EXAMPLE-BASED LEARNING: DEVELOPMENT OF BUSINESS APPLICATIONS WITH .NET TECHNOLOGIES

APRENDIZAJE BASADO EN EJEMPLOS: DESARROLLO DE APLICACIONES EMPRESARIALES CON TECNOLOGÍAS .NET

Marcos Gestal*, José M. Vázquez ** Enrique Fernández-Blanco***, Daniel Rivero****, Juan R. Rabuñal*****, Julian Dorado******, Alejandro Pazos******
Universidade da Coruña, España

Received: 04/12/14 Accepted: 01/05/15

ABSTRACT

For a long time, J2EE has been the dominating framework for the development of business applications. This fact resulted in a rich ecosystem of tools, manuals, tutorials, etc. that explain different implementation alternatives or peculiarities. The incursion of .NET Framework in the business environment has generated a strong demand of application implementation under this architecture. However, the quantity and quality of documents available significantly differs from its main alternative (J2EE). This documentation gap is especially visible and worrying when the objective is to teach the concepts of Framework, from a teacher's point of view, to the future graduates of the Information Systems Engineering program. This paper describes the teaching approach used in order to achieve the goal of having the students become familiar with this alternative framework and the usual model practices within it. Thus, it is based mainly on a set of basic tutorials that show the foundations of technology and two complete applications (miniportal and minibank) explaining how to apply design patterns when developing a business solution.

Keywords: systems engineering, software, programming language, design patterns, web applications, project-based learning, problem-based learning.

RESUMEN

El framework J2EE ha sido el gran dominador, durante mucho tiempo, en el desarrollo de aplicaciones empresariales. Esto hecho originó la aparición de un rico ecosistema de herramientas, manuales, tutoriales, etc., que explican las diferentes alternativas o peculiaridades

*mgestal@udc.es
**jmvazquez@udc.es
***efernandez@udc.es
****drivero@udc.es
*****juanra@udc.es
******julian@udc.es
*******apazos@udc.es

a la hora de su implementación. La irrupción de .NET Framework, en el ámbito empresarial, ha producido una fuerte demanda de implementación de aplicaciones bajo dicha arquitectura. Sin embargo, la cantidad o calidad de la documentación disponible dista considerablemente con respecto a la existente, su principal alternativa (J2EE). Esta laguna de documentación, es especialmente visible y preocupante cuando se establece como objetivo dar a conocer los conceptos del framework, desde un punto de vista docente, a los futuros egresados del Grado en Ingeniería en Informática. Este trabajo describe el enfoque docente seguido, para alcanzar el citado objetivo de familiarizar a los alumnos con este framework alternativo y las prácticas habituales de modelo dentro de éste. Para ello, se basa principalmente de un conjunto de sencillos tutoriales con los que mostrar los fundamentos de la tecnología y dos aplicaciones completas (miniportal y minibank) en las que se muestra cómo aplicar patrones de diseño, a la hora de abordar una aplicación empresarial.

Palabras clave: ingeniería de sistemas, software, lenguaje de programación, patrones de diseño, aplicaciones web, aprendizaje por proyectos, aprendizaje basado en problemas

INTRODUCTION

“Development Framework” is a mandatory course in the 4th year of the Information Systems Engineering program. It is a four-month course, and has a total of 6 ECTS credits. These are divided into 3 credits for expository instruction (lectures), and 3 for interactive instruction, which translates into 21 lecture hours and 90 practice hours for students. This also includes the estimated work outside lab hours. Thus, this is a fundamentally practical course in the last year, which seeks to familiarize the future alumni of the Information Systems Engineering program with the usual exercises they will encounter in the labor market.

The curriculum presents this course as a complement to “Advanced Programming,” where students see the J2EE framework. This research also improves the students’ training by showing them the .NET technologies’ point of view (Grimes, 2002).

This research not only shows a technological viewpoint, but also the most relevant and appropriate project rules to operate within this framework and solve common problems in the development of web applications (Zeldman & Ethan, 2009). Additionally, as it complements another course as mentioned above, it is possible to emphasize the differences with the framework seen by the students previously. Throughout the course, a series of tutorials on the necessary concepts to be used in their practical assignment are presented: design patterns, technology, developmental environment, etc. Also, a series of web applications are developed and made available to students in the way of tutorials, detailing how to implement the concepts previously shown: user authentication, data validation, transaction management, etc.

This integrated approach, presenting technology along with structuring methods, is encouraged by the fact that it is impossible to apply a technique appropriately if it is not related to an engineering methodology based on the correct application of samples. Additionally, some procedures are so difficult to understand in an isolated manner that they need to be contextualized within the problems that trigger the appearance of a model for certain technological solutions.

This research is heir to the Systems Integration course, from the former Information Systems Engineering discipline (Gestal, Rivero, Rabuñal, Dorado & Pazos, 2010). It is worth mentioning that the contents have been modified to adjust to the new schedule and training scheme of the Information Systems Engineering program.

The evaluation consists on creating a web application, but it is necessary to obtain a minimum grade in a test verifying that the concepts required for the practical assignment have been acquired.

For this research, we first detail the specific objectives of the course, and then describe the teaching methods. Moreover, we develop the concepts explained to students and the examples used for that purpose in depth. Finally, we draw a series of conclusions based on students' ratings.

Course Objectives

"Development Frameworks" focuses on the presentation of design and structural patterns to create and implement web applications with .NET technologies (Zeldman & Ethan, 2009).

Throughout the course, we develop a business-like web application with .NET. This assignment represents the regular operation of a real application (obviously limited in terms of functionality due to time constraints) of the concepts developed during the course.

Carrying out their practical assignment correctly will allow students to reach the following main objectives in the course:

- Knowing programming fundamentals through .NET technologies
- Mastering the basic architectural principles of business applications
- Understanding design techniques to develop business applications (specially web applications) through a layered architecture

In order to reach these objectives, the course provides 2 credits for lectures, which are focused on two fundamental aspects:

- Design and implementation of the model layer
- Design and implementation of the Web layer

These two points are reflected, developed and experimented throughout the practical assignment of the course, in order to obtain the 4 credits. As can be seen, these aspects detail the two main parts of the Model-View-Controller (MVC) (Gamma, Helm, Johnson & Vlissides, 1995). This archetype is presented in combination with the layer pattern (de la Torre, Zorrilla, Calvarro & Ramos, 2011) and represents the architectural guidelines in current business applications. As support for the explanation of these and other usual samples in the business field, we present two reference applications in the way of examples thoroughly documented and commented. These two examples, analyzed in the following sections, seek to explain how to adjust prototypes during framework implementation in an easy manner and to become the basis to develop the practical assignment further.

TEACHING METHODOLOGIES

Instructor Methodology

Despite the evidently practical approach of the course, there are also lectures. These theoretical lectures present the basic concepts necessary for the practical assignment, for example, dependency injection. The theoretical concepts are later materialized in the implementation of the framework. In this way, we put special emphasis on data base access technologies and the software patterns used in the creation of model, view and controller layers in business applications.

The practical assignment carried out throughout the course is group work, in order to imitate the typical structure of developing teams that are usually found in the advanced software field. This also enables students to become familiar with the design and with the good practices of software development in teams, such as keeping a software version repository, responsibility allocation, task planning and allocation, and the application of work methodologies.

The practical assignment, in its turn, tries to emulate the generic functioning of some of the most common web applications of renown business success (Betandwin, Amazon, etc.), although, evidently, with a smaller number of functions due to time constraints. Imitating applications is motivational for students as they can try, first hand, how to create such solutions (Bugeja, 2007).

Due to the extension and complexity of the practical assignment, its submittal in parts makes it possible for a larger number of students to meet deadlines and to reduce dropouts. This follows the typical strategy of methodological evolution for spiral software projects (Boehm, Madachy & Steece, 2000), which we also seek to make students become familiar with, since it is one of the most popular in the labor market.

EVALUATION METHODOLOGY

The largest weight is on the practical assignment. Therefore, more emphasis is put on the practical aspect when evaluating students.

As already commented on, the progress of the practical assignment during the course is divided in two submittal dates. In the first, which is not graded, the initial part is implemented.

The objective of the first submittal is to try to ensure that students are focusing well on the development, and that they are applying concepts appropriately. For this purpose, the instructor detects important mistakes and then guides students towards solutions. In the second, students correct the mistakes detected in the first submittal and add the remaining functions. During the evaluation of this second submittal, students must explain the functions implemented, how the work was divided, etc. With this, they are assigned an individual grade, which is the most important grade in the course.

In order to guarantee the complete correction of the practical assignment and for it to be consistent in all the groups, the same scheme is used, based on the appropriate verification of a series of control points. This correction is guided by a checklist in order to verify the completion of a series of aspects in the assignment, although, evidently, it will depend on some aspects of the practical assignment proposed each year. As an example, you can review one of these checklists in Attachment I in a more detailed manner.

Additionally, upon correction of the practical assignment, instructors will revise the quality of the report delivered or the remaining documents in more detail. To do this, based on the checklist notes, during the correction of the assignment, we will use a simplified document allowing for the final evaluation based on a series of criteria that are easily measurable, both quantitatively and qualitatively. To avoid subjectivity in the evaluation, each section corrected has the characteristics to be contained in the assignment noted, in order to obtain a certain grade. This document can be read in Attachment II.

.NET as a Learning Means

The framework, developed by Microsoft and known as .NET, was created in 2002 (Grimers, 2002), and was developed over more than a decade. It caught people's attention from the beginning, being an alternative to the J2EE proposal, when implementing robust, reliable and durable business applications.

This frequent use of the framework within business applications determines the fact that students, besides knowing and using J2EE solutions, also have an array of work possibilities through the proper training in the solutions offered by .NET.

Thus, our course objective is to explain the architectural patterns of business applications, using the .NET framework as support. Therefore, we suggest an introduction starting from the general models (Gamma et. al., 1995), and then the implementation of general concepts in real cases with this Microsoft's framework architecture (Zeldman & Ethan, 2009). It is worth mentioning that knowing the framework is not a primary objective, but to understand it as a means for students to comprehend the global ideas explained when seeing a real application.

Along this line, to understand the concepts of the archetypes explained, we propose a series of examples, which also serve as reference material later on. Each one of these shows how usual functions and structures within web pages work and are implemented. Later, students must apply them in their practical assignment. These tutorials are implemented under the philosophy of being a learning guide to technology, leaving aside aspects such as design efficiency and appropriateness. This approach derives from the fact that most technologies and prototypes explained are new to students, so we try to facilitate understanding as much as possible.

Therefore, students are provided with a series of examples which show, first, basic aspects of programming language C# (structures, exception management, etc.). The tutorial includes examples of connection to data bases through ADO.NET (connected and disconnected environment), as well as through ORM Entity Framework (Andrew & Schafer, 2006).

In the first part of the course, we seek to make students become confident with and acquire a language and development environment unknown to many before the course, but that are currently widely developed and implemented in the information systems field. That is why the complexity of the tutorials increases as the course advances.

This learning and adaptation stage, from other languages previously known by students, happens usually fast and with no trouble, as ours are last year students with a considerable background in other languages and frameworks.

However, we have detected a lack of training in web related issues in students. To this regard, we provide them with a list of tutorials that may help them overcome said deficiency, such as web safety, ASP.NET, view, XML, etc. We provide a complete list of tutorials in Figure 1.

Upon completion of this stage, we explain the architectural patterns on which most commercial web applications are based, emphasizing the benefits they present. All of this is always done when explaining how to implement these guidelines in the framework, so students experience firsthand how concepts materialize in their practical assignment. Thanks to this, we manage to lessen the students' learning curve gradient since they can verify, based on functional examples, the concepts required later on for the assignment.

The large number of examples also facilitates carrying out their assignments since the codes can also be used. In this way, students finally elaborate a complex web application (perfectly usable) without having to code it completely.

The examples provided focus on two small complete web applications: MiniPortal and MiniBank. They emphasize the fundamentals of business web applications. These applications cover different aspects (layers, authentication, transaction management, etc.), which students will later apply in carrying out their practical assignment. Next, a brief description of each one of the contents.

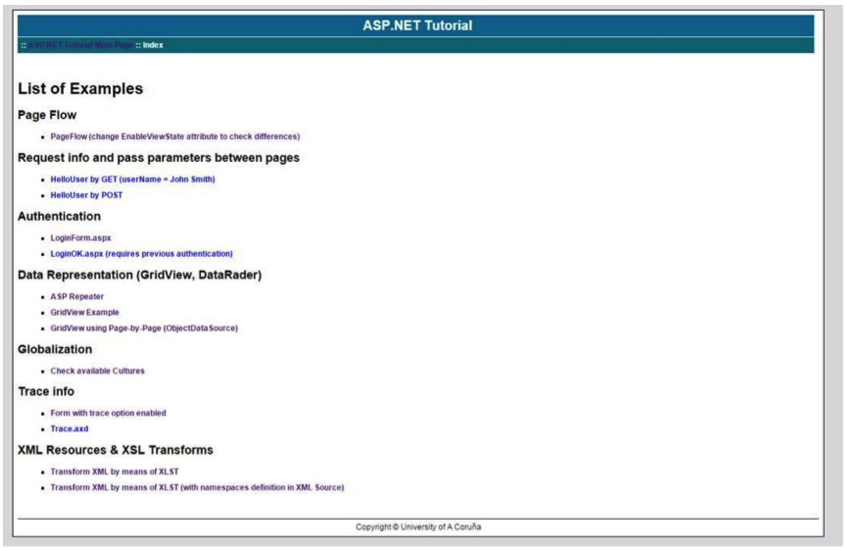
Implementing Business Logics with ADO.NET

Current application development is very much focused on the division of responsibilities that allows the increase of the most robust and easy maintenance apps. In order to achieve such division of responsibilities, we use an architectural pattern known as Layers (de la Torre Llorente et al., 2011). Such design argues that it is essential to group -under a same layer- all those functions belonging to one of the aspects of the application. Concretely, this prototype is usually combined with Model-View-Controller (MCV) (Gamma et al., 1995), which requires the separation of the business activity proper logic from the one necessary to represent the information contained in the business logic. These concepts are implemented through specific frameworks, such as ORM Entity Framework. The problem we find is that the terms used are too conceptual and far from lower levels. It is for that reason that the examples provided are implemented in ADO. NET, although the subject is focused on the use of Entity Framework (Leman, 2010), as this is the market trend. Therefore, the practical assignment starts with the implementation of the logical layer or model layer. This is the basic layer of the final application. In this way, students will have to implement the whole business logic and data persistence in one of their bases, following the layers paradigm. In other words, they will have to implement two layers: one for persistence and the other for the business operation, so changes in one will only affect the other minimally. Besides the architectural patterns mentioned, students use others, such as different structuring examples: facade, factory, template archetype, etc.

The persistence layer is implemented, first, through one of the framework classic technologies to connect to a data base, as the ADO.NET protocol. To do this, we use the MiniPortal application as an example, as mentioned before. This is a complete web application that implements a portal allowing user registry and authentication. Figure 2 shows the main business objects. This is one of the requirements that the students' practical assignment must meet, so the example provided should be reused with minimal changes.

Image of DAO Architecture

This domain objects are made persistent, using the ADO.NET architecture and the Factory and DAO patterns. ADO.NET provides access to the information stored in a data base in order to store the business object data, in this particular case, the UserProfileVO and the UserProfileDetailsVO. Additionally, ADO.NET



ASP.NET Tutorial

:: ASP.NET Tutorial Main Page :: Index

List of Examples

Page Flow

- PageFlow (change EnableViewState attribute to check differences)

Request info and pass parameters between pages

- HelloUser by GET (userName = John Smith)
- HelloUser by POST

Authentication

- LoginForm.aspx
- LoginOK.aspx (requires previous authentication)

Data Representation (GridView, DataRader)

- ASP Repeater
- GridView Example
- GridView using Page-by-Page (ObjectDataSource)

Globalization

- Check available Cultures

Trace info

- Form with trace option enabled
- Trace.axd

XML Resources & XSL Transforms

- Transform XML by means of XLST
- Transform XML by means of XLST (with namespaces definition in XML Source)

Copyright © University of A Coruña

Figure 1. Tutorials

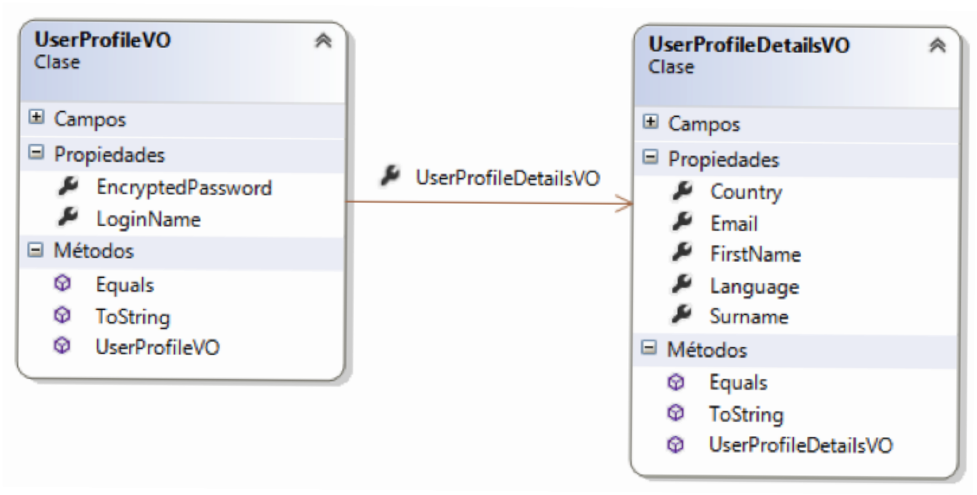


Figure 2. MiniPortal Domain Objects

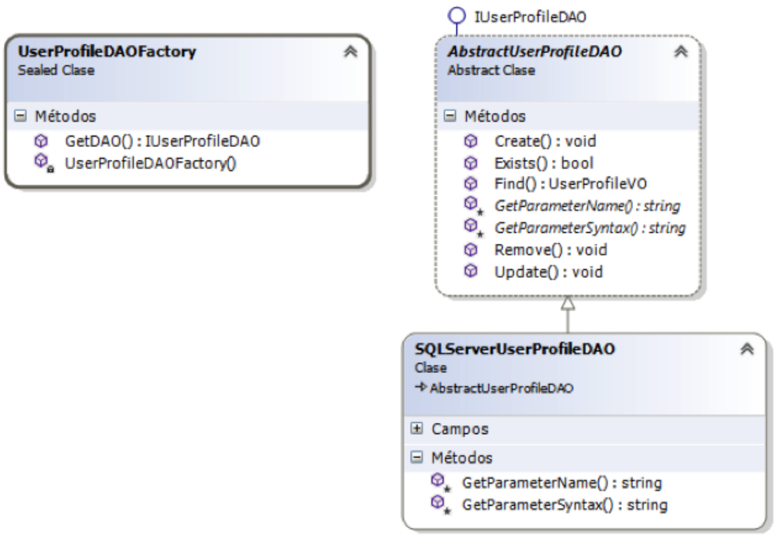


Figure 3. DAO Pattern Architecture for UserProfileVO

provides the DataSets function, which reduces data traffic by retrieving the data describing the relational data base and operating with the latter before consolidating the changes in a sole connection. By using the DAO prototype, students experience firsthand how to completely separate the Data Base connection logic from the business logic operations, thus encapsulating the persistence of an independent layer that can suffer changes with no repercussions in the upper layer. Figure 3 shows the architecture to achieve such separation.

In MiniPortal, students are shown the division of a business application, following the Model-View-Controller (MVC), in a practical manner. Such pattern is the basis to understand the separation of business application layers, as well as their scalability. It also exemplifies the use of certain design models (Session Facade, Business Delegate, Factory, etc.) to encapsulate each layer's functions. For example, in the type case, we use the Facade example in order to achieve this separation of functions gathered from use cases (operations supported by the application). Figure 4 shows such use cases in MiniPortal. In the implementation of the part-sample, we put special emphasis on the management of the data stored in the session (data to be accessed on different pages). Also, we show how to develop tests (using TestProject as a tool within .NET), how to use configurable parameters externally, etc.

With the other complete example (MiniBank), students are shown a simplification of bank account management. The domain objects, which are to be persistent in the data base, can be seen in Figure 5. MiniBank is a more complete and complex application than MiniPortal because, even though it does not register new users, it does have user authentication, as seen in the previous example. Besides authentication, MiniBank considers the basic operations on a current account, i.e. creation, search, cash deposits or withdrawals, as well as transfers between different accounts.

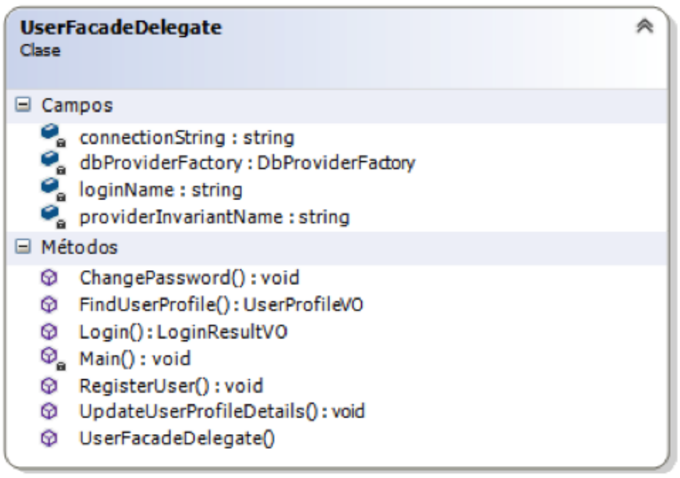
As with MiniPortal, MiniBank uses different design and architectural patterns typical in business applications, in order to achieve the separation of the layers' different functions. As with MiniPortal, MiniBank uses DAO and Facade extensively to achieve the separation between the business logic, persistence and the remaining functions. Figure 6 shows the model facade with all use cases in the application.

On the other hand, MiniBank tries to cover aspects MiniPortal does not contemplate and that students should use in their practical assignment. Thus, we show aspects related to the automatic generation of identifiers for data base storage, transaction management or Page-by-Page iterator pattern (that allows results pagination when these are too many to be shown at once).

Implementing View and Controller with ASP.NET

Once the design has been implemented, during the second submittal of the practical assignment, students implement the application's web layer, i.e. data presentation and the way it interacts with the business logic previously created. In order to do this, they follow MVC and then implement the application's controller and view with ASP.NET. Additionally, they are introduced another series of good practices when developing layers and strategies (Andrew and Schafer, 2006).

To exemplify the concepts developed in the application's view and controller, MiniPortal shows aspects related to page navigability (see Figure 7), profile management, internationalization, page parameter pass, model method calls, etc.

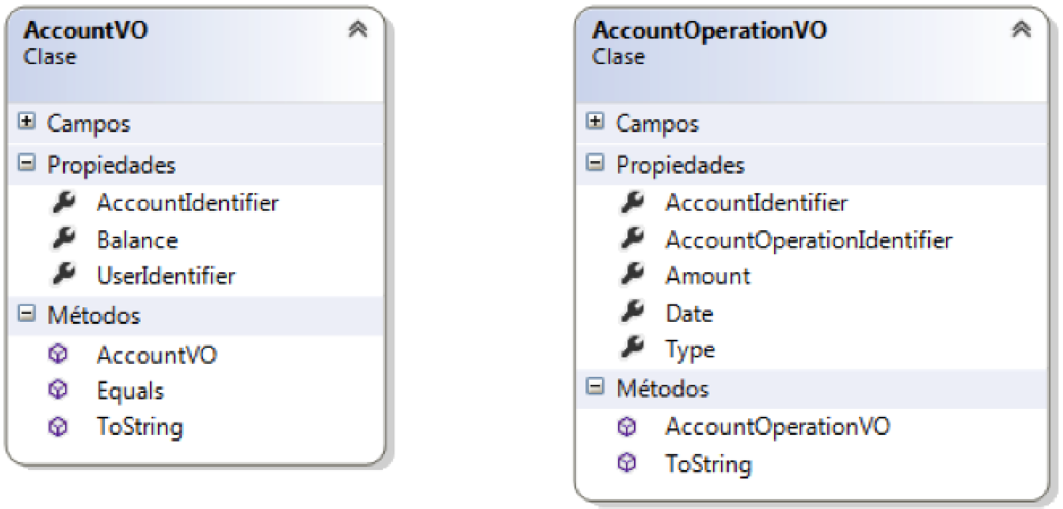


UserFacadeDelegate
Clase

Campos
connectionString : string
dbProviderFactory : DbProviderFactory
loginName : string
providerInvariantName : string

Métodos
ChangePassword() : void
FindUserProfile() : UserProfileVO
Login() : LoginResultVO
Main() : void
RegisterUser() : void
UpdateUserProfileDetails() : void
UserFacadeDelegate()

Figure 4. Screenshot of MiniPortal Use Cases



AccountVO
Clase

Campos

Propiedades
AccountIdentifier
Balance
UserIdentifier

Métodos
AccountVO
Equals
ToString

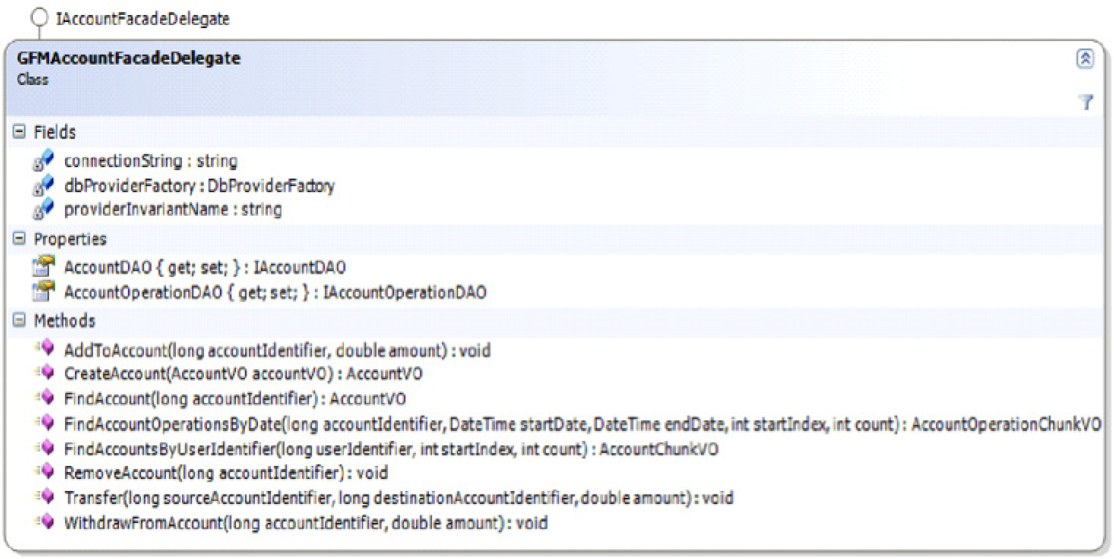
AccountOperationVO
Clase

Campos

Propiedades
AccountIdentifier
AccountOperationIdentifier
Amount
Date
Type

Métodos
AccountOperationVO
ToString

Figure 5. MiniBank Domain Objects



IAccountFacadeDelegate

GFMAccountFacadeDelegate
Class

Fields
connectionString : string
dbProviderFactory : DbProviderFactory
providerInvariantName : string

Properties
AccountDAO { get; set; } : IAccountDAO
AccountOperationDAO { get; set; } : IAccountOperationDAO

Methods
AddToAccount(long accountIdentifier, double amount) : void
CreateAccount(AccountVO accountVO) : AccountVO
FindAccount(long accountIdentifier) : AccountVO
FindAccountOperationsByDate(long accountIdentifier, DateTime startDate, DateTime endDate, int startIndex, int count) : AccountOperationChunkVO
FindAccountsByUserIdentifier(long userIdentifier, int startIndex, int count) : AccountChunkVO
RemoveAccount(long accountIdentifier) : void
Transfer(long sourceAccountIdentifier, long destinationAccountIdentifier, double amount) : void
WithdrawFromAccount(long accountIdentifier, double amount) : void

Figure 6. Screenshot of MiniBank Use Cases

One of the main aspects here are the different controller architectures that can be implemented even if the behavior is the same. The controller implemented with .NET follows an allocated architecture with a Page-Controller philosophy, while in other frameworks like Struts, there is only one generic controller for the whole Application-Controller application. At this point, we highlight the benefits and drawbacks of each architecture and where each one is more appropriate. We also introduce aspects related to web development within the framework, as are the difference between observing the complete panorama and partial perspectives, the inclusion of JavaScript (Harold & Means, 2004) to make the structure more dynamic, etc. Some of the technologies and concepts here are AJAX, JQuery, among others.

On the other hand, MiniBank is a little simpler, but it is useful to remind students of the concepts already seen with MiniPortal. However, everything related to results pagination (see Figure 8) through the Page-by-Page archetype, commonly used in the presentation of search results, for example, is new.

Implementing the Model with ORM Entity Framework

Once the concepts based on ADO.NET have been internalized, students are explained how these are translated into the ORM Entity Framework (Leman, 2010). They will have to use the latter to implement persistence. Therefore, students have been developing the other parts of the MVC with a temporary persistence. Additionally, students are asked to substitute the latter for the corresponding data persistence layer, implementing it with ORM Entity Frameworks. This technology is a current trend based on dependency injection. For this reason, we introduce this last concept here. This concept is the basis for the majority of recently implemented applications, thanks to its flexibility and easy maintenance.

We seek to make our students knowledgeable of the current development trend and to make them experience, firsthand, a usual process in information technology increase, as is technology migration. Additionally, this change seeks to make them aware of the advantages implied in the separation of design layers, as changes will be limited to a certain section, without modifying codes or altering the functions of upper layers.

EVALUATION

As discussed earlier, this course is eminently practical. Therefore, evaluation is based on that aspect. In consequence, instructors evaluate the practical assignment after its submittal. The group presents their defense of the assignment submitted. This defense includes verifying the implementation of the practical assignment through a checklist of critical points. Another aspect to evaluate during the process is the correct understanding by the group of the concepts applied in the development of the activity. Based on the seriousness of the mistakes detected, if any, the web application will be evaluated from 0 to 10 points.

Additionally, students will render a test. The objective of this evaluation is to determine if the student acquired the concepts correctly. The test is composed of a series of multiple questions, where only one answer is correct. Unanswered questions obtain no points, and those answered incorrectly subtract points.

In order to pass the course you must: (1) have obtained at least 5 over 10 for the web application, and (2) have obtained at least 4.5 over 10 in the test. In principle, the final score for students who meet those two conditions is the final grade, which is the weighted sum of the practice and the test scores. Given that this is a practical course, the practical assignment represents 60% of the final grade, and the test only 40%. Although instructors would like the practical assignment to have more weight, university regulations do not allow it.

CONCLUSIONS

According to students' surveys, this course is one of the most popular in the program, despite being one of the courses with the highest work load (or maybe because of it). This happens because students, in creating a real web application -even with limited functions-, see the results of their work constantly, and can interact with their application. The explanatory methodology, based on real case study of applications, strengthens students' training, avoiding long and dull theoretical sessions (De Miguel Díaz et al., 2006). These concepts, taught through examples they can and must apply in their practical project, consolidate their acquisition. Additionally, explaining concepts based on examples triggers more student participation, generating comments or suggestions to the instructor, which he can later introduce to the explanation with feedback. Such exchange of ideas motivates a continuous improvement process in the instructor, which translates in the increase of teaching quality and a faster adjustment to the audience (Navarro, 2007).

Regarding the examples, we can say they are highly accepted. Analyzing the server logs, we can observe that students' queries are just a small part of them all.

Finally, carrying out a real and functional practical assignment, adjusting to the growth standards for this type of projects in the industry, triggers extra motivation in students. They see concepts and generate contents which are directly applicable for graduated students, so we notice their greater interest and involvement. This greater involvement, in many cases, translates into the students' enrollment in a continuous cycle of training and recycling of the subject concepts (López, 2002), which is very beneficial for their work life, so as not to produce the obsolescence effect (Rodríguez & Barrios, 2014).

REFERENCES

Andrew, R. y Schafer, D. (2006). *HTML Utopia: Designing Without Tables Using CSS* (2nd. ed.). Collingwood: SitePoint.

Boehm, B. W., Madachy, R. y Steece, B. (2000). *Software cost estimation with Cocomo II with Cdrom*. Upper Saddle River, NJ: Prentice Hall PTR.

Bugeja, M. J. (2007). Distractions in the wireless classroom. *The Chronicle of higher education, 53*(21), C1-C4.

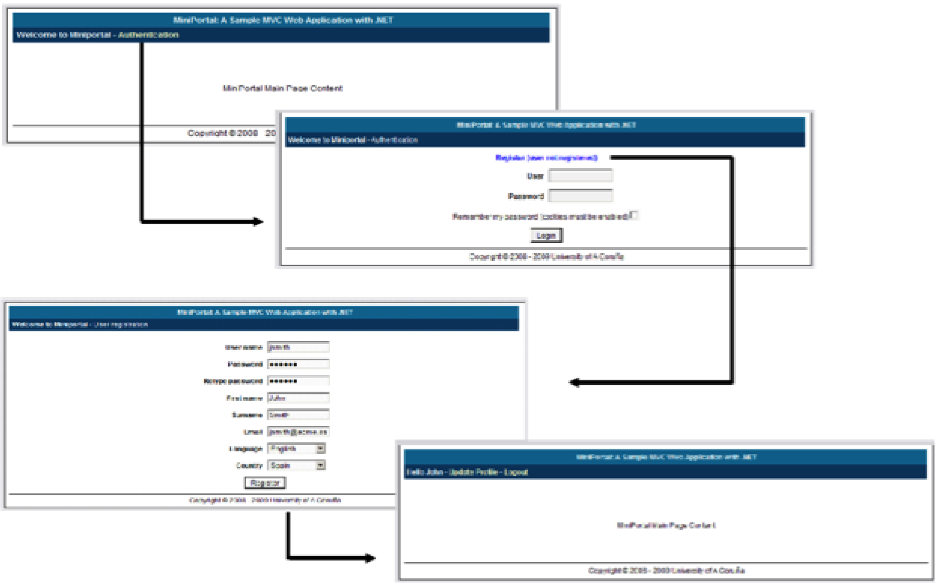
Comas, O. y Lastra, R. S. (2014). La obsolescencia de los saberes frente a las necesidades de aprender; un caso de estudio. *Reencuentro, 69*, 22-27.

Gamma, E., Helm, R., Johnson, R. y Vlissides, J. M. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software. A Pattern Language: Towns/Buildings/ Construction.* Reading: Addison-Wesley.

Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J. y Pazos, A. (2010). *Basics of Web Application Design: an Example-Based Learning Approach.* Paper presented at the International Conference on Computer Supported Education.

Grimes, F. (2002). *Microsoft .NET for Programmers.* New York: Manning Publications.

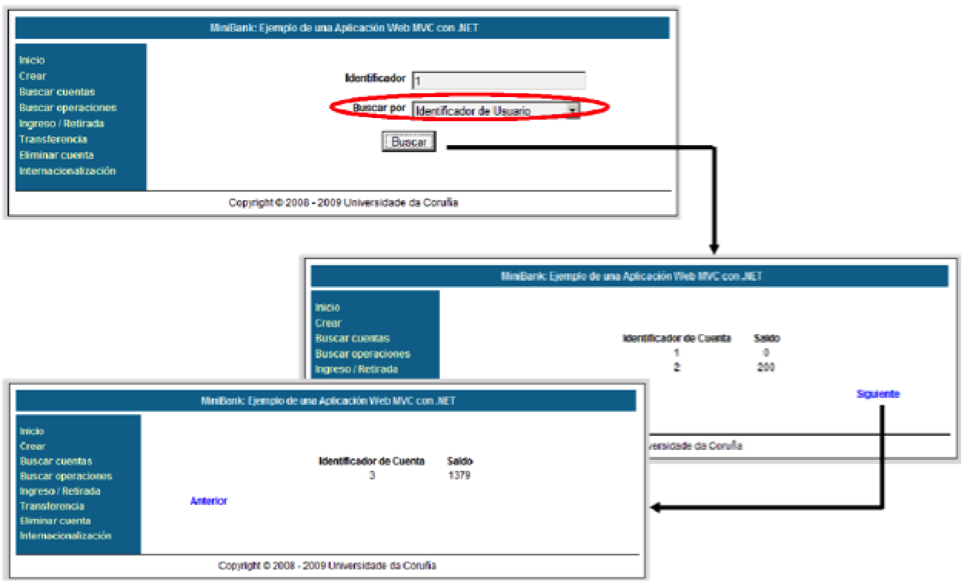
Harold, E. R. y Means, W. S. (2004). *XML in a Nutshell: A Desktop Quick Reference*


MiniPortal: A Sample MVC Web Application with .NET
Welcome to Miniportal - Authentication
MiniPortal Main Page Content
Copyright © 2008 - 2009 University of A Coruña
Register (user not registered)
Remember my password (cookies must be enabled)
Login
Welcome to Miniportal - User registration
Register
Hello John - Update Profile - Logout

User
Password

User name: jsmith
Password: •••••••
Retype password: •••••••
First name: John
Surname: Smith
Email: jsmith@acme.es
Language: English
Country: Spain

Figure 7. Example of Interaction with the MiniPortal Web Interface



Identificador: 1
Buscar por: Identificador de Usuario

Identificador de Cuenta	Saldo
1	0
2	200

Identificador de Cuenta	Saldo
3	1379

Figure 8. Example of Interaction through the MiniBank Web Interface

(3rd. ed.). California: O'Reilly.

Leman, J. (2010). *Programming Entiry Framework. Build Data Centric Apps with ADO.NET Entity Framework 4* (2nd. ed.). California: O'Reilly.

López, J. G. (2002). Motivación y autoaprendizaje: elementos clave en el aprendizaje y estudio de los alumnos. *Ensayos: Revista de la Facultad de Educación de Albacete* (17), 191-218.

Miguel Díaz, M. de, Alfaro Rocher, I., Apodaca Urquijo, P., Arias Blanco, J., García Jiménez, E. y Lobato Fraile, C. (2006). *Metodologías de enseñanza y aprendizaje para el desarrollo de competencias: orientaciones para el profesorado universitario ante el Espacio Europeo de Educación Superior.* Madrid: Alianza Editorial.

Prieto Navarro, L. (2007). *Autoeficacia del profesor universitario: eficacia percibida y práctica docente.* Narcea Ediciones.

Torre Llorente, C. de la, Zorrilla Castro, U., Calvarro Nelson, J. y Ramos Barroso, M. A. (2011). *Guía de Arquitectura N-Capas orientada al Dominio .NET 4.0.* Madrid: Krassis Press.

Zeldman, J. y Marcotte E. (2009). *Designing with Web Standards.* San Francisco: New Riders.

Attachment I
Development Frameworks (2nd Submittal)

Group Code: Dev.Fr.____	**Date:** ____	Observations:
Student 1		
Student 2		
Student 3		

CHECKPOINT	OK	COMMENTS
Repository		
1. Practical assignment submitted on, deadline?		
2. Correct repository structure		
Corrections Submittal 1 (note the corrections and solutions adopted)		
3. Is the entity model now correct?		
4. Does the practical assignment compile and execute correctly? [Deactivate cookies]		
Use Cases		
- Product		
5. Obtain all products [keyword search of product title]		
5.1. Allows to specify category as an option to restrict the search		
5.2. Search result includes:		
a) Name, registry date		
b) Category name and seller, DTO/DAO Call/ EagerLoad?		
c) Link “See comments”		
d) Link “Add comments”		
- User		
6. Register user [fields should be validated]		
7. Modification of registry information		
8. Authentication [with the possibility to remember password] and exit		
- Comment		
9. Add comment about the product		
a) Requires authentication		

b) [usability] Reports "Operation successfully completed" (optionally, it can redirect to the page "See comments" or show a link to that page). Other options may be valid.		
c) [improvement] Is the name of the product shown when commenting?		
d) Where is the commenting user ID obtained? SessionManager.Comment/SessionManager. GetUserSession ()/Codebehind access Session directly		
10. Obtain existing comments on a product		
a) Link "See comments" only shows when necessary (comments>0) ● 1 call per product/1 call per gridview		
b) Includes commenting user pseudonym		
c) How do you recover the pseudonym? Call to DAO–CustomVO–Navegac-service		
d) Does it have pagination support?		
- Ratings		
11. [improvement] Is the name of the product shown when rating?		
12. Rating includes number and justifying comment		
13. Each user can rate the same product once 13.1. How is the error indicated? Link disappears/ exception shown		
14. Own products may not be rated 14.1. How is the error indicated? Link disappears/ exception shown		
15. [usability] Reports "Operation successfully completed" (optionally, it can redirect to the page "See ratings" or show a link to that page)		
16. Rating view includes list of ratings, median and total 16.1. Name of rater can be seen (DTO/navigation/ CustomVO) 16.2. Is everything performed with only one model call?		
17. For each rating, date, user pseudonym, vote and text are shown		
- Favorite		
18. Add product to user's favorite list		
19. Obtain user's favorite products		
20. Delete favorite		
21. Others		

21.1. Link "See favorites" always visible for authenticated users		
21.2. Favorite name is a link to view details		
Optional: AJAX		
22. Where is it applied?		
23. Justification/knowledge of use mode		
24. Library used		
Optional: Cached search		
25. Cache creation (global.asax): ICacheManager cacheManager = CacheFactory. GetCacheManager();		
26. Servicio.Find() - Before a call against DAO, cache is verified object o1 = cacheManager.GetData("query");		
27. Cache update after new query		
28. Cache update after a registered query (to update result) cacheManager.Add("testkey2", "Some Text");		
29. Configuration in EL5.0 (expiry, numMax elements)		
Optional: tagging comments		
30. Add one or more tags on a comment		
30.1. Obtain existing tags		
31. Obtain comments related to a tag		
32. Comment view includes tag view		
33. It is possible to add/delete block tags		
33.1. Delete tags: shows previously added tags		
34. Tag cloud		
a) Tag is a link to the comments related to the tag		
b) Number of comments related to each tag is considered for size		

Usability	
35. What is the application's degree of usability? [What happens when a new comment is added? Is the user informed that the operation was completed successfully?]	fair, good, very good
Visual aspect	
36. [improvement] What does the application look like? (Usability aside)? [Are style pages used? Are icons used instead of links?]	fair, good, very good
37. Pagination ObjectDataSource / Next-Previous	

View (general aspect)		
38. There is no HTLM code in the controller		
39. Tag script not used in view layer		
Internationalization		
40. Use of Local Resources Files		
41. Use of Global Resources Files		
42. On what grounds are languages selected (user profile, search engine preferences, etc.) 42.1. Pages derive from SpecificCulturePage only when necessary		
Web.config		
*Safety: Access control to add rating page		
* Does the application work without cookies?		
Knowledge about setting options		
What facades does the application have? (User, Group, Comment, Favorite, Recommendation, Tag (Opt. 2) , Event)		
See SessionManager, have new methods been included?		
Report quality: [ ] Very bad, [ ] Bad, [ ] Average, [ ] Good, [ ] Very good		
Global impression: [ ] S, [ ] A, [ ] Not, [ ] Sob, [ ] MH		

Attachment II

Unified Evaluation Template of the Course

Call:	Dev.Fr.	
Group:	EXAM	Grade
Student 1:		
Student 2:		
Student 3:		

Basic part: 7 points	A1	
	A2	
	A3	

Design: 1.5 points				
□ Bad (0) Very serious design errors. Has not corrected mistakes of the first submittal. Class instructions have not been followed. MVC not followed. Usage cases missing.	□ Fair (0.40) Serious design errors. Has not corrected all mistakes of the first submittal. Class instructions have not been followed. MVC pattern not followed. Usage cases missing.	□ Average (0.80) No/minor design errors. Class instructions have been followed. Most use cases are present.	□ Good (1.2) The design is good. Class instructions have been followed. Some structure improvements are included. All usage cases are present.	□ Very good (1.5) The design is very good. Class instructions have been followed. Most structure improvements are included, possible in practice. Creation decision very well supported.
Notes:				

Implementation: 1 point				
□ Bad (0) The implementation contains **serious errors.** It does not work properly. Class instructions have not been followed. Method or attribute syntax is not correct. Exceptions not documented.	□ Fair (0.25) The implementation contains **many errors.** It does not work properly in some cases. Class instructions have not been followed. Method or attribute syntax is not correct. Exceptions not documented.	□ Average (0.5) The implementation contains **no serious errors.** It works properly Class instructions have been followed. Method or attribute syntax is correct (at least in most cases). Exceptions are documented.	□ Good (0.75) The implementation contains **no errors.** It works properly. Class instructions have been followed. Method or attribute syntax is correct. Exceptions are documented. Good implementation practices are followed.	□ Very good (1) The implementation contains **no errors.** It works properly. Class instructions have been followed. Method or attribute syntax is correct. Exceptions are documented. Good implementation practices are followed. Improvements are made. Performance issues are considered.
Notes:				

Defense 1.5 points					
A1	□ Bad (0)	□ Fair (0.40)	□ Average (0.80)	□ Good (1.2)	□ Very good (1.5)
A2	□ Bad (0)	□ Fair (0,40)	□ Average (0.80)	□ Good (1.2)	□ Very good (1.5)
A3	□ Bad (0)	□ Fair (0.40)	□ Average (0.80)	□ Good (1.2)	□ Very good (1.5)
	The student has very little knowledge of the concepts used in the practical assignment. He/she cannot answer questions asked.	The student has little knowledge of the concepts used in the practical assignment. He/she answers a few questions.	The student has acceptable knowledge of the concepts used in the practical assignment. He/she can answer questions asked.	The student has good knowledge of the concepts used in the practical assignment. He/she can answer questions asked with clarity. His explanations are supported.	The student has very good knowledge of the concepts used in the practical assignment. He/she can answer questions asked with clarity. His explanations are supported. The student has a very good knowledge of the concepts in the course.
Notes:					

Report: 1 point				
□ Bad (0) Report does not meet requirements. Lacking many sections. Redundant or bad quality information. Diagrams are not representative. Report makes no contribution to the correction of the practical assignment.	□ Fair (0.25) Report does not meet requirements. Lacking some sections. Redundant or bad quality information. Diagrams are not representative.	□ Average (0.5) Report meets requirements, including the following sections: Global architecture, model, graphic interface, a section for each additional part, known problems.	□ Good (0.75) Report meets requirements. Good writing. Facilitates correction of practical assignment.	□ Very good (1) Report meets requirements. Very good writing. Facilitates correction of practical assignment. Every important aspect is reflected in the report.
Notes:				
Sample quality: 0.5 points				
□ Bad (0)	□ Fair (0.2)	□ Average (0.3)	□ Good (0.4)	□ Very good (0.5)
Notes:				
Usability/Browsability 0.5 points				
□ Bad (0)	□ Fair (0.2)	□ Average (0.3)	□ Good (0.4)	□ Very good (0.5)
Notes:				
Improvements/Optimizations: 0.5 points				
□ Bad (0)	□ Fair (0.2)	□ Average (0.3)	□ Good (0.4)	□ Very good (0.5)
Notes:				
General functioning of the practical assignment (global perception): 0.5 points				
□ Bad (0)	□ Fair (0.2)	□ Average (0.3)	□ Good (0.4)	□ Very good (0.5)
Notes:				

Optional parts (3 points)

Comment tags (optional) Score: 1.5 points				**A1:**
				A2:
				A3:
□ Bad (0)	□ Fair (0.40)	□ Average (0.80)	□ Good (1.20)	□ Very good (1.5)
Notes:				
Search cache (optional) Score: 0.75 points				**A1:**
				A2:
				A3:
□ Bad (0)	□ Fair (0.20)	□ Average (0.40)	□ Good (0.60)	□ Very good (0.75)
Notes:				
AJAX (optional): Score: 0.75 points				**A1**
				A2:
				A3:
□ Bad (0)	□ Fair (0.20)	□ Average (0.40)	□ Good (0.60)	□ Very good (0.75)
Notes:				

© The authors. This article is being published by the Educational Quality Department's Research Area Revista Digital de Investigación en Docencia Universitaria, Universidad Peruana de Ciencias Aplicadas (UPC). This is an open-access article, distributed under the terms of the Attribution-ShareAlike 4.0 International Creative Commons License (http://creativecommons.org/licenses/by-sa/4.0/), which allows the non-commercial use, distribution and reproduction in any media, provided the original work is properly cited.