, consiste en una lista desplegable y filtrable de datos foráneos. El método *setFrTable* permite indicar al objeto cual es la tabla foránea, *setFrIndex* permite identificar cual es la clave única de la tabla foránea, mientras que mediante *setFrKeyFieldName* se puede especificar el campo a mostrar en la lista desplegable.



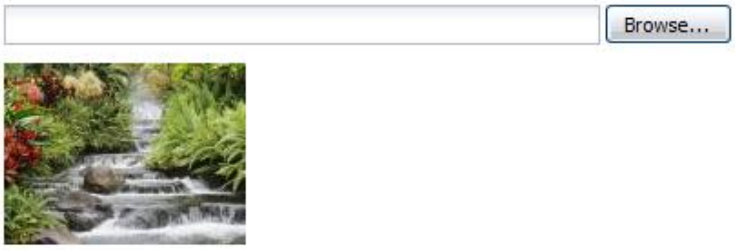
A menudo, la búsqueda por parte del usuario del elemento relacionado no es simple, y requiere, por ejemplo, de filtros específicos. También es frecuente que no se desee mostrar una sola columna foránea, si no múltiples. Para estos casos en que la lista desplegable no es suficiente se cuenta con la clase *NFrKeyType* que permite un mayor grado de flexibilidad. Para la selección de un valor foráneo, se tiene un botón que abre una ventana la cual visualiza el repositorio asociado a la tabla foránea, y en ella se permite filtrar, seleccionar y retornar un valor. El método *setFrSrc* permite especificar cuál es el repositorio foráneo a utilizar.

Campos para archivos

La clase *FileType* permite crear un control para poder operar con cualquier tipo de archivo. Para los casos de edición, el usuario puede elegir un archivo y luego se guardará el contenido del mismo en la base de datos. Para los casos de visualización se dispone de una operación de descarga.



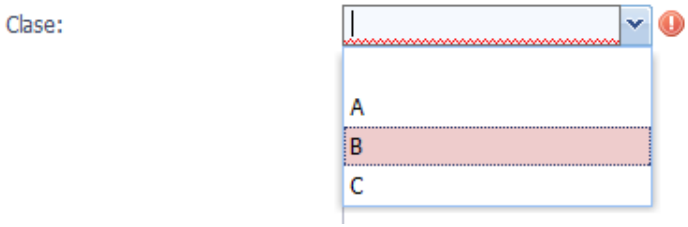
Las imágenes están representadas por la clase *ImageType* la cual presenta, además de la opción de descarga, una vista en miniatura de la imagen original a diferencia de *FileType*.



Campos booleanos

La clase *BooleanType* es utilizada para crear controles para valores booleanos. Para el caso en que el control es requerido se utilizará una casilla de selección la cual tiene 2 estados: seleccionada y no seleccionada. Para el caso en que el campo no es requerido se debe poder tener la posibilidad de dejarlo sin completar, lo que da lugar a un tercer estado. Este último caso es representado mediante una lista desplegable de estados.

Campos de valores definidos por el usuario



Existen casos en que un campo de la tabla de la base de datos admite solo un conjunto finito de valores definidos por el usuario. PHP4DB cuenta con tres clases distintas que permiten operar con este tipo de campos: *ComboBoxSetType*, *ListBoxSetType* y *RadioGroupSetType*. La clase *ComboBoxSetType* permite seleccionar uno de los valores definidos por el usuario mediante una lista desplegable, la clase *ListBoxSetType* mediante una lista de selección mientras que la clase *RadioGroupSetType* hace lo propio mediante una lista de *radio buttons.* Estas clases cuentan con el método *setValues* para especificar los valores definidos por el usuario.

Extensibilidad de tipos

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*. La Ilustración 20 muestra la jerarquía de clases de campos posibles.

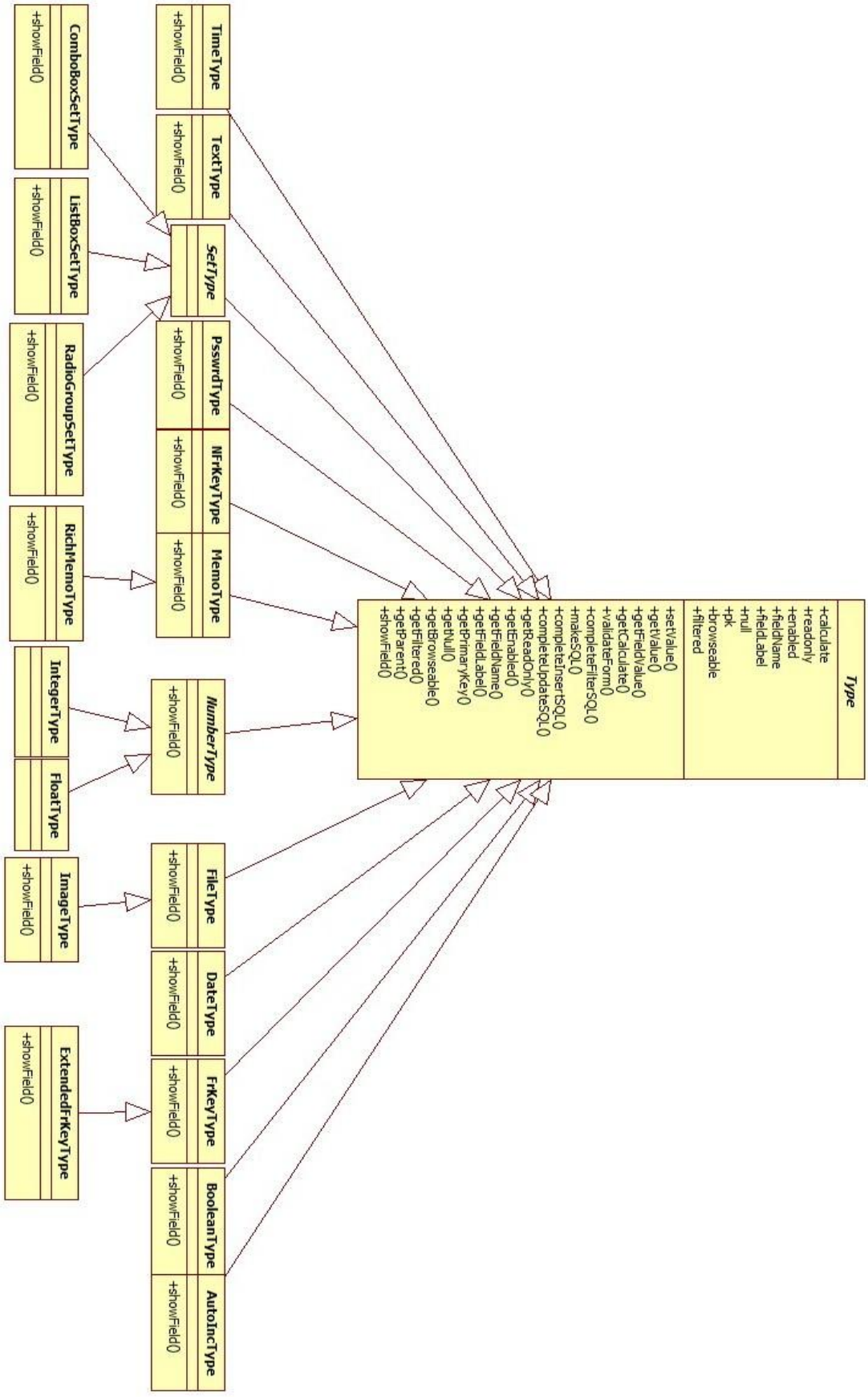


Ilustración 20: Subdiagrama de clases - Tipos de campos

4. Actions

Toda funcionalidad de PHP4DB está representada por clases Actions (Ilustración 21). Existen dos grandes grupos de Actions: las que heredan su comportamiento de FrontgroundAction y las que lo hacen de BackgroundAction. Las siguientes ilustraciones muestran los diagramas de clases de las Action.

Las clases que heredan comportamiento de FrontgroundAction (Ilustración 22) representan a todas las funcionalidades visuales en donde el usuario interactuará. Ejemplos de estas funcionalidades son: creación de formularios, de grillas, etc. A continuación se explica brevemente cada FrontgroundAction:

- BrowseAction
  Es la clase por default que se instancia cuando un repositorio es invocado. La clase BrowseAction puede estar dividida en dos secciones: Un formulario de filtrado de datos y una grilla que lista las tuplas de la tabla de dominio del repositorio. A partir de esta interface se puede seguir con el alta, baja y modificación de datos, con la ejecución de una ExtraFunctionality o con la impresión de reportes.
  - SelectAction
    A diferencia del BrowseAction, la clase SelectAction no permite seguir con el alta, baja y modificación de datos, con la ejecución de una ExtraFunctionality o con la impresión de reportes. Lo único que se permite es seleccionar una tupla de la grilla y retornarla a la ventana padre.
  - QuickBrowseAction
    En la sección anterior se señaló que una ExtraFunctionality puede ser expandible (Ilustración 19). Cuando esto ocurre, se instancia vía Ajax la clase QuickBrowseAction que permite listar los datos relacionados a la tupla donde la ExtraFunctionality es expandida.
- HelpFileAction
  Las clases BrowseAction, InsertAction, UpdateAction, DeleteAction, ReportAction y DeleteAction permiten instanciar un objeto de clase HelpFileAction, el cual visualiza al usuario una guía de ayuda explicando el funcionamiento de la interface de la Action corriente.
- UpgradeAction (Clase abstracta)
  - SimpleUpgradeAction (Clase abstracta)
    - InsertAction
      Esta clase visualiza el formulario asociado al repositorio con el fin de dar de alta una nueva tupla. Cuando el usuario confirma los datos se instancia un objeto de la clase ExecuteInsertAction, el cual procesará los datos ingresados.
    - UpdateAction
      Esta clase visualiza un formulario para actualizar una tupla de la tabla de dominio del repositorio. Inicialmente el formulario se inicializa con los valores de la tupla seleccionada en la grilla del BrowseAction, para que el usuario pueda modificarlos. Cuando el usuario confirma los

datos, se instancia un objeto de la clase ExecuteUpdateAction, el cual procesará los datos actualizados.

  - MultipleUpgradeAction (Clase abstracta)
    - MultipleInsertAction
      Un objeto de clase MultipleInsertAction presenta un formulario que permite asociar tuplas de una tabla foránea con una tupla maestra de otra tabla. Se listan las filas de la tabla foránea y el usuario selecciona las que desea asociar a la fila maestra. Cuando el usuario confirma la selección, se instancia un objeto de la clase ExecuteMultipleInsertAction, el cual procesará las filas seleccionadas.

- DeleteReportAction (Clase abstracta)
  - ReportAction
    La grilla del BrowseAction visualiza las columnas de la tabla asociada al repositorio que el desarrollador considera de mayor significancia. La clase ReportAction permite visualizar la tupla en su totalidad, incluyendo las columnas visualizadas en la grilla como las que no.
  - DeleteAction
    Esta clase visualiza la tupla que se desea eliminar, pidiéndole al usuario la confirmación de la operación. Luego de confirmarse la operación se instancia un objeto de la clase ExecuteDeleteAction.

Las clases que heredan comportamiento de BackgroundAction (Ilustración 23) representan a todas las funcionalidades que se ejecutan en el servidor y generan una respuesta, generalmente consumidas por un FrontgroundAction. Ejemplos de estas funcionalidades son: almacenamiento de datos en la base de datos, recuperación de datos, ejecución de ExtraFunctionality, etc. A continuación se explica brevemente cada BackgroundAction:

- GetNFrFieldDescDataAction
  En los formularios de alta y de modificación, la clase NFrKeyType cuenta con un botón el cual permite seleccionar una tupla de la tabla foranea por intermedio de una nueva ventana (SelectAction) que incluye el filtrado de datos y la grilla. Luego de seleccionar una tupla y de retornar el valor al formulario se actualiza mediante Ajax una etiqueta que presenta el dato previamente seleccionado. La clase GetNFrFieldDescDataAction genera el valor a presentar en dicha etiqueta.
- GetFrFieldDataStoreAction
  En los formularios de alta y de modificación, la clase FrKeyType cuenta (ademas de la lista desplegable) con un botón que permite agregar nuevas tuplas en la tabla foranea y otro para recargar el contenido de la lista. Cuando se recarga via Ajax, se instancia un objeto de la clase GetFrFieldDataStoreAction el cual genera los valores actualizados para luego recargar la lista desplegable.
- DownloadFileAction
  Cuando un repositorio cuenta con un campo de clase FileType se debe poder disponer de la posibilidad de descargar el archivo guardado en la tupla de la BD. PHP4DB permite, tanto en las filas de la grillas del BrowseAction como en el ReportAction, instanciar un objeto de clase DownloadFileAction el cual descarga el archivo almacenado en la tupla de la BD.

- ExecuteExtraFunctionalityAction
  Esta clase ejecuta la ExtraFunctionality seleccionada por el usuario en la grilla del BrowseAction o en el ReportAction
- SQLAction (Clase abstracta)
  - ExecuteInsertAction
    La clase ExecuteInsertAction procesa los datos ingresados en el formulario creado por la clase InsertAction, prepara y ejecuta la sentencia "Insert" SQL para dar de alta una nueva tupla
  - ExecuteUpdateAction
    La clase ExecuteUpdateAction procesa los datos ingresados en el formulario creado por la clase UpdateAction, prepara y ejecuta la sentencia "Update" SQL para actualizar la tupla
  - ExecuteDeleteAction
    La clase ExecuteDeleteAction prepara y ejecuta la sentencia "Delete" SQL para eliminar la tupla confirmada previamente en el formulario generado por la clase DeleteAction
  - ExecuteMultipleInsertAction
    Luego de seleccionar las filas que se desean asociar a la fila maestra, en el formulario generado por la clase MultipleInsertAction, se instancia un objeto de la clase ExecuteMultipleInsertAction, el cual se encarga de preparar y ejecutar las sentencias "Insert"SQL para asociar las filas seleccionadas.
  - ExecuteMultipleDeleteAction
    La clase ExecuteMultipleDeleteAction prepara y ejecuta las sentencias "Delete" SQL para eliminar aquellas filas seleccionadas en la grilla del BrowseAction.
  - ExecuteExtGridEditorUpdate
    La grilla del BrowseAction puede permitir editar su contenido. Cuando esto ocurre, se instancia un objeto de clase ExecuteExtGridEditorUpdate el cual prepara y ejecuta las sentencias SQL para actualizar los valores de la grilla
- ExportToCSVFileAction
  La clase ExportToCSVFileAction permite generar un archivo en formato CSV de todas las tuplas filtradas en la grilla del BrowseAction
- WriteJSONDataAction
  Las grillas del framework ExtJS visualizan la información de sus páginas mediante Ajax, leyendo del servidor los valores en formato JSON [Web13]. La clase WriteJSONDataAction genera estos valores.
- PrintPDFAction
  Esta clase genera un reporte en formato PDF de los datos de una tupla específica
- PrintTableDataInPDFAction
  La clase PrintTableDataInPDFAction permite generar un reporte en formato PDF de todas las tuplas filtradas en la grilla del BrowseAction
- ViewImageAction
  - ViewThumbnailSizeImageAction
    Cuando un repositorio cuenta con un campo de clase ImageType, PHP4DB genera mediante la clase ViewThumbnailSizeImageAction una imagen

pequeña a visualizar en las filas de las grillas del BrowseAction y en el ReportAction de una tupla especifica.

- ViewRealSizeImageAction

  La clase ViewRealSizeImageAction permite, a partir de la clase ViewThumbnailSizeImageAction, descargar y visualizar la imagen completa guardada en las tuplas de la tabla del repositorio.

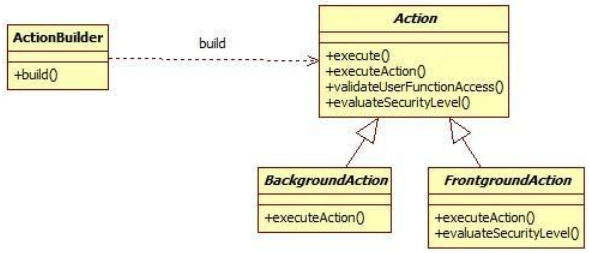


Ilustración 21: Diagrama de clases de Actions

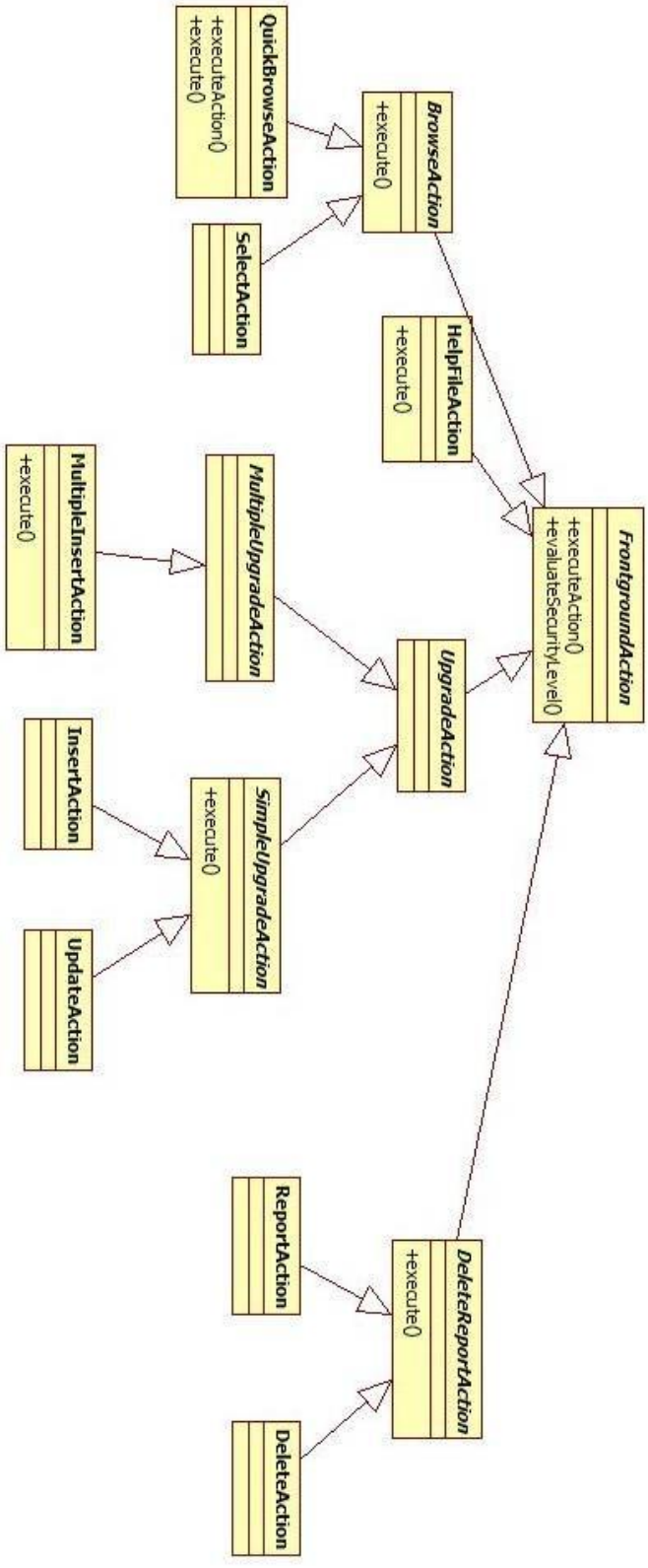


Ilustración 22: Subdiagrama de clases de FrontgroundAction

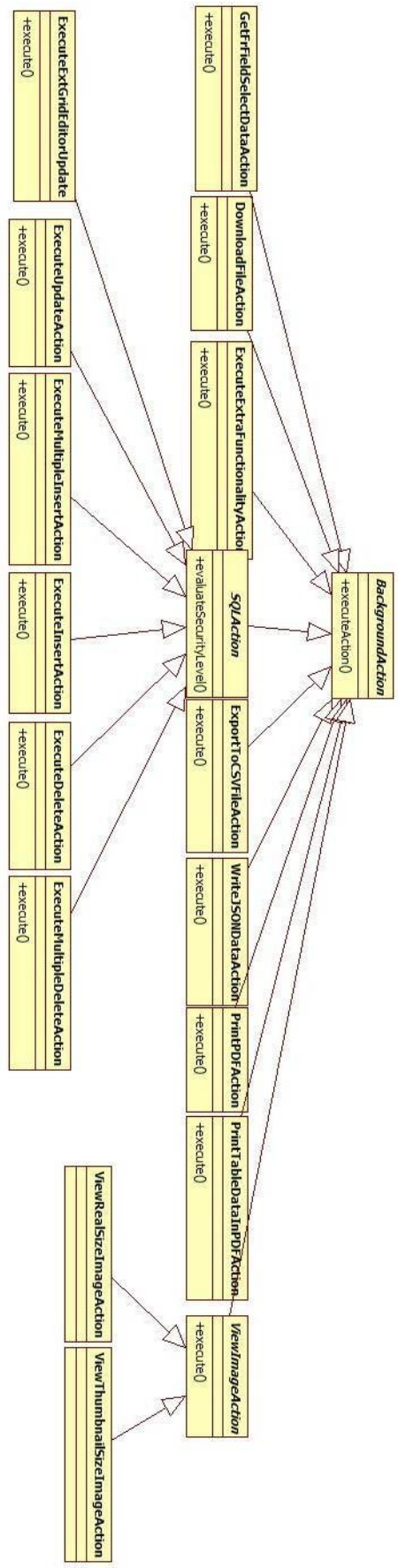


Ilustración 23: Subdiagrama de clases de BackgroundAction

5. Eventos

Toda funcionalidad básica de un repositorio puede ser resuelta automáticamente con solo definir la subclase Form asociada al mismo. Existen casos en que algunos repositorios necesitan reaccionar ante determinadas cambios de su estado interno.

PHP4DB permite de manera sencilla ejecutar trozos de código en respuesta a acciones o eventos (sucesos) que ocurren durante la ejecución de un repositorio. Por ejemplo, cuando se visualiza el formulario de alta de datos, el núcleo de PHP4DB detecta si se encuentra definido algún método asociado a dicho evento en el FDS que se encuentra en ejecución. En caso de existir, realiza la ejecución de dicho método, en su defecto continua trabajando normalmente. Ejemplos de estos eventos lo representan *before_execute_insert()* o *after_execute_insert()*, los que aumentan el nivel de acciones que se pueden realizar en la operatoria normal del framework.

La clase Form define los métodos para capturar los eventos. Estos métodos pueden ser redefinidos en sus subclases por los programadores. A continuación se presenta la lista de métodos de la clase Form que pueden ser redefinidos, una breve descripción de su alcance y la especificación de los parámetros que este recibe.

Form::before_execute_delete()
Se ejecuta antes de dar de baja la tupla del repositorio.

Form::after_execute_delete()
Se ejecuta luego de dar de baja la tupla del repositorio.
@param boolean $qryResult indica si la baja fue exitosa
@return string mensaje de confirmación/error específico del evento

Form::before_execute_insert_update()
Se ejecuta antes de dar de alta o modificar la tupla.
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_EXECUTE_INSERT' o 'PHP4DB_EXECUTE_UPDATE')

Form::after_execute_insert_update()
Se ejecuta luego de dar de alta o modificar la tupla.
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_EXECUTE_INSERT' o 'PHP4DB_EXECUTE_UPDATE')
@param boolean $qryResult indica si la operación fue exitosa
@return string mensaje de confirmación/error específico del evento

Form::before_validate_insert_update()
Se ejecuta antes de validar si los valores de los campos del formulario son validos
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_VALIDATE_INSERT' o 'PHP4DB_VALIDATE_UPDATE')

Form::after_validate_insert_update()
Se ejecuta luego de validar si los valores de los campos del formulario son validos
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_VALIDATE_INSERT' o 'PHP4DB_VALIDATE_UPDATE')
@param boolean $valido indica si los valores de los campos son validos, y permite modificarlo

bajo demanda del programador
@param string $errWarning indica el msg de error a visualizar en el formulario (en caso de error) y permite modificarlo

Form::after_execute_multipleinsert()
Se ejecuta luego de ejecutar cada insert de los n seleccionados en el MultipleInsertAction
@param boolean $qryResult indica si el insert se produjo correctamente
@param mixed $slaveValue identifica a la fila esclava que se quiere asociar

Form::before_prepare_query()
Se ejecuta antes de preparar el query a ejecutar que recupera los datos del repositorio de la bd

Form::after_prepare_query()
Se ejecuta luego de preparar el query a ejecutar que recupera los datos del repositorio de la bd

Form::before_print_browse_row()
Se ejecuta antes de visualizar cada fila en la grilla
@param array $fila valores de la fila y permite modificar sus datos

Form::before_create_master_title()
Se ejecuta en los formularios hijos antes de imprimir el titulo que identifica al padre
@param string $masterTitle titulo maestro el cual puede ser modificado por el programador

Form::before_execute_extraFunctionality()
Se invoca antes de ejecutar una funcionalidad Extra
@param mixed $idSelected clave que identifica a la fila seleccionada en la cual se desea ejecutar la funcionalidad extra
@param ExtraFunctionality $extraF funcionalidad extra seleccionada
@param boolean $continueExecution booleano en el cual el programador puede evitar que se continúe la ejecución de $extraF
@param string $msgWarning el programador pueda especificar un mensaje de error para cuando no se desea continuar la ejecución

Form::before_execute()
Se ejecuta antes de ejecutar la Action que le corresponde

Form::before_execute_query()
Se ejecuta antes de unir las partes del sql que recupera los datos del repositorio
@param string $select parte de la clausula select
@param string $join_on parte de la clausula join
@param string $where parte de la clausula where
@param string $orderBy parte de la clausula orderby

Form::before_show_functionality_for_record()
Se ejecuta antes de visualizar las funcionalidades extras para cada fila de la grilla
@param mixed $recordID valor que identifica la tupla
@param ExtraFunctionality $extraF funcionalidad extra a imprimir
@param array $params paramestros extra que se quiere asociar a la extra functionality
@param boolean $show permite indicar si la funcionalidad extra se tiene que mostrar o no

Form::before_create_form_filter()
Se ejecuta antes de crear el formulario de filtro de datos

Form::before_create_form_insert_update()
Se ejecuta antes de crear el formulario de insert o update
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_FORM_INSERT' o 'PHP4DB_FORM_UPDATE')

Form::before_create_view_delete_report()
Se ejecuta antes de crear la vista de una tupla o la vista previa a eliminar una tupla
@param string $eventLabel etiqueta que identifica la operación ('PHP4DB_VIEW_DELETE' o 'PHP4DB_FORM_REPORT')

Form::before_show_form_insert_update()
Se ejecuta antes de cargar los datos a los campos del formulario de insert o update
@param array $fila arreglo con los datos a cargar al formulario

6. Ejemplos

En este capítulo se ejemplifican posibles usos del framework. Para ello, se propone la realización de una aplicación web que permita administrar las tesis de grado de la Universidad Nacional de La Plata. Esta aplicación debe registrar la información relacionada a los trabajos de tesis: los temas que son referenciados, las publicaciones generadas, la bibliografía utilizada, los tutores y autores intervinientes, y el seguimiento de trámites de los alumnos, desde que se realizó la presentación de la propuesta hasta el momento de la exposición de la misma, registrando el resultado obtenido. Como se puede observar, el sistema propuesto no cuenta con una alta complejidad, pero permitirá ilustrar de manera detallada las ventajas obtenidas al utilizar PHP4DB en el desarrollo de aplicaciones web. La Ilustración 24 presenta el modelo de datos que soporta el sistema de administración de tesis de grado.

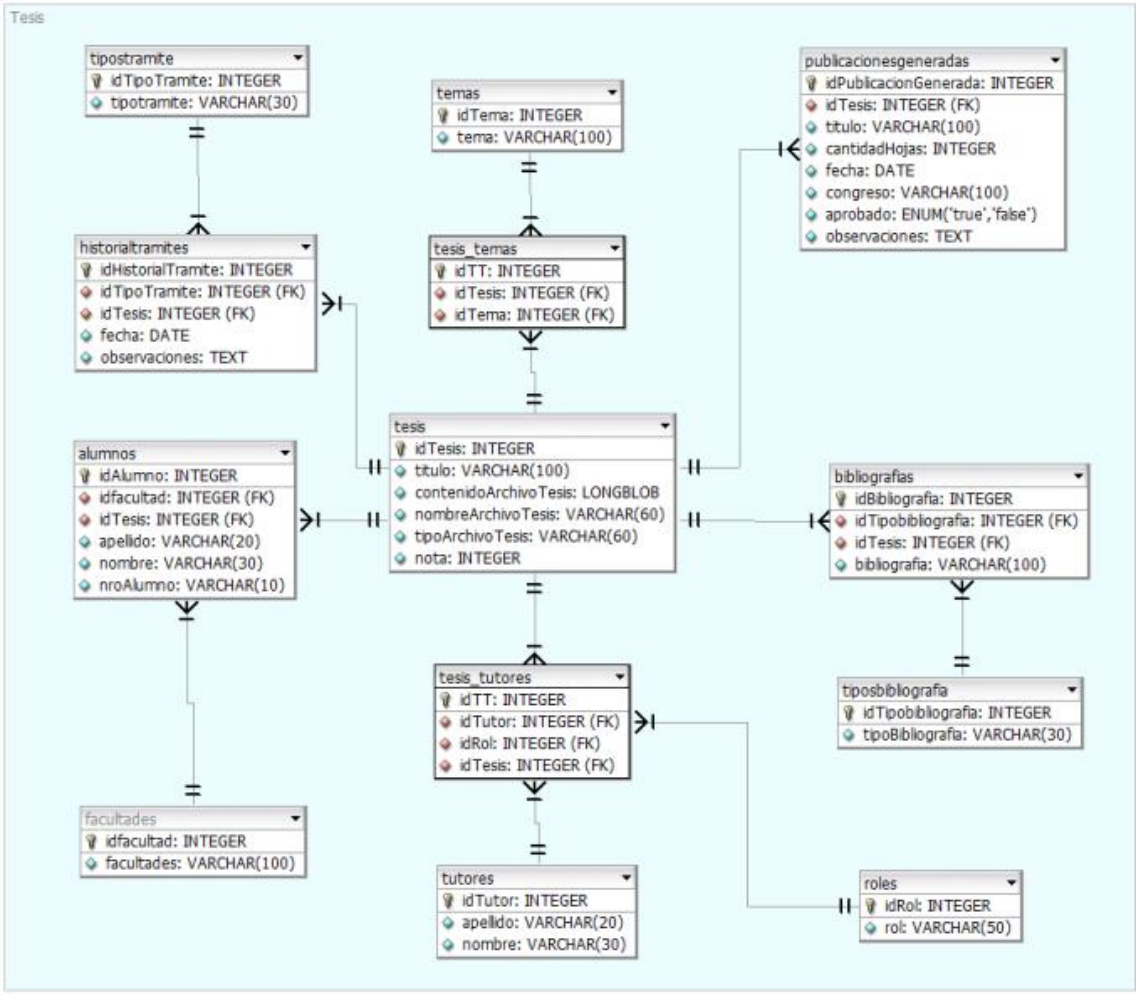


Ilustración 24: Modelo de datos sistema de administración de tesis de grado

En el desarrollo de aplicaciones, resulta más conveniente comenzar con aquellos módulos que referencian las tablas de dominio de la aplicación y luego aumentar la complejidad y desarrollar los módulos principales. A continuación se presenta el módulo para administrar las

facultades y otro para administrar los roles de los tutores de tesis (ej: Director, Co-Director). Los módulos para administrar los tipos de bibliografía, los temas de tesis, los tutores de los alumnos y los posibles tipos de trámites que una tesis puede generar, se deben desarrollar de igual manera, quedando fuera del alcance de este ejemplo.

La clase FacultyForm (Figura 1) extiende a la clase Form, nativa de PHP4DB. La clase concreta configura el modulo, indicando los títulos a utilizar, la tabla de la base de datos referenciada y sus campos. En el capítulo 4 se mencionó que el framework cumple un nivel C2 de seguridad, mediante un esquema de acceso por perfiles. En este repositorio se debe verificar que la función con código "CONF_5" esté asociada a alguno de los perfiles del usuario que intente operar con el modulo.

```
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class FacultyForm extends Form{

        function initialize(){

                parent::initialize();

                $this -> setDomainTable( "facultades" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "CONF_5");

                $entidad = "facultad";
                $entidades = "facultades";

                $this -> setTitle("browse", "Listado de $entidades" );
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle("report", "Visualización de $entidad" );
                $this -> setTitle("update", "Actualización de $entidad" );
                $this -> setTitle("insert", "Alta de $entidad" );
                $this -> setTitle("executeDelete", "Baja de $entidad exitosa" );
                $this -> setTitle("executeInsert", "Alta de $entidad exitosa" );
                $this -> setTitle("executeUpdate", "Actualización de
                                                    $entidad exitosa" );

                $aField = new AutoIncType();
                $aField -> setFieldName( "idfacultad" );
                $aField -> setFieldLabel( "idfacultad" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "facultad" );
                $aField -> setFieldLabel( "Facultad" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

        }
}

$aForm = new FacultyForm();
$aForm -> execute();

?>
```

Figura 1: Clase FacultyForm

De forma casi idéntica la Figura 2 muestra la clase RoleForm que administra los roles de los tutores.

```php
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class RoleForm extends Form{

        function initialize(){

                parent::initialize();

                $this -> setDomainTable( "roles" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "CONF_4" );

                $entidad = "rol";
                $entidades = "roles";

                $this -> setTitle( "browse", "Listado de $entidades");
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeDelete", "Baja de $entidad exitosa");
                $this -> setTitle( "executeInsert", "Alta de $entidad exitosa");
                $this -> setTitle( "executeUpdate","Actualización de
                                                    $entidad exitosa");

                $aField = new AutoIncType();
                $aField -> setFieldName( "idRol" );
                $aField -> setFieldLabel( "idRol" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "rol" );
                $aField -> setFieldLabel( "Rol" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

        }

}

$aForm = new RoleForm();
$aForm -> execute();

?>
```

Figura 2: Clase RoleForm

La próxima tarea es implementar el repositorio para administrar tesis, para luego poder implementar los repositorios de las entidades relacionadas.

La clase ThesisForm (Figura 3) además de contar con la información de los campos que componen la tabla tesis, cuenta con la configuración de las operaciones aplicables a esta:

- Relacionar una tesis con sus autores

- Relacionar una tesis con sus directores
- Clasificar las tesis en temas posibles de tesis
- Administrar las publicaciones generadas por las tesis
- Administrar los tramites efectuados durante el desarrollo de las tesis
- Administrar la bibliografía utilizada en las tesis

Estas operaciones se representan con objetos de la clase *ExtraFunctionality* explicada en la sección Relación entre repositorios del capítulo Implementación de PHP4DB.

```
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class ThesisForm extends Form{

        function initialize(){
                parent::initialize();

                $this -> setDomainTable( "tesis" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "TES_1" );

                $entidad = "tesis";
                $entidades = "tesis";

                $this -> setTitle( "browse", "Listado de $entidades");
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeDelete", "Baja de $entidad exitosa");
                $this -> setTitle( "executeInsert", "Alta de $entidad exitosa");
                $this -> setTitle( "executeUpdate","Actualización de
                                                    $entidad exitosa");

                $aField = new AutoIncType();
                $aField -> setFieldName( "idTesis" );
                $aField -> setFieldLabel( "idTesis" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "titulo" );
                $aField -> setFieldLabel( "Título" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( true );

                $this -> addComponent( $aField );

                $aField = new FileType();
                $aField -> setFieldName( "contenidoArchivoTesis" );
                $aField -> setFieldLabel( "Archivo" );
                $aField -> setFileNameField( 'nombreArchivoTesis' );
                $aField -> setFileTypeField( 'tipoArchivoTesis' );
                $aField -> setBrowseable( true );
                $aField -> setNull( true );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
```

```
// Continúa…

$aField = new IntegerType();
$aField -> setFieldName( "nota" );
$aField -> setFieldLabel( "Nota" );
$aField -> setBrowseable( true );
$aField -> setNull( true );
$aField -> setFiltered( false );

$this -> addComponent( $aField );


/**
 * Se agregan las funcionalidades extra de una tesis
 **/

$xf = new ExtraFunctionality("ALU");
$xf -> setLabel( "Autores de tesis" );
$xf -> setSrc( "alumnos.php" );
$xf -> setExpandible( true );
$xf -> setMaster( true );
$xf -> addView(
       new ComboExtraFunctionalityView("../images/user_gray.png")
);
$xf -> addView(
       new ImageExtraFunctionalityView("../images/user_gray.png")
);

$this -> addExtraFunctionality( $xf );

$xf = new ExtraFunctionality("TUT");
$xf -> setLabel( "Tutores de tesis" );
$xf -> setSrc( "tesis_tutores.php" );
$xf -> setExpandible( true );
$xf -> setMaster( true );
$xf -> addView( new ComboExtraFunctionalityView() );

$this -> addExtraFunctionality( $xf );

$xf = new ExtraFunctionality("TRA");
$xf -> setLabel( "Historial de trámites de tesis" );
$xf -> setSrc( "historialTramites.php" );
$xf -> setExpandible( true );
$xf -> setMaster( true );
$xf -> addView( new ComboExtraFunctionalityView() );

$this -> addExtraFunctionality( $xf );

$xf = new ExtraFunctionality("TEM");
$xf -> setLabel( "Temas de tesis" );
$xf -> setSrc( "tesis_temas.php" );
$xf -> setMaster( true );
$xf -> addView( new ComboExtraFunctionalityView() );

$this -> addExtraFunctionality( $xf );

$xf = new ExtraFunctionality("BIB");
$xf -> setLabel( "Bibliografía de tesis" );
$xf -> setSrc( "bibliografias.php" );
$xf -> setMaster( true );
$xf -> addView( new ComboExtraFunctionalityView() );

$this -> addExtraFunctionality( $xf );

$xf = new ExtraFunctionality("Pub");
$xf -> setLabel( "Publicaciones generadas con la tesis" );
$xf -> setSrc( "publicacionesgeneradas.php" );
$xf -> setMaster( true );
$xf -> addView( new ComboExtraFunctionalityView() );

$this -> addExtraFunctionality( $xf );
```

```
        } // End initialize

} // End class block

        $aForm = new ThesisForm();
        $aForm -> execute();
?>
```

Figura 3: Clase ThesisForm

Creado el repositorio para administrar tesis, resta implementar aquellos módulos ligados a las entidades que se relacionan con la entidad tesis. La Figura 4 visualiza la clase StudentsForm la cual representa el repositorio para relacionar una tesis con los alumnos que la escriben. Este repositorio, a diferencia de los anteriores, no es independiente ya que depende fuertemente del repositorio de tesis. Esta clase de repositorios dependientes, requieren de un campo maestro que relacione el repositorio con el repositorio padre. En la clase StudentsForm el campo maestro es idTesis, el cual filtra los autores de la tesis seleccionada en ThesisForm.

```
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class StudentsForm extends Form{

        function initialize(){
                parent::initialize();

                $this -> setDomainTable( "alumnos" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "TES_2" );
                $this -> setReturnUrl( 'tesis.php' );

                $entidad = "alumno";
                $entidades = "alumnos";

                $this -> setTitle( "browse", "Listado de $entidades" );
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeUpdate","Actualización de
                                                        $entidad exitosa");
                $this -> setTitle( "executeDelete","Baja de $entidad exitosa" );
                $this -> setTitle( "executeInsert","Alta de $entidad exitosa" );

                $aField = new AutoIncType();
                $aField -> setFieldName( "idAlumno" );
                $aField -> setFieldLabel( "idAlumno" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new FrKeyType();
                $aField -> setFieldName( "idfacultad" );
                $aField -> setFieldLabel( "Facultad" );
                $aField -> setFrKeyFieldName( "facultades" );
                $aField -> setFrIndex( "idfacultad" );
                $aField -> setFrTable( "facultades" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
```

```
                // Continúa…

                /**
                        El campo idTesis actúa como campo maestro,
                        ligándose con la tesis seleccionada en ThesisForm
                **/

                $aField = new FrKeyType();
                $aField -> setFieldName( "idTesis" );
                $aField -> setFieldLabel( "Tesis" );
                $aField -> setFrKeyFieldName( "titulo" );
                $aField -> setFrIndex( "idTesis" );
                $aField -> setFrTable( "tesis" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setMaster( true );

                $this -> addComponent( $aField );


                $aField = new TextType();
                $aField -> setFieldName( "apellido" );
                $aField -> setFieldLabel( "Apellido" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "nombre" );
                $aField -> setFieldLabel( "Nombre" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "nroAlumno" );
                $aField -> setFieldLabel( "Nro. de Alumno" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
        }
}

        $aForm = new StudentsForm();
        $aForm -> execute();
?>
```

Figura 4: Clase StudentsForm

El siguiente módulo desarrollado permite relacionar una tesis con sus directores. Previamente se debería contar con el modulo para administrar tutores/directores de manera similar a la clase FacultyForm (Figura 1) o RoleForm (Figura 2). Disponiendo de tutores cargados en el sistema es posible indicar quienes intervienen en una tesis determinada. A continuación se visualiza la clase ThesisTutorForm (Figura 5) la cual representa el repositorio para relacionar una tesis con los tutores que la dirigen. Este repositorio, al igual que el de la clase StudentsForm (Figura 4), es fuertemente dependiente del repositorio de tesis y requiere del campo maestro para relacionar ambos repositorios. En la clase ThesisTutorForm el campo maestro es idTesis, al igual que en StudentsForm. El campo idTutor es representado mediante la clase NFrKeyType, ya que en los formularios es más cómodo para el operador del sistema

seleccionar el tutor desde una ventana de selección aparte, donde se puede mostrar más información de los tutores junto a la posibilidad de poder aplicar filtros. Para estos tipos de casos los FrKeyType no son suficientes, ya que elegir un tutor de una lista desplegable puede no ser lo más natural.

```php
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class ThesisTutorForm extends Form{

        function initialize(){
                parent::initialize();

                $this -> setDomainTable( "tesis_tutores" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "TES_6" );
                $this -> setReturnUrl( 'tesis.php' );

                $entidad = "tutor de tesis";
                $entidades = "tutores de la tesis";

                $this -> setTitle( "browse", "Listado de $entidades" );
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeUpdate","Actualización de
                                                        $entidad exitosa");
                $this -> setTitle( "executeDelete","Baja de $entidad exitosa" );
                $this -> setTitle( "executeInsert","Alta de $entidad exitosa" );

                $aField = new AutoIncType();
                $aField -> setFieldName( "idTT" );
                $aField -> setFieldLabel( "idTT" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new FrKeyType();
                $aField -> setFieldName( "idTesis" );
                $aField -> setFieldLabel( "Tesis" );
                $aField -> setFrKeyFieldName( "titulo" );
                $aField -> setFrIndex( "idTesis" );
                $aField -> setFrTable( "tesis" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setMaster( true );
                /** idTesis actua como campo maestro,
                        ligandose con la tesis seleccionada en ThesisForm **/

                $this -> addComponent( $aField );

                $aField = new NFrKeyType();
                $aField -> setFieldName( "idTutor" );
                $aField -> setFieldLabel( "Tutor" );
                $aField -> setNFrFields( array( "apellido" => "Apellido",
                                                "nombre" => "Nombre"));
                $aField -> setFrIndex( array("idTutor") );
                $aField -> setFrTable( "tutores" );
                $aField -> setFrSrc( "tutores.php" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
```

```
                // Continúa…


                $aField = new FrKeyType();
                $aField -> setFieldName( "idRol" );
                $aField -> setFieldLabel( "Rol" );
                $aField -> setFrKeyFieldName( "rol" );
                $aField -> setFrIndex( "idRol" );
                $aField -> setFrTable( "roles" );
                $aField -> setFrSrc( "../configuracion/roles.php" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
        }

}

        $aForm = new ThesisTutorForm();
        $aForm -> execute();
?>
```

Figura 5: Clase ThesisTutorForm

El siguiente módulo implementado permite clasificar una tesis en temas. Esto se ve representado mediante la clase ThesisTopicForm (Figura 6). Al igual que las clases StudentsForm y ThesisTutorForm el campo idTesis actúa como campo maestro relacionando los temas de tesis con la tesis. El funcionamiento de este repositorio difiere de los anteriores en cómo se carga de la información: se necesita poder seleccionar múltiples temas de tesis y asociarlos a la tesis en curso ágilmente. Es claro que esta asociación es diferente a la alta de datos convencional presente en los repositorios anteriores. Debido a esto, es que se indica *$this -> setActionIndexValue('insert','style', 'multipleInsertStyle')* y se establece al campo idTema como “esclavo”, con el fin de dar al formulario el comportamiento de asociación en lugar del de simple carga de datos.

```php
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class ThesisTopicForm extends Form{

	function initialize(){
		parent::initialize();

		$this -> setDomainTable( "tesis_temas" );
		$this -> setLimit( 10 );
		$this -> setUserFunctionCode( "TES_5" );
		$this -> setReturnUrl( 'tesis.php' );

		$entidad = "tema de tesis";
		$entidades = "temas de la tesis";

		$this -> setTitle( "browse", "Listado de $entidades" );
		$this -> setTitle("fdelete","Confirmación de baja de $entidad");
		$this -> setTitle( "report", "Visualización de $entidad" );
		$this -> setTitle( "update", "Actualización de $entidad" );
		$this -> setTitle( "insert", "Alta de $entidad" );
		$this -> setTitle( "executeUpdate","Actualización de
                                                    $entidad exitosa");
		$this -> setTitle( "executeDelete","Baja de $entidad exitosa" );
		$this -> setTitle( "executeInsert","Alta de $entidad exitosa" );

		$aField = new AutoIncType();
		$aField -> setFieldName( "idTT" );
		$aField -> setFieldLabel( "idTT" );
		$aField -> setBrowseable( false );
		$aField -> setNull( false );
		$aField -> setFiltered( false );
		$aField -> setPrimaryKey( true );

		$this -> addComponent( $aField );

		$aField = new FrKeyType();
		$aField -> setFieldName( "idTesis" );
		$aField -> setFieldLabel( "Tesis" );
		$aField -> setFrKeyFieldName( "titulo" );
		$aField -> setFrIndex( "idTesis" );
		$aField -> setFrTable( "tesis" );
		$aField -> setBrowseable( false );
		$aField -> setNull( false );
		$aField -> setFiltered( false );
		$aField -> setMaster( true );
		/** idTesis actua como campo maestro,
			ligandose con la tesis seleccionada en ThesisForm **/

		$this -> addComponent( $aField );

		$aField = new FrKeyType();
		$aField -> setFieldName( "idTema" );
		$aField -> setFieldLabel( "Tema" );
		$aField -> setFrKeyFieldName( "tema" );
		$aField -> setFrIndex( "idTema" );
		$aField -> setFrTable( "temas" );
		$aField -> setBrowseable( true );
		$aField -> setNull( false );
		$aField -> setFiltered( false );
		$aField -> setSlave( true );

		$this -> addComponent( $aField );

		// Se desea poder asociar tuplas esclavas con el valor maestro
		$this -> setActionIndexValue('insert','style', 'multipleInsertStyle');
	}
}

	$aForm = new ThesisTopicForm();
	$aForm -> execute();
?>
```

Figura 6: Clase ThesisTopicForm

Otra de las operaciones aplicables a una tesis consiste en mantener el control de las publicaciones que la misma genera. La Figura 7 muestra la clase GeneratedPublicationsForm y ejemplifica la diversidad de tipos de campos que un repositorio puede tener:

- textos cortos (TextType) para el titulo de la publicación y nombre del congreso donde fue publicada
- valores enteros (IntegerType) para la cantidad de hojas de la publicación
- campos fecha (DateType) para la fecha de la publicación
- campos booleanos (BooleanType) para indicar si fue aprobado o no
- textos extensos (MemoType) para indicar observaciones sobre la publicación

La relación entre las publicaciones generadas y la tesis, como es de esperar, está dada por el campo maestro idTesis.

```
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class GeneratedPublicationsForm extends Form{

        function initialize(){
                parent::initialize();

                $this -> setDomainTable( "publicacionesgeneradas" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "TES_7" );
                $this -> setReturnUrl( 'tesis.php' );

                $entidad = "publicacion generada con la tesis";
                $entidades = "publicaciones generadas con la tesis";

                $this -> setTitle( "browse", "Listado de $entidades" );
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeUpdate","Actualización de
                                                        $entidad exitosa");
                $this -> setTitle( "executeDelete","Baja de $entidad exitosa" );
                $this -> setTitle( "executeInsert","Alta de $entidad exitosa" );

                $aField = new AutoIncType();
                $aField -> setFieldName( "idPublicacionGenerada" );
                $aField -> setFieldLabel( "idPublicacionGenerada" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new FrKeyType();
                $aField -> setFieldName( "idTesis" );
                $aField -> setFieldLabel( "Tesis" );
                $aField -> setFrKeyFieldName( "titulo" );
                $aField -> setFrIndex( "idTesis" );
                $aField -> setFrTable( "tesis" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setMaster( true );
                /** idTesis actua como campo maestro,
                        ligandose con la tesis seleccionada en ThesisForm **/


                $this -> addComponent( $aField );
```

```
                // Continúa…

                $aField = new TextType();
                $aField -> setFieldName( "titulo" );
                $aField -> setFieldLabel( "Título Publicación" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new IntegerType();
                $aField -> setFieldName( "cantidadHojas" );
                $aField -> setFieldLabel( "Cantidad de hojas" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new DateType();
                $aField -> setFieldName( "fecha" );
                $aField -> setFieldLabel( "Fecha" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new TextType();
                $aField -> setFieldName( "congreso" );
                $aField -> setFieldLabel( "Congreso" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new BooleanType();
                $aField -> setFieldName( "aprobado" );
                $aField -> setFieldLabel( "Aprobado" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new MemoType();
                $aField -> setFieldName( "observaciones" );
                $aField -> setFieldLabel( "Observaciones" );
                $aField -> setBrowseable( true );
                $aField -> setNull( true );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
        }
}

        $aForm = new GeneratedPublicationsForm();
        $aForm -> execute();
?>
```

Figura 7: Clase GeneratedPublicationsForm

El desarrollo de una tesis de grado lleva a los alumnos a generar una serie de trámites en su casa de estudio: presentación de propuesta de tesis, informes de avances, pedido de prórroga de tiempo, entrega final, etc. En la aplicación web de ejemplo se desea poder llevar el histórico

de trámites de las tesis. A continuación se presenta la clase HistoryStepForm (Figura 8) que permite cargar trámites a una tesis seleccionada.

Por su parte, el módulo que permite ingresar la bibliografía utilizada en una tesis debe ser implementado de similar manera que la clase HistoryStepForm quedando fuera del alcance de este ejemplo.

```
<?php

include ("../PHP4DB/PHP4DB-processing.php");

Class HistoryStepForm extends Form{

        function initialize(){
                parent::initialize();

                $this -> setDomainTable( "historialTramites" );
                $this -> setLimit( 10 );
                $this -> setUserFunctionCode( "TES_8" );
                $this -> setReturnUrl( 'tesis.php' );


                $entidad = "historial de trámites de tesis";
                $entidades = "historiales de trámites de la tesis";

                $this -> setTitle( "browse", "Listado de $entidades" );
                $this -> setTitle("fdelete","Confirmación de baja de $entidad");
                $this -> setTitle( "report", "Visualización de $entidad" );
                $this -> setTitle( "update", "Actualización de $entidad" );
                $this -> setTitle( "insert", "Alta de $entidad" );
                $this -> setTitle( "executeUpdate","Actualización de
                                                        $entidad exitosa");
                $this -> setTitle( "executeDelete","Baja de $entidad exitosa" );
                $this -> setTitle( "executeInsert","Alta de $entidad exitosa" );

                $aField = new AutoIncType();
                $aField -> setFieldName( "idHistorialTramite" );
                $aField -> setFieldLabel( "idHistorialTramite" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setPrimaryKey( true );

                $this -> addComponent( $aField );

                $aField = new FrKeyType();
                $aField -> setFieldName( "idTesis" );
                $aField -> setFieldLabel( "Tesis" );
                $aField -> setFrKeyFieldName( "titulo" );
                $aField -> setFrIndex( "idTesis" );
                $aField -> setFrTable( "tesis" );
                $aField -> setBrowseable( false );
                $aField -> setNull( false );
                $aField -> setFiltered( false );
                $aField -> setMaster( true );
                /** idTesis actua como campo maestro,
                        ligandose con la tesis seleccionada en ThesisForm **/

                $this -> addComponent( $aField );

                $aField = new DateType();
                $aField -> setFieldName( "fecha" );
                $aField -> setFieldLabel( "Fecha" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
```

```
                // Continúa…

                $aField = new FrKeyType();
                $aField -> setFieldName( "idTipoTramite" );
                $aField -> setFieldLabel( "Tipo tramite" );
                $aField -> setFrKeyFieldName( "tipoTramite" );
                $aField -> setFrIndex( "idTipoTramite" );
                $aField -> setFrTable( "tipostramite" );
                $aField -> setFrSrc( "../configuracion/tipostramite.php" );
                $aField -> setBrowseable( true );
                $aField -> setNull( false );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );

                $aField = new MemoType();
                $aField -> setFieldName( "observaciones" );
                $aField -> setFieldLabel( "Observaciones" );
                $aField -> setBrowseable( true );
                $aField -> setNull( true );
                $aField -> setFiltered( false );

                $this -> addComponent( $aField );
        }
}

        $aForm = new HistoryStepForm();
        $aForm -> execute();
?>
```

Figura 8: Clase HistoryStepForm

Se ha ejemplificado la creación de repositorios, campos y la relación de un repositorio padre con otro hijo. Resta ejemplificar la utilización de eventos en repositorios. Un posible requerimiento para la aplicación de ejemplo es que al momento de registrar el trámite "Entrega Final de Tesis" para una tesis, sea notificada la persona encargada de publicar las portadas en la cartelera de la facultad. Luego el encargado accederá a la aplicación web, descargará la tesis del repositorio de tesis, imprimirá la portada y la publicará en la cartelera. Este nuevo requerimiento genera dos cuestiones a tener en cuenta en el repositorio HistoryStepForm:

- Se debe validar, previamente a registrar el tramite "Entrega Final de Tesis", que se haya cargado el documento de la tesis en el repositorio de tesis, para que luego el encargado de imprimir la portada pueda recuperar el trabajo
- Si se registra el tramite "Entrega Final de Tesis" se debe enviar un mail al encargado

Estas cuestiones son ejemplificadas en la Figura 9, donde se vuelve a visualizar la clase HistoryStepForm pero, en esta ocasión, implementando los eventos que resuelven el nuevo problema.

A lo largo de este capítulo se presentaron diversos ejemplos de repositorios implementados mediante el framework PHP4DB para una simple aplicación web para la administración de tesinas de grado. Ejemplos de complejidad media y alta, junto a una completa documentación, quedan planteados en el capítulo Trabajo Futuro.

```
Class HistoryStepForm extends Form{

        function after_validate_insert_update( $eventLabel, &$valido, &$errWarning ){

                $rowTipoTramite = $this -> getConnection() -> queryRow(
                                        "SELECT tipotramite
                                        FROM tipostramite
                                        WHERE           idTipoTramite           =
                                        {$_POST['idTipoTramite']}"
                );

                /**
                 * Si el tipo de trámite es "Entrega Final de Tesis",
                 * hay que verificar si el documento de la tesis se encuentra cargado
                 * */

                if( $rowTipoTramite['tipotramite'] == "Entrega Final de Tesis"){

                        $rowContenidoTesis = $this -> getConnection() -> queryRow(
                                                "SELECT contenidoArchivoTesis
                                                FROM tesis
                                                WHERE idTesis = {$_POST['idTesis']}"
                        );

                        /**
                         * Si el documento no está cargado no se permite continuar
                         * procesando el formulario y se muestra el mensaje de error
                         * */
                        if( is_null( $rowContenidoTesis['contenidoArchivoTesis'] ) ){
                                $valido = false;
                                $errWarning = "Para registrar el tramite se debe
                                        haber cargado el trabajo en la respectiva tesis";
                        }
                }
        }

        function after_execute_insert_update( $eventLabel,&$qryResult ){

                $rowTipoTramite = $this -> getConnection() -> queryRow(
                                        "SELECT tipotramite
                                        FROM tipostramite
                                        WHERE idTipoTramite = {$_POST['idTipoTramite']}"
                );

                /**
                 * Si la operación es Insert y se ejecutó correctamente,
                 * y el tipo de trámite es "Entrega Final de Tesis"...
                 * Mando mail al encargado de publicar la portada en la cartelera
                 * */
                if ((($eventLabel == "PHP4DB_EXECUTE_INSERT") && $qryResult) &&
                ( $rowTipoTramite['tipotramite'] == "Entrega Final de Tesis")){

                        $rowContenidoTesis = $this -> getConnection() -> queryRow(
                                                "SELECT titulo
                                                FROM tesis
                                                WHERE idTesis = {$_POST['idTesis']}"
                        );

                        $mensaje = "La tesis titulada '{$rowContenidoTesis['titulo']}'
                                        está lista para ser publicada";
                        mail('encargadoTesis@info.unlp.edu.ar',
                                'Nueva tesis lista para publicar', $mensaje);
                }
        }

        function initialize(){
                /**
                 * Misma implementación que en Figura 8
                 * */
        }

}
```

Figura 9: Clase HistoryStepForm con eventos

7. Comparaciones con otros Frameworks

A continuación se visualiza una comparación (Tabla 1) en base a las funcionalidades provistas por los Frameworks populares de php [Web14]. Los atributos que se tomaron para realizar la comparación son:

- Versión de PHP: Algunos frameworks están pensados para ser utilizados tanto en PHP4 como en PHP5, otros admiten solo PHP5 donde se tiene una mejora profunda en cuanto al modelo de objetos y la posibilidad de manejar excepciones, entre otras.
- MVC: Indica si el framework cuenta con Soporte para construir aplicaciones Model-View-Controller.
- Abstracción DBMS: Indica si el framework soporta múltiples motores de base de datos sin tener que realizar cambio alguno.
- Mapeo objeto tupla: Indica si el framework soporta mapeo objeto-registro, generalmente implementando el patrón ActiveRecord.
- Templates: Indica si el framework cuenta con un motor de templates.
- Cacheo: Indica si el framework incluye un objeto para administrar la memoria cache u otra manera de administrar su contenido.
- Ajax: Indica si el framework incluye soporte para Ajax.
- Seguridad: Indica si el framework cuenta con un módulo para manejar la autenticación de usuarios y demás cuestiones relacionadas a la seguridad
- Otras utilidades: Indica si el framework cuenta con otros módulos de utilidades, como por ejemplo módulo de soporte para PDF.

	Versión PHP	MVC	Abstracción DBMS	Mapeo objeto tupla	Templates	Cacheo	Ajax	Seguridad	Otras Utilidades
Zend	5	Si	Si	No	Si	Si	No	Si	Si
CakePHP	4 5	Si	Si	Si	Si	Si	Si	Si	No
Symfony	5	Si	Si	Si	Si	Si	Si	Si	No
Seagull	4 5	Si	Si	No	Si	Si	No	Si	Si
WACT	4 5	Si	Si	No	Si	No	No	No	No
Prado	5	No	Si	No	Si	No	Si	Si	Si
PHP on TRAX	5	Si	Si	Si	No	No	Si	No	No
ZooP	4 5	Si	Si	No	Si	No	Si	Si	Si
eZ Components	5	No	Si	No	No	Si	No	No	Si
CodeIgniter	4 5	Si	No	No	Si	Si	No	No	Si
PHP4DB	5	No	Si	No	No	No	Si	Si	Si

Tabla 1: Comparación de frameworks php

Aun resta evolucionar el framework PHP4DB, cubriendo otras características frecuentes en otros. Muchas de éstas se plantean en el capítulo Trabajo Futuro. No obstante, es importante

señalar que no se tuvo como objetivo cubrir los aspectos de todos los frameworks, sino concentrarse en:

- minimizar los tiempos de desarrollo de software
- minimizar los tiempos de testeo de software
- automatizar tareas rutinarias, permitiendo implementar un proceso iterativo, donde el tiempo de entrega de productos funcionales sea mínimo
- actuar como herramienta de prototipación si el proyecto a desarrollar necesitase de dicha metodología.

Una manera interesante de evaluar el cumplimiento de los objetivos consiste en analizar cómo se realiza determinada operación mediante un framework y como lo hace el objeto de estudio PHP4DB. Se eligió desarrollar un simple blog mediante el framework CakePHP, uno de los más populares y completos, y luego compararlo con el desarrollo de un simple blog mediante PHP4DB.

Al ser CakePHP un framework MVC, es necesario crear las clases para el controlador (Figura 10), el modelo (Figura 11) y las distintas vistas: formulario de alta (Figura 12), edición (Figura 13), vista completa de un post (Figura 14) y listado de posts (Figura 15).

```php
<?php
class PostsController extends AppController {

	var $name = 'Posts';

	function index() {
		$this->set('posts', $this->Post->find('all'));
	}
	function view($id = null) {
		$this->Post->id = $id;
		$this->set('post', $this->Post->read());
	}

	function add() {
		if (!empty($this->data)) {
			if ($this->Post->save($this->data)) {
				$this->flash('Your post has been saved.', '/posts');
			}
		}
	}

	function delete($id) {
		$this->Post->del($id);
		$this->flash('The post with id: '.$id.' has been deleted.', '/posts');
	}

	function edit($id = null) {
		$this->Post->id = $id;
		if (empty($this->data)) {
			$this->data = $this->Post->read();
		} else {
			if ($this->Post->save($this->data)) {
				$this->flash('Your post has been updated.','/posts');
			}
		}
	}

}
?>
```

Figura 10: Controlador de Blog - controllers/postscontroller.php (CakePHP)

```php
<?php

class Post extends AppModel {

	var $name = 'Post';

	var $validate = array(
		'title' => array(
			'rule' => 'notEmpty'
		),
		'body' => array(
			'rule' => 'notEmpty'
		)
	);
}
?>
```

Figura 11: Modelo de Blog - models/posts.php (CakePHP)

```php
<h1>Add Post</h1>

<?php

echo $form->create('Post');

echo $form->input('title');

echo $form->input('body', array('rows' => '3'));

echo $form->end('Save Post');

?>
```

Figura 12: Vista de Alta de post de Blog - views/posts/add.ctp (CakePHP)

```php
<h1>Edit Post</h1>

<?php

	echo $form->create('Post', array('action' => 'edit'));

	echo $form->input('title');

	echo $form->input('body', array('rows' => '3'));

	echo $form->input('id', array('type'=>'hidden'));

	echo $form->end('Save Post');

?>
```

Figura 13: Vista de Edición de post de Blog - views/posts/edit.ctp (CakePHP)

```php
<h1><?php echo $post['Post']['title']?></h1>

<p><small>Created: <?php echo $post['Post']['created']?></small></p>

<p><?php echo $post['Post']['body']?></p>
```

Figura 14: Vista de completa de post de Blog - views/posts/view.ctp (CakePHP)

```
<h1>Blog posts</h1>

<?php echo $html->link('Add Post','/posts/add')?>

<table>

	<tr>

		<th>Id</th>

		<th>Title</th>

		<th>Actions</th>

		<th>Created</th>

	</tr>


	<!-- Here is where we loop through our $posts array, printing out post info -
->

	<?php foreach ($posts as $post): ?>

	<tr>

		<td><?php echo $post['Post']['id']; ?></td>

		<td>

			<?php echo $html->link($post['Post']['title'],
"/posts/view/".$post['Post']['id']); ?>

		</td>

		<td>

		<?php                    echo                    $html->link('Delete',
"/posts/delete/{$post['Post']['id']}", null, 'Are you sure?' )?>

		<?php echo $html->link('Edit', '/posts/edit/'.$post['Post']['id']);?>

		</td>


		<td><?php echo $post['Post']['created']; ?></td>

	</tr>

	<?php endforeach; ?>


</table>
```

Figura 15: Listado de post de Blog - views/posts/index.ctp (CakePHP)

La Figura 16 presenta el mismo módulo desarrollado mediante PHP4DB.

```php
<?

include ("../PHP4DB/PHP4DB-processing.php");

Class PostsForm extends Form{

	function initialize(){
		parent::initialize();

		$this -> setDomainTable( "posts" );
		$this -> setLimit( 10 );

		$entidad = "post";
		$entidades = "posts";

		$this -> setTitle( "browse", "Listado de $entidades" );
		$this -> setTitle( "fdelete", "Confirmación de baja de $entidad" );
		$this -> setTitle( "report", "Visualización de $entidad" );
		$this -> setTitle( "update", "Actualización de $entidad" );
		$this -> setTitle( "insert", "Alta de $entidad" );
		$this -> setTitle("executeUpdate", "Actualización de $entidad exitosa");
		$this -> setTitle( "executeDelete", "Baja de $entidad exitosa" );
		$this -> setTitle( "executeInsert", "Alta de $entidad exitosa" );

		$aField = new AutoIncType();
		$aField -> setFieldName( "id" );
		$aField -> setFieldLabel( "id" );
		$aField -> setBrowseable( false );
		$aField -> setNull( false );
		$aField -> setFiltered( false );
		$aField -> setPrimaryKey( true );

		$this -> addComponent( $aField );

		$aField = new TextType();
		$aField -> setFieldName( "title" );
		$aField -> setFieldLabel( "Título" );
		$aField -> setBrowseable( true );
		$aField -> setNull( false );
		$aField -> setFiltered( true );

		$this -> addComponent( $aField );

		$aField = new RichMemoType();
		$aField -> setFieldName( "body" );
		$aField -> setFieldLabel( "Mensaje" );
		$aField -> setBrowseable( false );
		$aField -> setNull( false );
		$aField -> setFiltered( false );

		$this -> addComponent( $aField );
	}

}

	$aForm = new PostsForm();
	$aForm -> execute();
?>
```

Figura 16: Clase PostsForm (PHP4DB)

Se puede observar que PHP4DB genera una menor cantidad de líneas de código concentrando todo el desarrollo en una única y simple clase, facilitando el desarrollo y posterior mantenimiento.

8. Resultados Obtenidos

Desde la definición del Framework, este ha sido utilizado en diversos proyectos, lo que ha permitido capturar experiencias, traducidas en la continua evolución de este producto. En líneas generales, los resultados obtenidos pueden catalogarse en varios grupos:

- **Mejora del tiempo de respuesta para implementar los requerimientos más elementales de un sistema de software.** En este punto se puede concluir que luego de la implantación del producto, se redujeron en un mínimo de 50% y hasta el 80% los tiempos requeridos para el desarrollo de la algorítmica clásica y básica de un Sistema de Software (altas, bajas, modificaciones, filtrados y listados generales).
- **Rápido aprendizaje en el uso de la herramienta.** En general, los programadores con experiencia en lenguajes web tienen una curva de aprendizaje muy rápida para lograr utilizar los aspectos generales y aún particulares de PHP4DB. Además, profesionales informáticos con poca experiencia en el desarrollo de sistemas web, particularmente utilizando PHP, lograron obtener resultados interesantes dentro de primera semana de utilización.
- **Alto nivel de confianza y reducción de los tiempos de prueba.** Al disponer de una herramienta estable, los desarrollos tradicionales obtenidos automáticamente no necesitaron pruebas exhaustivas del software, lo que produce mejoras en términos de calidad de desarrollo de software. Estas pruebas se limitan a evaluar con el usuario/cliente, la funcionalidad generada y compararla con la requerida.
- **Auditoría y control de cambios.** Es posible que el administrador del sistema defina de manera dinámica la información que será auditada por PHP4DB, como elemento adicional dentro del sistema desarrollado. Además, dentro de las atribuciones del administrador estará la definición de aquellos datos considerados indispensables para el sistema, sobre los cuales un usuario estándar con bajos permisos no podrá realizar determinadas operaciones.
- **Versatilidad para la incorporación de código particular por el programador.** De esta forma, es posible incorporar fácilmente la funcionalidad propia a cada sistema. Si bien las posibilidades de la herramienta está en continua evolución, siempre hay características puntuales que se necesita del software. PHP4DB está pensado para incorporar de una manera sencilla código específico.

Como se mencionó previamente, el Instituto de Investigación en Informática III-LIDI 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.

- **Área 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 Comparative Market Analysis (CMA) [Cas06].

- **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 pre asignado.
  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 [Sai05] [Del06a].
- **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 Red provincial de grupos de Investigación y desarrollo en áreas de Ciencia de la Computación e Informática (RedPIBA) 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 [Web12] [Pes06].
  Esta red provincial consta de 1300 nodos, sobre cada uno de los cuales se realiza auditoría 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.
- **Sistema de Planeamiento de la Producción.** Un sistema del tipo MRP-II proporciona la planificación y control eficaz de todos los recursos de la producción de una empresa.
  Un MRP-II implica la planificación de todos los elementos que se necesitan para llevar a cabo el plan maestro de producción, no sólo de los materiales a fabricar y vender, sino de las capacidades de fábrica en mano de obra y máquinas.
  Este sistema da respuesta a las preguntas, cuánto y cuándo se va a producir, y a cuáles son los recursos necesarios para ello.
  Los sistemas MRP-II han sido orientados principalmente hacia la identificación de los problemas de capacidad del plan de producción (disponibilidad de recursos frente al consumo planificado), facilitando la evaluación y ejecución de las modificaciones oportunas en el planificador. Para ello, a través del plan maestro de producción y de las simulaciones del comportamiento del sistema productivo de la empresa, se tendrá el control para potencialmente detectar y corregir las incidencias generadas de una manera ágil y rápida.
  Un sistema MRP-II ofrece una arquitectura de procesos de planificación, simulación, ejecución y control, cuyo principal cometido es cumplir con los objetivos de la producción de la manera más eficiente, ajustando las capacidades, la mano de obra, los inventarios, los costos y los plazos de producción.
  Este último sistema, Planeamiento de la Producción, presenta 40 CU en su documento de requerimientos, habiendo desarrollado 35 de ellos de manera automática con PHP4DB.

- **Relaciones Internacionales de la UNLP.** El objetivo es mantener información relacionada a la participación de los miembros de la UNLP en actividades con el exterior, y así fomentar el conocimiento, la colaboración y la responsabilidad de informar y registrar dichas participaciones. El sistema se divide en dos subsistemas: *Gestión de convenios;* administra la información relacionada con los convenios entre la Universidad (o alguno de sus miembros) e instituciones del exterior, *Gestión de características de las facultades;* administra información relacionada con las fortalezas y debilidades de las diferentes unidades académicas de la UNLP, además de las líneas de investigación/trabajo y la participación en diferentes redes internacionales.
  Los módulos del sistema fueron desarrollados en su totalidad mediante PHP4DB, sin tener que desarrollar módulos desde el inicio.
- **Sistema de Seguimientos de Expedientes.** El Sistema de Seguimiento de Expedientes ha sido diseñado para realizar el seguimiento interactivo de los Expedientes de la Facultad de Informática de la UNLP y sus respectivos pases de áreas. Permite almacenar un resumen del contenido de los Expedientes e Incidentes que se producen en la Universidad, como así también registrar el movimiento de los mismos por las distintas dependencias (Pases).
  El sistema define tres niveles de acceso: Administrativos del área responsable del registro de expedientes, administrativos de las áreas que realizan pases y autoridades de la Facultad.

Además, PHP4DB fue motivo de publicaciones en revistas [Del06a] [Del07] y en congresos con referato internacional [Del06b] [Del08a] [Del08b]. Así mismo, se obtuvo el registro de propiedad intelectual titulado *"Framework para el Desarrollo Ágil de Aplicaciones Web"* con fecha 11 de Junio de 2007 y fue identificado con el número de expediente 575396.

9. Trabajo Futuro

La línea de trabajo actual y futura tienen varias aristas:

• Soporte de Cache

Sea por el incremento de visitantes ó por el uso de scripts complejos, la ejecución de constantes consultas a la BD suele volver lento un servidor, con el consiguiente retraso a la hora de mostrar una página web.

Una de las mejores alternativas para solucionar este problema, es implementar un sistema de Cache con PHP. Con éste, en vez de realizar todas las consultas a la BD para servir una página, simplemente se requerirá un archivo de texto. El resultado, es un enorme ahorro de recursos y una más rápida respuesta del servidor.

• Soporte multi-idioma

Agregar una capa sobre PHP4DB que permita al sistema presentar sus interfaces en diferentes idiomas sin tener que duplicar su contenido. De esta manera las aplicaciones tendrán un alcance más global.

• Evolución hacia un PHP4DB con interfaz personalizable por el usuario final

Un objetivo planteado es permitir que la interfaz de interacción sea personalizable por el usuario final del producto con el fin de darle mayor versatilidad. Actualmente se está discutiendo las mejores opciones para llevar a cabo una solución integral a estas expectativas.

• Documentación completa

La documentación es una parte esencial de cualquier paquete de software. En los frameworks es deseable que la documentación sea clara y completa. No solo les resulta útil a los desarrolladores del framework, sino que también lo es para aquellos que lo utilizan para construir aplicaciones. Si la documentación está acompañada de ejemplos, más frutos dará. Dedicar el tiempo adecuado para documentar completamente el framework y ejemplificar aplicaciones de complejidad media y alta desarrolladas con PHP4DB son objetivos a cumplir inminentemente.

• Datos restringidos

Existen casos en que determinados grupos de usuarios pueden acceder a ciertas tuplas o columnas de ciertas tablas, pero no a todas. Por ejemplo, se podría desear que los empleados de determinada empresa no vean las facturas con montos superiores a cincuenta mil pesos. Caso contrario, podría suceder que los empleados puedan acceder a todas las facturas pero no tengan permiso para ver el monto de las mismas. Si bien estas cuestiones hoy pueden ser satisfechas en PHP4DB desde código, sería deseable que estas condiciones sean establecidas dinámicamente (obteniendo la información desde la BD) sin la necesidad de modificar el código del repositorio.

10. Conclusiones

La utilización del Framework PHP4DB en los proyectos definidos en el capítulo Resultados Obtenidos puede considerarse esencial. El motivo de esta aseveración se debe a que la herramienta permitió automatizar un gran porcentaje de los CU proveniente de la Ingeniería de Requerimientos, con el consiguiente aumento de la productividad final del equipo de desarrollo.

La Figura 17 presenta la relación entre los CU automatizados de algunos de los proyectos, sobre aquellos CU que necesitaron programación específica.

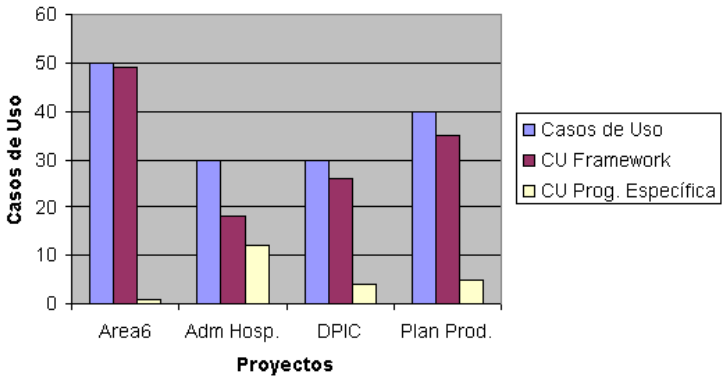


Figura 17: Comparación de CU implementados con y sin el Framework

A partir del análisis de la figura se puede observar los beneficios obtenido con PHP4DB. El tiempo de desarrollo se redujo al menos en un 50% (dependiendo de cada problema), con la consecuente satisfacción del usuario dada la temprana disponibilidad de los productos requeridos; logrando que la herramienta asista al desarrollo ágil de aplicaciones.

Un beneficio adicional obtenido con PHP4DB consistió en que las interfaces resultantes fueron homogéneas. Esto redunda en una satisfacción extra por parte del usuario/cliente y además facilitó el mantenimiento posterior.

El Framework permite acortar plazos en el ciclo de vida de desarrollo de software, no sólo en la fase de programación sino también en la de diseño y de prueba. Esto genera expectativas positivas para el desarrollo de aplicaciones que requieran ser orientadas a la web.

Un beneficio extra a partir de la utilización de PHP4DB tuvo que ver con el desarrollo de prototipos descartables. La herramienta fue probada con tal finalidad en reuniones de tipo BrainStorming con clientes reales, permitiendo generar prototipos demostrativos en los plazos típicos de este tipo de reuniones, 2 horas. Siguiendo en esta línea, en proyectos de gran envergadura, se utilizó el framework como herramienta de prototipado en reuniones tipo Joint Application Development (JAD), donde se requiere de prototipos para obtener y validar requerimientos esenciales.

11. Agradecimientos

A mis padres, por el apoyo incondicional que me brindaron a lo largo de mi carrera.

A Hugo Ramón y Rodolfo Bertone, por transmitirme y contagiarme la pasión por esta ciencia.

A la familia del III-LIDI, por permitirme crecer como persona y como profesional.

A Pablo Thomas y Germán Cáseres, y a todos aquellos que colaboraron o participaron desinteresadamente con el desarrollo de PHP4DB y, por consiguiente, con el desarrollo de esta tesina, hago extensivo mi más sincero agradecimiento.

A todos mis seres queridos que me ayudaron a culminar esta hermosa etapa de la vida.

12. Referencias

[Bat90] Diseño conceptual de Bases de Datos. Batini, Navathe Cieri. Addison Wesley. 1990

[Big96] Reusability framework, assessment, and directions. Biggerstaff Ted and Richter Charles. IEEE Software 4(2): 41-49, March 1987.

[Cas06] Use of asynchronous JavaScript and XML for Comparative Market Analysis, Caseres, Delía, Thomas, Ramón, Bertone, CACIC 2006, San Luis, Octubre de 2006.

[Cor02] Localización e internacionalización de sitios web. Noelia Corte MSc en Electronic Publishing (City University, Londres)

[Dav95] Principles of Software Development. A. Davis. Mc Graw Hill. 1995.

[Del06a] Framework for Agile Development: Its use in Web Applications for Hospitals, L. Delía, M. Iglesias, G. Cáseres , H. Ramón, P. Thomas, R. Bertone. EJIS (Eletronic Journal of Information Systems), Edition 09, Nº III 2006 special number on Information Systems on Health Care (human) ISSN 1677-3071, 2006, http://www.inf.ufsc.br/resi/

[Del06b] Framework para el Desarrollo Ágil de Aplicaciones Web, Lisandro Delía, Germán Cáseres, Hugo Ramón, Pablo Thomas, Rodolfo Bertone. III Workshop de Ingeniería de Software y Base de Datos - XII Congreso Argentino de Ciencias de la Computación CACIC'06. San Luis, Argentina. 17 al 21 de octubre de 2006. Pág. 289-299. ISBN: 950-609-050-5

[Del07] Framework for Web Application Agile Development, Lisandro Delía, Germán Cáseres, Hugo Ramón, Pablo Thomas, Rodolfo Bertone, Iberoamerican Science & Technology Education Consortium (ISTEC) Editorial Address – ISTEC Executive Offices – University of New Mexico – Albuquerque – USA. Journal of Computer Science & Technology Vol. 7 Nro 1 ISSN 1666-6046 – Pag. 86 – 90. Marzo 2007 http://journal.info.unlp.edu.ar/journal/

[Del08a] PHP4DB: Evolving Towards New Objectives for the Flexible Development of Web Applications, Lisandro Delía, Germán Cáseres, Hugo Ramón, Pablo Thomas, Rodolfo Bertone. JCC 2008. Jornadas Chilenas de Computación. Punta Arenas, Chile. 10-15 Noviembre, 2008. Encuentro Chileno de Computación. ISBN: 978-956-319-507-1

[Del08b] PHP4DB: Evolucionando hacia nuevos objetivos para el Desarrollo Ágil de Aplicaciones Web, Lisandro Delía, Germán Cáseres, Hugo Ramón, Pablo Thomas, Rodolfo Bertone. V Workshop de Ingeniería de Software y Base de Datos - XIV Congreso Argentino de Ciencias de la Computación CACIC'08. Chilecito – La Rioja, Argentina. 06 al 10 de octubre de 2008.

[Deu89] Design reuse and frameworks in the Smalltalk-80 programming system. Deutsch. In Software Reusability, volume II, pp. 55-71, Ted Biggerstaff and Alan Perlis, editors. Reading, MA: ACM Press/Addison Wesley, 1989

[Doe85] Orange Book. Department Of Defense. Library N° S225, 711. EEUU. 1985. http://www.doe.gov

[Dze04] Using Database Framework in Web Applications. Dzenan Ridjanovic and Vensada Okanovic. IEEE MELECON 2004, May 12-15,2004, Dubrovnik, Croatia

[Fay99] Building application frameworks: object-oriented foundations of framework design. Mohamed Fayad, Douglas Schmidt, Ralph Johnson. John Wiley & sons.

[Gam95] Design Patterns: Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Addison-Wesley Professional Computing Series.

[Gar05] Ajax: A New Approach to Web Applications. Jesse James Garret. 18 de Febrero de 2005. http://adaptivepath.com/ideas/essays/archives/000385.php

[Hum99] Introduction to the Team Software Process - Watts S Humphrey - Addison-Wesley Professional 1999

[Joh88] Designing reusable classes. Johnson R. E., Foote Brian. Journal of Object-Oriented Programming, 1(5) June/July. 1988: 22-35.

[Joh93] How to Design Frameworks. Johnson, R. E. (1993). Tutorial Notes for the 1993 Conference on Object Oriented Programming, Systems, Languages and Systems (OOPSLA '93).

[Lea00] Building Application Servers. Rick Leander. Cambridge University Press – 2000

[Liu00] Structural testing of Web applications. Chien-Hung Liu; Kung, D.C.; Pei Hsia; Chih-Tung Hsu. Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on 8-11 Oct. 2000 Page(s):84-96 Digital Object Identifier 10.1109/ISSRE.2000.885863

[Lou95] Loucopoulos, P., Karakostas, V., System Requirements Engineering, McGraw-Hill, 1995, London.

[Mar00] Understanding Object-Oriented Framework Engineering. Marcus E. Markiewicz, Carlos J. P. de Lucena. PUC-RioInf.MCC38/00 October, 2000

[Ols98] Developing User Interfaces - Dan R. Olsen. Morgan Kaufmann, 1998

[Pae07] Utilización de programación orientada a aspectos en aplicaciones enterprise. Nicolás Paez. Tesis de grado en Ingeniería en Informática. Noviembre 2007. Facultad de Ingeniería. Universidad de Buenos Aires

[Pes06] Sistemas de Software Distribuidos y Base de Datos Distribuidas. P. Pesado, H. Ramón, P. Thomas, M. Boracchia, R. Champredonde, A. Pasini, F. Chichizola, M. Iglesias, L. Marrero, M. B. Albanessi, L. Delía, G. Ricci. VIII Workshop de Investigadores en Ciencias de la Computación. WICC 2006. Morón- Buenos Aires, Argentina. 1 y 2 de Junio de 2006. ISBN: 950-9474-34-7 – Pág. 289-294

[Pre98] Ingeniería de Software. Un enfoque práctico. Roger Pressman. Mc Graw Hill. 1998.

[Ram05] Some experiments with the performance of LAMP architecture. Ramana, U.V.; Prabhakar, T.V. Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on 21-23 Sept. 2005 Page(s):916 - 920 Digital Object Identifier 10.1109/CIT.2005.169

[Ric04] Analysis, testing and re-structuring of Web applications. Ricca, F.; Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on 11-14 Sept. 2004 Page(s):474 - 478 Digital Object Identifier 10.1109/ICSM.2004.1357838

[Rod03] Programación de aplicaciones WEB. Rodriguez de la Puente Santiago, Carretero Perez Jesus, Perez Costoya Fernando. Paraninfo, 2003.

[Sai05] SAIH – Documento de Requerimientos. Hospital Felipe Pelaez. Municipalidad de Florentino Ameghino. IEEE 830. Instituto de Investigación en Informática. 2005, saih-lidi.info.unlp.edu.ar

[Sam01] Software Engineering With Reusable Components - Johannes Sametinger. Springer 2001

[Sha06] Taxonomy of Java Web Application Frameworks. Shan, T.C.; Hua, W.W.; e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on Oct. 2006 Page(s):378 - 385 Digital Object Identifier 10.1109/ICEBE.2006.98

[Som05] Ingeniería de software. 7ma edición. Ian Sommerville. Addison Wesley 2005.

[Tor04] Overlooked Aspects of COTS-Bases Development. M. Torchiano, M. Morisio., IEEE Software, 2004

[Wal04] Generación Automática de Código a partir del modelo de datos. M. Walsamakis, M. Mansutti, R. Bertone, R. Champredonde . CACIC2004. La Matanza. Octubre 2004

[Web1] http://www.agile-spain.com/agilev2/principios_agiles

[Web2] http://alxplus.blogspot.com/2006/08/aplicaciones-web-vs.html

[Web3] http://es.wikipedia.org/wiki/Framework

[Web4] http://www.zend.com/

[Web5]www.symfony-project.org/

[Web6]http://www.kumbiaphp.com/

[Web7] http://www.rubyonrails.org/

[Web8] http://cakephp.org/

[Web9] http://www.extjs.com/

[Web10] http://pear.php.net/package/MDB2

[Web11] http://pear.php.net/

[Web12] http://www.dpic.sg.gba.gov.ar http://dpic-lidi.info.unlp.edu.ar

[Web13] http://www.json.org/

[Web14] http://www.phpframeworks.com/

[Wri06] Agile language development: the next generation. Wright, W.; Moore, D.. Aerospace Conference, 2006 IEEE 4-11 March 2006 Page(s):6 pp. Digital Object Identifier 10.1109/AERO.2006.1656063