# Framework for Web Application Agile Development

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

**Institute of Research on Computer Science LIDI (III-LIDI) [6]**
**Faculty of Computer Science - UNLP**

**ABSTRACT**

Any system interacting with a data base requires modules capable of operating data stored in it. Its development times generally vary between 50 to 60 % of the time used for the application life cycle. The present paper describes the architecture and characteristics of a Framework for the agile generation of Web Applications, called PHP4DB. Its main objectives are to drastically reduce the job time, minimize errors, and tuning, as well as respect a homogeneous interface between each module. These characteristics allow the development team to focus and make emphasis on the tasks particular to the application domain. For a better understanding of its advantages, some of the projects in which the Framework has been used are presented together with the respective analysis of the results obtained.

**Key Words:** Software Engineering, Agile Developments, WEB Applications.

## 1. INTRODUCTION

SE (Software Engineering) basis is to have an established process for the development of Software Systems. A process defines a working framework for a set of key areas, known as KPA (Key Process Area), which should be established for the effective delivery of a software product. [1] [2]

In order to generate a robust and quality process layer, we should start from specifying the requirements of the problem [17]. These requirements should: (1) represent and understand the information domain of a problem; (2) define the functions the software will perform; (3) represent the expected behavior of the software (as a consequence of external events); (4) divide the models representing information, function, and behavior so as to discover the details by hierarchical layers. [3]

In order to keep the development of a IS (Information System) within the scope of the planning, we may minimize, among others, the time necessary to carry out the codification. Even though this time is minimum within a system's development cycle, the iterative tasks non specific to the application domain generally take between 50 to 60 % of the total assigned time. In addition, tuning, functionality debugging, and generation of the user's interface yield temporal values that cannot be considered as worthless. [4] [5]

Once the requirements are established it is possible to develop a complete, agile, and dynamic data model representing them as properly as possible. [6]

From here, a system implanting the required functionality will need basic modules managing the information contained in the DB (Data Base). The development and maintenance of each of these modules require spending time in routine tasks, keeping the consistency of the interface and correctness. [7]

The development team should focus on programming the minimum functionalities (using a programming language), updating the DB (generally with another specific language), building data upload forms, grids, combining visual components, among other activities [8]. In addition, an aspect important to any application – in particular when we are dealing with a IS – is coherence in the development of the interface, presenting the information and interacting with the user in a homogeneous and consistent way.

These objectives are generally tedious for developers. It is here, then, when four generation programming languages (4GL) (such as Clarion [9]) and automatic code generation CASE tools are particularly useful.

An automatic code generator is a tool which derives, from certain patterns, the source code of an application. The use of these tools reduces the time necessary for the software development, minimize errors, consequently reducing the debugging and tuning times. 4GLs consist in procedures which generate the source code in function of what is expressed in the application design or data model. For this, the user specifies the program functionality, or part of it, and the tool determines how to perform such task. [1]

However, the automatic code generation is not enough in many cases, since applications obtained from a 4GL have such *staticity* level that any change in the data model produces a great impact in the maintenance.

The working context –basically close to XP-type agile methodologies [18]- leads to an environment in which the data model dynamically undergoes conversions and / or

[1] Scholar III-LIDI UNLP - Fac. of Computer Science, ldelia@lidi.info.unlp.edu.ar
[2] Scholar III-LIDI UNLP - Fac. of Computer Science, gcaseres@lidi.info.unlp.edu.ar
[3] Full-Time Co-Chair Professor DE, UNLP - Fac. of Computer Science, hramon@lidi.info.unlp.edu.ar
[4] Full-Time Co-Chair Professor , UNLP - Fac. of Computer Science, pthomas@lidi.info.unlp.edu.ar
[5] Full-Time Co-Chair Professor , UNLP - Fac. of Computer Science, rbertone@lidi.info.unlp.edu.ar
[6] III-LIDI UNLP - Fac. of Computer Science, calle 50 y 115, La Plata (1900), Buenos Aires, Argentina, Tel/Fax +54 221 4227707 http://www.lidi.info.unlp.edu.ar

adaptations. Having a static CASE tool does not solve the problem successfully.

It is then necessary to think about the development of a CASE capable of dynamically adapting an application to continuous changes caused over the data model, keeping the regularity on the produced user interfaces.

## 2. PRESENTATION OF THE PROBLEM

Within the working context of the authors, the Institute of Research on Computer Sciences III-LIDI, several projects have been developed with similar characteristics, which require a high percentage of tables with the classical operations such as lists, filters, reports, data ADM (add/delete/modify). Each table, together with its classical associated operations, will be called herein *repositories.*

These projects are basically Web Systems [10], due to the need of accessing information from physically remote locations, and are developed with open-source tools, within LAMP (Linux + Apache + MySQL + PHP) environments [11] and interact with DBs integrated by heterogeneous information. This last causes a great effort to generate the interface of each *repository.* In consequence, it is essential to have a generic software layer that automates these tasks.

The complexity in developing this layer depends on the type of application in question. In applications of RAD type (Rapid Application Development), such as Delphi, PowerBuilder, VisualBasic, it is possible to parameterize components so as to get *repositories* with lesser effort [12].

In Web applications – based on client-server technologies [13] – the solution is rather more complex. That is, the process should be solved both on the client's side (with JavaScript, Java Applets), and the server's (with PHP, ASP, JSP, etc.), together with a way to show information (HTML, XML + CSS).

The development proposed – a Framework called PHP4DB – has been developed to solve the presented problems. PHP4DB is an object-oriented tool entirely developed in PHP, whose objective is to generalize as much as possible the software layer so as to automate routine codification tasks in a LAMP environment. In the following sections, PHP4DB is presented and its behavior is analyzed.

## 3. ARCHITECTURE AND DESCRIPTION

### Functionality

PHP4DB was a fully evolving development: a series of basic objectives were posed, which once achieved, they allowed "evolving" both in complexity and completeness. In the current version it is possible to carry out the following tasks:

- Visualize data of a *repository* through a paged grid.
- Dynamically filter data of the *repository* according to features defined for such end
- Obtain a quick view of a grid row
- Data ADM through a pre-established form
- Generate a PDF report of all the data visualized in the grid or of a particular data
- Relate a particular data to other functionality external to the *repository*
- Audit in XML format each operation carried out by the user in the *repository*

For this, PHP4DB dynamically communicates with the DB of the problem to be solved so as to recover or update the information there contained.

The development of this tool was meant, from the very beginnings, to be carried out in free-license products. For this reason, the DBMS used was MySQL. In subsequent versions we observed that the limitations implanted by the use of a particular DBMS were not adequate, and for this reason the Framework was evolved in order to abstract itself from the particular DB engine. In order to get the required abstraction, the PEAR (PHP Extension and Application Repository) DB library was used. [14]

Figure 1 presents the running of a *repository* as part of a system. Initially, as main access to such *repository*, a paged grid of data is shown together with an associated filter form. From here, it is possible to access all the remaining functionalities of the Framework. In the grid the listed data are described, which can be derived to a PDF report –as previously mentioned.

Actions can be applied to the shown data; some of them are basic, such as modification or deletion, and other may be specific to the *repository* and related to the behavior defined by the system requirements. All the specific functionality is associated to each grid row and is applied over it. The presentation may be done through a drop-down list or a tool bar.

In addition, as basic functionality, it is possible to insert new elements. Figure 2 presents an example of this form, while figure 3 presents the view of a particular record before a query.

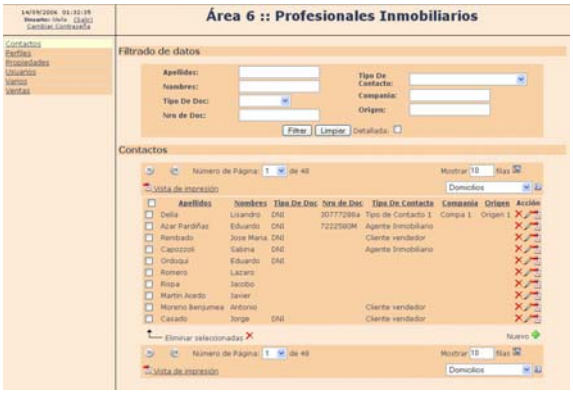


**Figure 1: Example of a repository**


**Figure 2: Add/Modify Form**


**Figure 3: Quick view of a record**

## Structure

PHP4DB is designed as a centralized core in charge of creating all the functionality mentioned in section 3. For it, each defined *repository* makes use of the core's functionality in order to present information.

The core needs to be configured for each project specifically, in order to respond to each application developed in the Institute. This gives rise to ProyectDataScript (PDS), an application configuration file, which has the description of the project's DB (server, DBMS, user, password), the style interfaces will have (CSS, Icons), as well as the necessary additional information.

By means of the PDS, the core has the system configuration and the information which is common to all the *repositories.*

On its part, a file called FDS (FormDataScript) was defined, and is in charge of providing all the specific information of each *repository* and of the table to which it refers. With this data descriptor, PHP4DB can provide all the functionality for the associated *repository.* FDS describes, among many other things, the following information:

- Titles for each *repository* operation
- Name of the DB table, to which the *repository* refers
- Fields of the DB table, where for each of them we have:
  - Field Name
  - Significant Label to show
  - Visibility in grid/reports
  - Visibility in filter
  - Field Type
- Access for each function of the repository, with the objective to enable/disable functions according to the user's profile.

Figure 4 presents the structure of the PHP4DB Framework. Each FDS contains the information of a particular *repository*. When one of these *repositories* is invoked, the FDS sends all the information to the PHP4DB core, which carries out some of its functions, recovering the DB's information.

It is worth to mention that each new functionality incorporated to the Framework's core, such as data export to a particular format, is obtained by each *repository* without any modification. The same happens with the maintenance of each functionality or error correction; every *repository* will receive these benefits, without altering its contents.

It is interesting to stress that when we need to create a new *repository,* it is not necessary to add any programming (be it PHP, HTML, or SQL code). We only need to create the FDS associated to the *repository.*

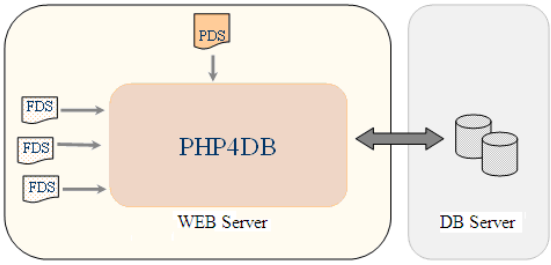


**Figure 4: Structure of the PHP4DB Framework**

## Assistant for the creation of FDS

Even though developers can generate/modify FDS manually, a tool which automates this task has been created, saving time and avoiding errors due to the cumbersomeness of the manual option.

This tool is a desktop application, called PHP4DB Assistant, which in very few steps allows creating FDS for each *repository*. Taking into account that PHP4DB is a Framework meant to coexist with several projects simultaneously, the first task consists in using the FDS associated to the project in which the new *repository* is to be added. PHP4DB Assistant visualizes the tables of the project's DB, where the user chooses the table to which the new *repository* will refer.

Once the table is selected, the information of its fields is automatically deployed, and the user should then configure some details such as: (1) the *labels* of the fields, (2) the type of basis data of each field (text, number, foreign key, etc.), (3) the titles of the different actions, (4) the functionalities which will be active for the *repository*, etc. Figure 5 presents a summary of these details.

Once configured, the FDS file is stored in a web server, and we are ready to present the *repository* from any browser.

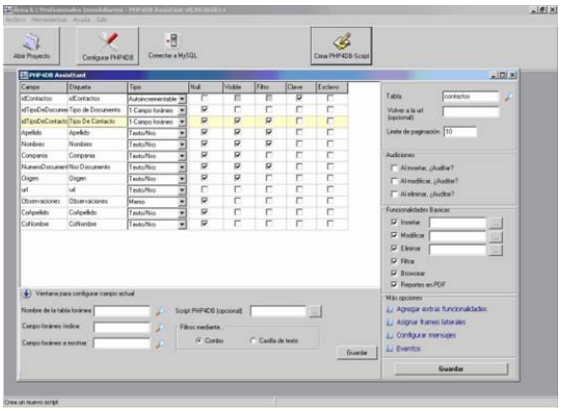

**Figure 5: Creation of a repository with PHP4DB Assistant**

**Characteristics of the repositories**

**Types of fields**: In FDS files, the information about the fields to be presented in the forms, grids, filters, and reports is described. These fields vary one from each other. For instance, the way to present a date in a form should not be the same as that to show a text. For this reason PHP4DB identifies each field with a particular type.

At present, the Framework can work with the following field types:

- Short and long texts
- Integers and floating number
- Dates
- Images
- Foreign fields
- Booleans

Having an object-oriented design, adding a new data type to the Framework is not a costly task. In this way, it is possible to easily extend the Framework, enhancing the *repositories'* scope.

**Events**: Even though all the basic functionalities of a *repository* can be automatically solved, there exist cases in which some of them need to behave in other ways.

PHP4DB offers the possibility that the *repositories* have orientation to events, allowing running action in given moments of the execution. In order to do this, PHP4DB verifies whether the FDS being run has defined the event corresponding to the running point. If it is defined, PHP4DB invokes the event; otherwise, it keeps on working normally. Events such as *before_execute_insert()* or *after_execute_insert()*, just to quote some, increase the Framework's dynamism level.

**Relation to other functionalities**: It was mentioned that within the ideal working environment would be that in which the development team dedicates the time in modules which require a specific programming, without loosing it in the development of the *repositories*.

In the case of having specific modules, there exists the need to relate them to other *repositories*. Figure 6 shows how a particular record can be related to other functions of the system. By means of a drop-down list, or simple icons in each grid row, an action can be applied to a selected record. These actions imply calls to other modules which have been specifically developed, or else, other *repositories* created with the PHP4DB Assistant.

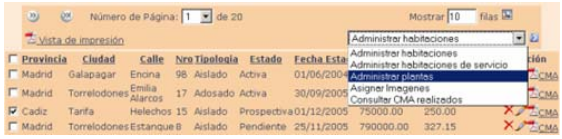


**Figure 6: Relation to other functions**

## 4. RESULTS OBTAINED

As previously mentioned, the Institute has developed a large number of systems with transference. For this reason, having a tool like PHP4DB has minimized the codification time, among many other things. Next, a description of the scope of some of these projects can be found.

**Area 6 – Real State Professionals**

Web-oriented CRM (Customer Relationship Management) Multi-Real State System, at present under development in Spain. Its objective is to manage real state activities inherent in a real-state life cycle, from its entrance to the market to its sale. In addition, it provides the service of objectively estimating the sale price that a real state should have, basing on other real states with similar characteristics. This is the central process of this application and is called CMA (Comparative Market Analysis)

**Hospital Full Management Software**

The objective sought under the project called SAIH-LIDI consists in the complete computerization of hospitals, both with self-management and those depending of a pre-assigned budget.

This is accomplished by integrating the attendance at medical centers, admission to hospitals, and external services (which in some cases may consists in patients' referrals), generating a basic Medical Record for each patient. In addition, it allows managing the collection from Medical Insurance Services. All this allows generating an informative quality framework towards the patient and external consultations, solving the internal management of the hospital.

**Provincial Directorate of Computer Science and Communications of the Province of Buenos Aires**

The Province of Buenos Aires is managing a public bidding process to provide the "Data Transmission and Order Channels Service (*Servicio de Transmisión de Datos y Canales de Ordenes*)" for the Single Provincial Network of Data Communication (*Red Única Provincial de Comunicación de Datos*). RedPIBA (Provincial Network of Research and Development Teams in areas of Computer Sciences) was in charge of defining the procedure guidelines so as to unify criteria and the mechanism to obtain the acceptance of nodes, defining the framework for their training in specific tasks, and controlling the projects monitoring. [15]

This provincial network has 1300 nodes, over each of which an audit is carried out and then each enters the new network production. For this, the province was divided into 6 zones, each of them with a University belonging to

the RedPIBA as head: UNLP (National University of La Plata), UNLM (National University of La Matanza), UTN (National Technologic University), UNLu (National University of Luján), UNC (National University of the Center), UNS (National University of the South). Each one has a coordinator and two teams of technical specialists. Also, there exists a central coordination team.

Coordination tasks for the certification were carried out by means of a WEB application developed with PHP4DB.

## 5. CONCLUSIONS

The use of a Framework in the projects was crucial, since it was possible to automate a high percentage of use cases (UCs).

In the project Area6, of the 50 UCs only one was specifically programmed (CMA) [16], being the most complex function which requires statistics and particular ways of use. SAIH-LIDI, on its part, has 30 UCs implemented up to the present. Of these 30 UCs, only 12 (Shifts and Chemists') were specifically implemented. DPIC has 30 UCs and only 4 received particular programming.

The benefit obtained with PHP4BD is quite clear. The development time was significantly reduced by the use of the tool, with the subsequent satisfaction of the user due to the early availability of the required products. In addition, the centralization provided by PHP4DB has allowed obtaining homogeneous interfaces, easing the posterior maintenance.

Finally, it is worth to notice the expectation created by the availability of this framework for future applications that require web orientation.

## 6. FUTURE WORK

Even though the Framework has such a maturity that enables operability with any table of any DB engine, the technological whirlwind leads to taking into account other information domains such as XML files or views encompassing information of several tables. With these extensions the Framework usability will be enhanced, allowing it to get along with a larger quantity of Systems.

The incorporation of new type of data for PHP4BD will also be beneficial. Examples of these benefits are the possibility of storing any type of file in tables (.doc,.mp3,.mpeg), or having fields whose values are defined by the user, among many others. Any of these will increase the product usability even more.

Last but not list, an important task is to take the assistant mentioned in section 3 into a WEB platform, so as to create/modify the *repositories* from any location.

## 7. REFERENCES

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

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

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

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

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

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

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

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

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

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

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

[12] Johannes Sametinger, Software Engineering with Reusable Components, Springer, 2001.

[13] Rick Leander, Building Application Servers, Cambridge University Press, 2000.

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

[15] http://www.dpic.sg.gba.gov.ar

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

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

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