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

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

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

Recibido: 04/12/14 Aceptado: 01/05/15

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 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

---

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

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

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.

INTRODUCCIÓN

La asignatura de "Marcos de Desarrollo", es obligatoria en el 4.º curso del Grado en Ingeniería en Informática de la Universidade da Coruña. Su duración es cuatrimestral, teniendo asignados un total de 6 créditos ECTS. Estos últimos, se encuentran repartidos en 3 créditos para las sesiones expositivas en formato de clases magistrales y los otros 3 para la carga interactiva, lo cual se traduce en 21 horas de sesiones teóricas y 90 horas de trabajo práctico, por parte del alumno, contando con el trabajo estimado fuera de las horas de laboratorio. Es por ello que se está, fundamentalmente, ante una materia práctica que pretende, en su último curso, familiarizar a los futuros egresados con los ejercicios habituales que encontrarán, tras su entrada al mundo laboral.

La asignatura se presenta, dentro del plan de estudios, como complementaria con otra denominada "Programación Avanzada", en la cual los alumnos ven el framework J2EE, mientras que el trabajo que aquí se muestra, perfecciona su formación al demostrarles el punto de vista de las tecnologías .NET (Grimes, 2002).

En la asignatura no sólo se expone el punto de vista tecnológico, sino que también se presentan las reglas más relevantes y adecuadas del proyecto a la hora de operar con este framework y solventar problemas comunes dentro del crecimiento de aplicaciones web (Zeldman y Ethan, 2009). Así mismo, debido a esa complementariedad de la que se ha hablado, es posible resaltar las diferencias con el framework que los alumnos vieron previamente. Durante su evolución, se imparten una serie de tutoriales acerca de los conceptos necesarios para la realización de la práctica: patrones de diseño, tecnología, entorno de desarrollo, etc. Adicionalmente, se desarrolla y pone a disposición del alumnado una serie de aplicaciones web, a modo de tutoriales, en los que se detalla la implementación de los diferentes conceptos previamente mostrados: autenticación de usuarios, validación de datos,

gestión de transacciones, etc.

Este enfoque integrado, de presentación de la tecnología junto con los patrones de estructuración, viene principalmente motivado porque es imposible la adecuada aplicación de una técnica si no va ligada a una metodología de ingeniería basada en la correcta aplicación de patrones. Además, es sumamente complejo entender algunos procedimientos de manera aislada, si no se contextualizan dentro de los problemas que provocan la aparición del modelo de determinadas soluciones tecnológicas.

El estudio que aquí se expone, es presentado como heredero de la asignatura de Integración de Sistemas, de la antigua Ingeniería en Informática (Gestal, Rivero, Rabuñal, Dorado y Pazos, 2010). Cabe resaltar que sus contenidos han sido modificados para ajustarse al nuevo esquema temporal y formativo del Grado en Ingeniería en Informática.

Su evaluación viene determinada por la realización de una aplicación web; si bien es necesario obtener una nota mínima en un examen tipo test, en el que se verifica la correcta asimilación de los conceptos requeridos para la realización de la práctica.

Para el presente estudio, en primer lugar, se detallan los objetivos específicos de la asignatura, así como la descripción de la metodología docente. Además, se profundiza más en los conceptos explicados a los alumnos y los ejemplos utilizados para tal fin. Finalmente, se extrae una serie de conclusiones basadas en las valoraciones de los alumnos.

Objetivos de la asignatura

El tratado "Marcos de Desarrollo", se centra en la presentación de los patrones de diseño y estructurales para la formación e implementación de aplicaciones en el entorno web, con tecnologías .NET (Zeldman y Ethan, 2009).

A lo largo del curso, se desarrolla una práctica en la cual se implementa una aplicación web de características empresariales con .NET. Este trabajo, constituye la operativa normal en una aplicación real (aunque evidentemente limitada en cuanto a funcionalidades, debido a las restricciones de tiempo) de los conceptos mostrados durante la asignatura.

La correcta realización de la práctica, permitirá al alumno alcanzar los siguientes objetivos marcados como prioritarios para la materia:

- Conocer los fundamentos de programación mediante las tecnologías .NET
- Dominar los principios arquitectónicos fundamentales de las aplicaciones empresariales
- Saber técnicas de diseño para desarrollar aplicaciones empresariales (especialmente en lo que respecta a aplicaciones Web) mediante una arquitectura en capas

Para alcanzar estos objetivos, la materia cuenta con 2 créditos de sesiones magistrales y se centran en dos puntos fundamentales:

- Diseño e implementación de la capa modelo
- Diseño e implementación de la capa Web

Estos puntos se ven reflejados, desarrollados y experimentados durante el curso, en las prácticas realizadas para merecer los 4 créditos contemplados en la materia. Como se puede ver, los puntos detallan las dos partes principales del Model-View-Controller (MVC) (Gamma, Helm, Johnson y Vlissides, 1995). Este arquetipo se presenta en combinación con el patrón Layer (de la Torre, Zorrilla, Calvarro y Ramos, 2011) y constituye la base de las pautas arquitecturales en aplicaciones empresariales actuales. Como soporte para la explicación de estos y otros patrones habituales en el ámbito empresarial, se presentan dos aplicaciones de referencia, a modo de ejemplos profusamente

documentados y comentados. Estos dos ejemplos, que se analizarán en los siguientes apartados, tienen como objetivos, por un lado, facilitar la explicación de cómo se adaptan los prototipos a la hora de implementarlos en el framework, y, por otro lado, servir como base para el crecimiento posterior de la práctica.

METODOLOGÍAS PEDAGÓGICAS

Metodología docente

A pesar del enfoque eminentemente práctico de la asignatura, durante la concreción de la misma se hace también uso de sesiones magistrales. En ellas se muestran, de manera teórica, los conceptos básicos necesarios para la práctica, como por ejemplo la inyección de dependencias. Estos conceptos teóricos, se concretan posteriormente en cómo se implementan dentro del framework de avance. Así, se hace especial énfasis en las tecnologías de acceso a bases de datos y los patrones de diseño de software empleados en la formación de las capas modelo, vista y controlador de las aplicaciones empresariales.

La práctica desarrollada durante el curso se realizará en grupo, a fin de imitar la estructura típica de equipos de trabajo, que se encuentran habitualmente en el sector de adelanto de *software.* Este hecho, por otra parte, posibilita que los alumnos se familiaricen con los diseños y además con buenas prácticas de desarrollo de software en equipo, como puede ser el mantenimiento de un repositorio con versiones del *software,* la asignación de responsabilidades, la planificación y asignación de tareas y la aplicación de metodologías de trabajo.

La práctica, por su parte, intenta imitar el funcionamiento genérico de alguna de las aplicaciones web más comunes y de reconocido éxito empresarial (Betandwin, Amazon, etc.), aunque, evidentemente, con un número menor de funcionalidades debido al reducido tiempo disponible para su realización. El hecho de imitar aplicaciones sirve de motivador al alumnado, ya que comprueban de primera mano cómo se pueden crear dichas soluciones (Bugeja, 2007).

Debido a la extensión y complejidad de la práctica, la división de su entrega en iteraciones facilita que un mayor número de alumnos alcance los plazos pactados y se produzca un menor número de abandonos. Esto sigue la estrategia típica de evolución metodológica de proyectos *software* en espiral (Boehm, Madachy y Steece, 2000), con la que, además, se busca que los alumnos se familiaricen, ya que es una de las más empleadas en el mundo laboral.

Metodología de evaluación

El mayor peso de la materia, recae sobre el apartado práctico de la misma. Es por ello que, en este aspecto, se hace mayor hincapié a la hora de evaluar el trabajo hecho por los alumnos.

Tal y como se ha comentado, el tiempo de progreso de la práctica planteada durante el curso se divide en función de dos iteraciones o fechas de entrega. En la primera de estas iteraciones, la cual no acarrea una nota, se implementa la parte inicial. El objetivo de esta primera entrega es intentar garantizar que el alumno enfoca bien el desarrollo y que aplica los conceptos de manera adecuada. Para ello, el profesor intenta detectar errores importantes y, en ese caso, orienta al alumno hacia su resolución. En la segunda iteración, el alumno corrige los errores detectados en la primera y añade el resto de funcionalidades. Durante la corrección de esta segunda iteración, los alumnos deben explicar las funcionalidades implementadas, cómo se han dividido el trabajo, etc. Con ello, se les asigna una nota individualmente y que marca la mayor parte

ISSN: 2223 - 2516

de la calificación de la asignatura.

Para garantizar una corrección completa de la práctica, y ecuánime entre todos los grupos, se sigue el mismo esquema, basado en la adecuada realización de una serie de puntos de control. Esta corrección está guiada por un *checklist* o guión, que verificará una serie de puntos en la realización del trabajo, aunque evidentemente dependerá en algunos aspectos de la práctica concreta planteada cada año. A modo de ejemplo, puede consultarse uno de estos guiones con mayor detalle en el anexo I.

Adicionalmente, una vez corregida la práctica, el profesorado revisará con mayor detalle la calidad de la memoria entregada o del resto de documentación aportada. Para ello, basándose en las anotaciones del *checklist,* durante la corrección del trabajo, se cubrirá un documento simplificado que permita otorgar la valoración final en base a una serie de criterios fácilmente medibles, tanto de forma cualitativa como en modo cuantitativo. Para evitar la subjetividad en la evaluación, cada epígrafe corregido tiene anotadas las características que ha de contener la práctica del alumno, para ser merecedora de una u otra escala de evaluación. Este documento puede ser consultado en el anexo II.

.NET como medio de aprendizaje

El framework, desarrollado por Microsoft y conocido como .NET, surge en el año 2002 (Grimes, 2002) y se desarrolló a lo largo de más de una década. Desde un primer momento captó mucha atención, al plantearse como una firme alternativa a la propuesta de J2EE, al momento de implementar aplicaciones empresariales robustas, fiables y perdurables.

Este uso frecuente del framework, dentro de las aplicaciones empresariales, plantea el hecho de precisar que los alumnos, además de conocer y utilizar las soluciones J2EE, amplíen su abanico de posibilidades laborales mediante una correcta formación dentro de las soluciones ofrecidas por .NET.

Con este punto en mente se propone, como objetivo de la asignatura, explicar los patrones arquitecturales de las aplicaciones empresariales, utilizando para ello y como apoyo el framework .NET. Por tanto, lo que se sugiere es una introducción desde los modelos generales (Gamma et al., 1995), pero, posteriormente, se ve cómo implementar dichos conceptos generales, de manera real, en la arquitectura propia del framework de Microsoft (Zeldman y Ethan, 2009). Cabe remarcar que conocer el framework no es un objetivo prioritario, sino que este es el vehículo para que los alumnos comprendan mejor las ideas globales explicadas, al ver una concreción de las mismas.

De esta forma, para comprender los conceptos de los patrones explicados, se realiza una serie de pequeños ejemplos que, además, sirven como material de referencia posterior. Cada uno de estos, muestra el funcionamiento o implementación de un aspecto concreto de las funcionalidades y estructuras habituales dentro de las aplicaciones web y que, posteriormente, deberán ser implementadas en la práctica a desarrollar. Estos tutoriales se implementan bajo la filosofía de servir de guía de aprendizaje de la tecnología, dejando de lado aspectos como la eficiencia o la corrección del diseño. Este planteamiento viene derivado del hecho que la mayor parte de las tecnologías y prototipos explicados son nuevos para el alumno, por lo que intenta facilitarse, en la medida de lo posible, su comprensión.

Por tanto, al alumno se le entrega una serie de ejemplos en los que se muestran, en primer lugar, aspectos básicos del lenguaje

de programación C# (estructuras, gestión de excepciones, etc.). El tutorial incluye, también, ejemplos de conexión a bases de datos mediante ADO.NET (entorno conectado y desconectado), así como a través del ORM Entity Framework (Andrew y Schafer, 2006).

En la primera parte del curso, lo que se busca es que los alumnos adquieran soltura con un lenguaje y entorno de crecimiento que a la mayoría les es ajeno antes del comienzo de esta asignatura, pero que actualmente cuenta con una gran demanda e implantación en las empresas del sector informático. Es por ello que la complejidad de los tutoriales se va incrementando, a medida que avanza el curso.

Esta fase, de aprendizaje o adaptación, desde otros lenguajes conocidos previamente por el alumnado, suele desarrollarse de manera muy rápida y nada traumática, ya que, como se ha apuntado anteriormente, se trata de estudiantes de último año con un amplio bagaje de desarrollo en otros lenguajes y frameworks.

Se ha detectado una deficiencia en los alumnos, referente a la falta de formación en cuestiones relativas a la web. Así, a los alumnos se les provee de una lista de tutoriales que pueden hacer para solventar dichas carencias, como pueden ser la seguridad web, el conocimiento de ASP.NET, la visualización, el XML, etc. Una lista completa de tutoriales facilitados se puede ver en la Figura 1.

Al completarse esta etapa, se explica los patrones arquitecturales en los que se basa la mayor parte de las aplicaciones web comerciales, resaltando beneficios que éstas presentan. Todo esto se hace siempre al referir cómo se pueden implementar estas pautas en el framework, con lo que los alumnos ya lo están experimentando de primera mano, en las prácticas, como se concretan dichos conceptos. Debido a ese incremento, se posibilita atenuar la pendiente de la curva de aprendizaje del alumno, puesto que puede comprobar, sobre ejemplos funcionales, los conceptos requeridos posteriormente para la realización de la práctica.

La expansión de ejemplos facilita, además, la realización de las prácticas, puesto que el código de estos sirve de base para que se desenvuelvan las mismas. De esta manera, el alumno finalmente elabora una aplicación web compleja (y perfectamente usable) sin tener que codificarla en su totalidad.

Los ejemplos proporcionados, se centran principalmente en dos aplicaciones web completas, de reducido tamaño: MiniPortal y MiniBank. En ellas, se hace énfasis en los fundamentos de la concreción de aplicaciones web empresariales. Estas aplicaciones cubren diferentes aspectos (división en capas, autenticación, gestión transacciones, etc.), que posteriormente el alumno deberá aplicar para la realización de las prácticas. A continuación, se realiza una breve descripción de cada uno de estos materiales.

Implementación de la lógica de negocio con ADO.NET

El actual progreso de aplicaciones, se centra mucho en la separación de responsabilidades que posibiliten el aumento de las más robustas y de fácil mantenimiento. Para lograr dicha separación de responsabilidades, se hace uso de un patrón arquitectural conocido con el nombre de Layers (de la Torre LLorente et al., 2011). Dicho diseño argumentaría que es preciso agrupar bajo una misma capa (layer) todas aquellas funcionalidades propias de uno de los aspectos de la aplicación. En concreto, este prototipo se suele combinar con el Model-View-Controller (MCV) (Gamma et al., 1995), que requiere separar la lógica propia de la actividad empresarial de aquella

ISSN: 2223 - 2516

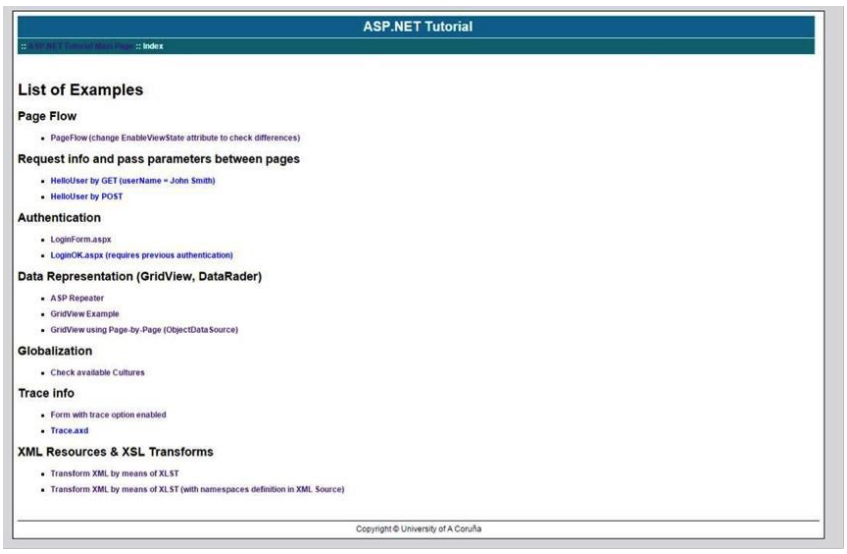

Figura 1. Tutoriales

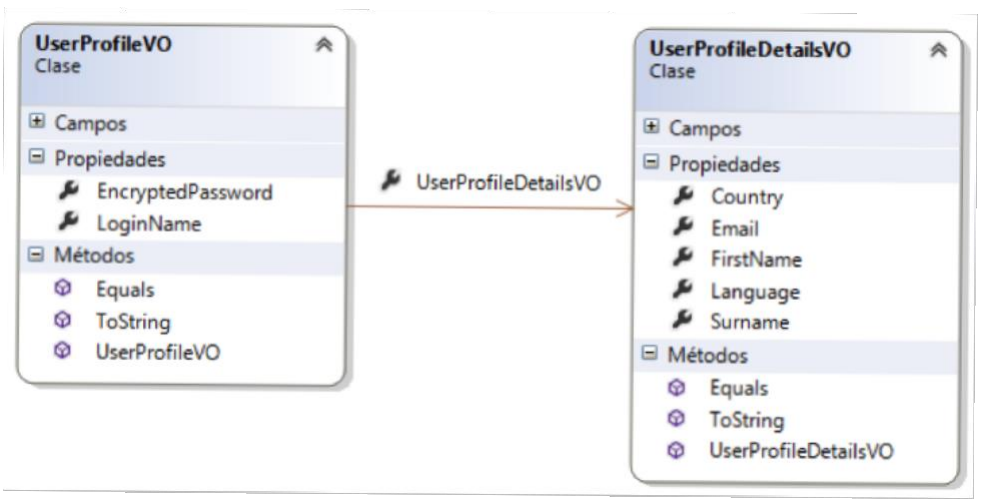

Figura 2. Objetos del Dominio en MiniPortal

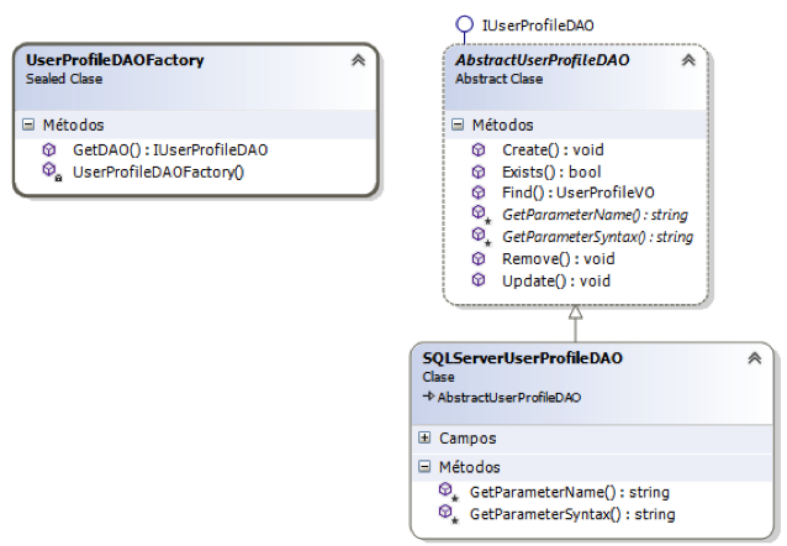

Figura 3. Arquitectura del Patrón DAO para el UserProfileVO

ISSN: 2223 - 2516

necesaria para la representación de la información contenida en la lógica de negocio. Estos conceptos se implementan mediante frameworks específicos, como el ORM Entity Framework. El problema que se encuentra, es que los términos manejados son demasiado conceptuales y alejados del bajo nivel. Es por ello que los ejemplos se ofrecen implementados en ADO.NET, aunque la materia se centra en el uso de Entity Framework (Leman, 2010), ya que es la tendencia que demanda el mercado. Por ello, al comienzo las prácticas inician su andadura mediante la implementación de la capa de lógica de negocio o capa modelo. Ésta constituye la capa base de la aplicación final. Así, los alumnos tendrían que implementar toda la lógica del negocio y la persistencia de los datos en una base de éstos, siguiendo el paradigma Layers. Es decir, tendrían que implementar dos capas: una para la persistencia y otra para la operativa de la empresa, de tal manera que los cambios en una afectasen mínimamente a la otra. Además de los mencionados patrones arquitecturales, los alumnos hacen uso de otros, como pueden ser distintos ejemplos de estructuración: el arquetipo fachada, el factoría, el plantilla, etc.

La capa de persistencia se implementa, en un primer momento, mediante una de las tecnologías clásicas del framework para la conexión a una base de datos, como es el protocolo ADO.NET. Para ello, se utiliza como ejemplo, tal y como se ha comentado anteriormente, la aplicación MiniPortal. Ésta es una aplicación web completa, que implementa un portal que permite el registro y autenticación de usuarios. En la Figura 2 se puede ver los principales objetos de negocio. Éste será uno de los requisitos que la práctica de los alumnos deberá soportar, por lo que el ejemplo proporcionado podrá ser reutilizado, con mínimos cambios.

Imagen de la arquitectura de un DAO

Estos objetos del dominio son persistidos, utilizando la arquitectura ADO.NET y los patrones Factoría y DAO. Para ello, ADO.NET nos provee de acceso a la información almacenada en una base de datos, con el fin de almacenar la información de los objetos del negocio. En este caso concreto, el UserProfileVO y el UserProfileDetailsVO. Además, ADO.NET provee de la funcionalidad DataSets, que permite reducir el tráfico de datos al poder traer los que describen la base de datos relacional y operar con ella antes de consolidar los cambios en una sola conexión. Mediante el uso del patrón DAO, los alumnos experimentan, de primera mano, cómo es posible separar completamente la lógica de conexión de la Base de Datos de las operaciones en la lógica de negocio, encapsulando por tanto la persistencia en una capa independiente, que puede sufrir alteraciones sin que éstas repercutan en la capa superior. La arquitectura para lograr dicha separación se puede ver en la Figura 3.

En MiniPortal, se muestra al alumno, de una manera práctica, la división de una aplicación empresarial siguiendo el Model-View-Controller (MVC). Dicho patrón, constituye la base para entender la separación en capas de las aplicaciones empresariales, así como de la escalabilidad de las mismas. Además, se ejemplifica el empleo de ciertos modelos de diseño (Session Facade, Business Delegate, Factory, etc.) para encapsular las funcionalidades de cada capa. Por ejemplo, en el caso del tipo, se hace uso del ejemplo Facade a fin de conseguir esa separación de las funcionalidades recogidas en los casos de uso (operaciones soportadas por la aplicación).

ISSN: 2223 - 2516

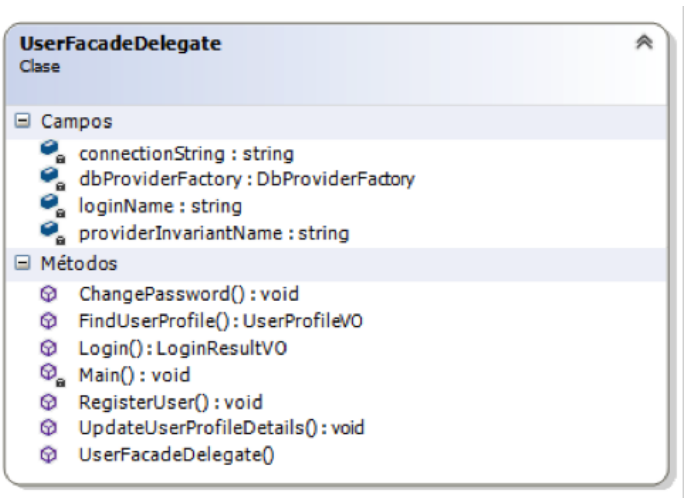

Figura 4. Fachada con los casos de uso de MiniPortal

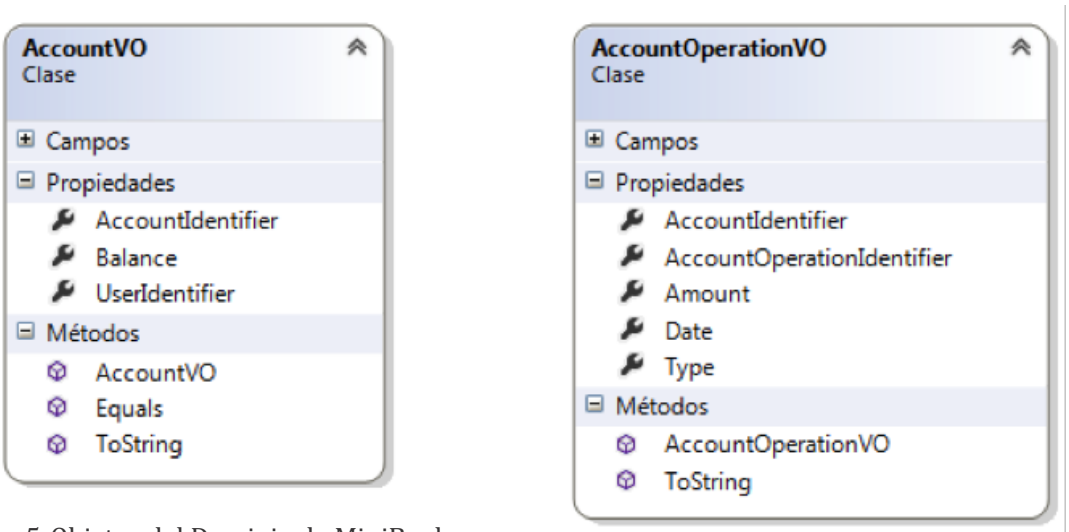

Figura 5. Objetos del Dominio de MiniBank

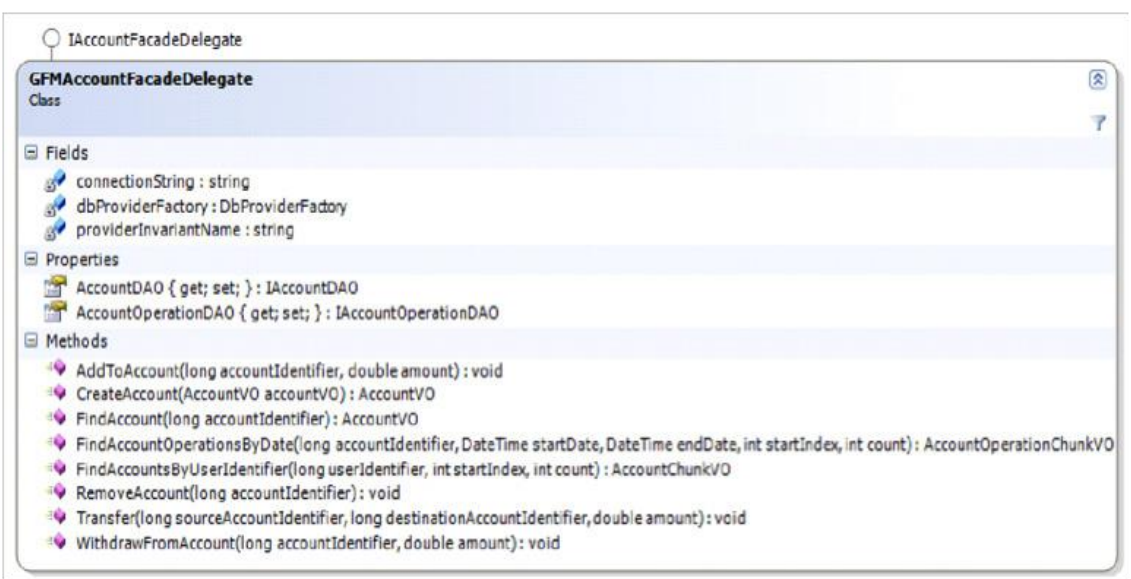

Figura 6. Fachada con los casos de uso de MiniBank

ISSN: 2223 - 2516

Dichos casos de uso para la aplicación MiniPortal, se muestran en la Figura 4. En la implementación de la parte-muestra, se hace especial énfasis en la gestión de los datos almacenados en la sesión (datos que han de ser accedidos en diferentes páginas). Además, se muestra cómo realizar el desarrollo de pruebas (para lo que se emplea TestProject, como herramienta dentro de .NET), empleo de parámetros configurables de manera externa, etc.

En el otro ejemplo completo que se entrega (aplicación MiniBank), se muestra al alumno una simplificación de la gestión bancaria de cuentas. Los objetos del dominio, que serán los que persistan en base de datos, se pueden ver en la Figura 5. MiniBank es una aplicación más completa y compleja que MiniPortal, ya que, aunque no tiene el alta de nuevos usuarios, sí cuenta con la parte de autenticación del usuario, que se vio en el ejemplo anterior. Además de esta autenticación, MiniBank contempla la operativa básica de operaciones sobre una cuenta corriente, es decir, creación, búsqueda, ingresos o retiradas de efectivo, así como transferencias entre distintas cuentas.

Al igual que en el caso anterior de MiniPortal, MiniBank hace uso de diferentes patrones de diseño y arquitecturales típicos de las aplicaciones empresariales, con el fin de obtener una separación de las distintas funcionalidades en capas. Al igual que Mini Portal, en lo que le corresponde, MiniBank hace uso de manera extensiva de los DAO y Facade para obtener una separación entre la lógica de negocio, la persistencia y el resto de funcionalidades. En la Figura 6, se muestra la Fachada del modelo con todos los casos de uso contemplados dentro de la aplicación.

Por su parte, Mini Bank trata de cubrir aspectos no contemplados en MiniPortal, y que el alumno ha de emplear en la realización de la práctica. Así, se muestran aspectos relativos a la generación automática de identificadores para el almacenamiento en bases de datos, gestión de transacciones o patrón Page-by-Page iterator (que permite la paginación de los resultados cuando estos son excesivos para ser mostrados de una sola vez).

Implementación de la vista y el controlador con ASP.NET

Una vez implementado el diseño, durante la segunda iteración de la práctica, los alumnos implementan la capa web de la aplicación, es decir, la presentación de los datos y la manera de interactuar con la lógica de negocio que han creado previamente. Para ello, se sigue el MVC y se pasa a implementar el controlador y la vista de la aplicación mediante el uso de ASP.NET. Además, se les introducen otra serie de buenas prácticas a la hora de desarrollar estas capas y estrategias (Andrew y Schafer, 2006).

Para ejemplificar los conceptos desarrollados en la vista y el controlador de la aplicación, en MiniPortal se muestran aspectos relativos a la navegabilidad entre páginas (ver Figura 7), gestión de perfiles, internacionalización, paso de parámetros entre páginas, llamadas a métodos del modelo, etc.

Uno de los principales aspectos que se resaltan aquí, son las diferentes arquitecturas de controlador que se pueden implementar aunque el comportamiento sea el mismo. Así, destacan como el controlador implementado con .NET sigue una arquitectura distribuida con una filosofía Page-Controller, mientras, en otros frameworks como Struts hay un único controlador genérico para toda la aplicación Application-Controller. En este punto, se hace hincapié en los beneficios e inconvenientes de cada arquitectura y en donde son más adecuadas una o la otra. Así mismo, se aprovecha para introducir facilidades relativas al desarrollo web

ISSN: 2223 - 2516

dentro del framework, como son la diferencia entre observar el panorama completo y las perspectivas parciales, la inclusión de Javascript (Harold y Means, 2004) para incluir dinamismo en la estructura, etc. Algunas de las tecnologías y conceptos que aquí se comentan son AJAX, JQuery, otros.

Por su parte, MiniBank es algo más sencilla, pero sirve para recalcarles a los alumnos los conceptos ya mostrados con MiniPortal. No obstante, sí es nuevo todo lo relativo a la paginación de resultados (ver Figura 8) mediante el arquetipo Page-by-Page, de uso más que frecuente en, por ejemplo, la presentación de resultados de una búsqueda.

Implementación del modelo con ORM Entity Framework

Una vez asimilados los conceptos en base a ADO.NET, a los alumnos se les explica cómo dichos conceptos se trasladan dentro del ORM Entity Framework (Leman, 2010), que será lo que tendrán que utilizar para implementar la persistencia. Por tanto, los estudiantes han ido desarrollando las otras partes del MVC con una persistencia temporal. Además, a los pupilos se les pide que ésta la sustituya por la capa correspondiente a la persistencia de datos, y la implementen con la tecnología ORM Entity Frameworks. Esta tecnología es una tendencia actual, la cual se basa en la inyección de dependencias. Por eso mismo, este momento es el que se utiliza para introducirles éste último concepto, en el cual se basan la mayor parte de las aplicaciones implementadas recientemente, debido a su flexibilidad y fácil mantenimiento.

Lo que se busca es que conozcan una de las tendencias actuales de crecimiento y que experimenten, de primera mano, un proceso habitual en el incremento informático, como es la migración de tecnologías. Además, este cambio pretende que se den cuenta de las ventajas que implica la separación en capas del diseño, ya que los cambios se limitarán a una parte, sin necesidad de tocar códigos y sin verse alterada la funcionalidad de las capas superiores.

EVALUACIÓN

Tal y como se ha comentado anteriormente, la asignatura tiene un marcado carácter práctico. Por lo tanto, la evaluación de la misma ha de ser obviamente basada en dicho aspecto. Para ello, tras la entrega de la práctica se produce la corrección de la misma. Ésta se realiza mediante la defensa, por parte del grupo, del trabajo entregado. Esta fundamentación incluye la verificación de la implementación de la práctica, para lo que se corroboran una serie de puntos críticos. Otro aspecto a comprobar, durante este proceso, es el correcto entendimiento por parte del grupo de los conceptos aplicados en el desarrollo de la actividad examinada. En función de la gravedad de los errores detectados, en caso de existir alguno, la aplicación Web se valorará con una puntuación de 0 a 10.

Adicionalmente, se realizará un examen tipo test. El objetivo de esta evaluación es comprobar que el alumno asimiló los conceptos correctamente. La prueba se compone de un conjunto de preguntas con varias respuestas posibles, de las que solamente una es correcta. Las preguntas no contestadas no puntúan, y las contestadas erróneamente lo hacen negativamente.

Para aprobar la asignatura es preciso: (1) tener aprobada la aplicación Web con un 5 sobre 10 y (2) sacar como mínimo 4,5 puntos (sobre 10) en el examen tipo test. En principio, el puntaje final de un alumno que cumpla con estas dos condiciones es la nota final, esta se compone de la suma ponderada de la nota de prácticas y la nota del test. Ante el carácter

ISSN: 2223 - 2516

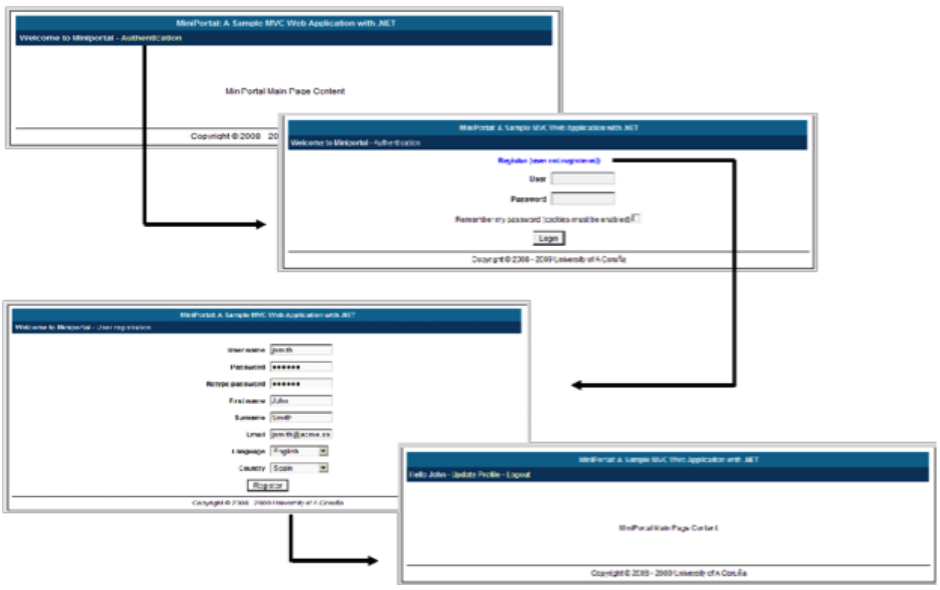

Figura 7. Ejemplo de interacción con el interfaz web de MiniPortal

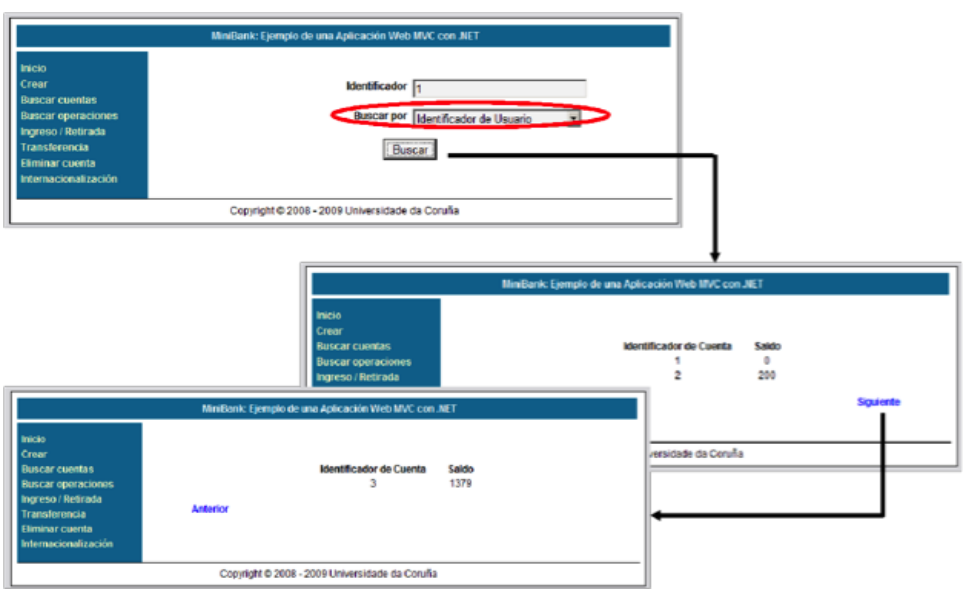

Figura 8. Ejemplo de interacción a través del interfaz web de MiniBank

ISSN: 2223 - 2516

práctico de la asignatura, la nota de prácticas supone el 60% de la nota final, mientras que el test únicamente pesa el 40%. Si bien a los profesores de la asignatura les gustaría que el peso de las prácticas fuese mayor, la normativa de la Universidad no lo permite.

CONCLUSIONES

A tenor de las encuestas del alumnado, la asignatura está entre las mejores valoradas de la titulación. Y ello, a pesar (o precisamente debido a esto) de ser una de las materias en las que la carga de trabajo es más elevada. Esto sucede debido a que el alumno, al realizar una aplicación web real (aunque con funcionalidad reducida), constantemente ve el fruto de su trabajo, lo que lleva a obtener una aplicación con la que es posible interactuar. La metodología explicativa, basada en el estudio de casos reales de aplicaciones, refuerza la formación del alumno, evitándole largas y áridas sesiones teóricas (De Miguel Díaz et al., 2006). Estos conceptos, expuestos en base a ejemplos que después ellos pueden y deben aplicar en el proyecto a desarrollar en prácticas, consolidan su asimilación. Por su parte, el explicar los conceptos en base a ejemplos provoca una mayor participación del alumnado, generando comentarios o sugerencias para el docente, el cual puede recoger e incorporar a la explicación con una retroalimentación de conocimiento. Dicho intercambio de nociones provoca en el docente un proceso de mejora continua, que se traduce en un avance en la calidad docente y una adaptación más rápida a la audiencia (Navarro, 2007).

Con respecto a los ejemplos realizados, podría decirse que su aceptación es elevada puesto que, analizando los logs de los servidores web en los que están ubicados, puede observarse que las consultas hechas por los alumnos son sólo una pequeña parte de todas las existentes.

Finalmente, el desarrollo de una práctica con aspecto y funcionalidad real, acogiéndose a los estándares de crecimiento de este tipo de proyectos en la industria, provoca una motivación extra en el alumno. Esto se debe a que el estudiante ve conceptos y genera contenidos que después son directamente aplicables a un graduado, por lo que se nota un mayor interés e implicación del educando. Esa mayor implicación, en muchos casos, se traduce en la incorporación del alumnado a un ciclo continuo de formación y reciclaje sobre los conceptos de la materia (López, 2002), el cual será muy beneficioso en su posterior vida laboral para no producir el efecto de obsolescencia (Rodríguez y Barrios, 2014).

REFERENCIAS

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

ISSN: 2223 - 2516

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* (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.

Anexo I

Marcos de desarrollo (2.ª iteración)

Cód. grupo: mad___	**Fecha: ___**	Observaciones:
Alumno 1		
Alumno 2		
Alumno 3		

CHECKPOINT	**OK**	**COMENTARIOS**
Repositorio		
1. Práctica entregada en ¿plazo?		
2. Estructura en repositorio correcta		
Correcciones Iteración 1 (anotar las correcciones y soluciones adoptadas)		
3. ¿El modelo de entidades es ahora correcto?		
4. ¿La práctica compila y se ejecuta correctamente? [Desactivar cookies]		
Casos de uso		
- Producto		
5. Obtener todos los productos [búsqueda por keyword del título producto]		
5.1. Permite especificar de manera opcional la categoría, para restringir la búsqueda		
5.2. Resultado búsqueda incluye:		
a) Nombre, fecha de alta		
b) Nombre categoría y vendedor ¿DTO / Llamada DAO / EagerLoad?		
c) Enlace "Ver comentarios"		
d) Enlace "Añadir comentario"		
- Usuario		
6. Registrar usuario [deben validarse los campos]		
7. Modificación de la información de registro		
8. Autenticación (con posibilidad de recordar contraseña) y salida		
- Comentario		
9. Añadir un comentario sobre un producto		
a) Solicita autenticación		

ISSN: 2223 - 2516

b) [usabilidad] Informa "Operación realizada correctamente" (Opcionalmente, puede redirigir a la página de "Ver comentario" o mostrar un enlace a esta página. Otras opciones pueden ser válidas)		
c) [mejora] ¿Se muestra el nombre del producto al momento de realizar el comentario?		
d) ¿De dónde se obtiene id del usuario que comenta? SessionManager.Comment/SessionManager. GetUserSession()/Codebehind accede a Session directamente		
10. Obtener los comentarios existentes sobre un producto		
a) Enlace "ver comentarios" sólo aparece si es necesario (comments>0) ● 1 llamada por producto / 1 llamada por gridview		
b) Incluye pseudónimo del usuario que realizó el comentario		
c) ¿Cómo se recupera pseudónimo? Llamada a servicio–DAO–CustomVO–Navegac		
d) ¿Soporta paginación?		
- Valoraciones		
11. [mejora] Se muestra el nombre del producto al momento de realizar valoración		
12. Valoración incluye número y comentario justificativo		
13. Cada usuario puede valorar una vez a un mismo producto 13.1. ¿Cómo se indica el error? Desaparece enlace/muestra excepción		
14. No se permite valorar los productos propios 14.1. ¿Cómo se indica el error? Desaparece enlace/muestra excepción		
15. [usabilidad] Informa "Operación realizada correctamente" (Opcionalmente, puede redirigir a la página de "Ver valoraciones" o mostrar un enlace a esta página.		
16. Visualización de valoraciones incluye la lista de valoraciones, su media y el total 16.1. Se muestra nombre del valorador (DTO/navegación/ CustomVO) 16.2. ¿Se realiza todo con una sola llamada al modelo?		
17. Para cada valoración se muestra fecha, seudónimo usuario que la realizó, voto, texto		

ISSN: 2223 - 2516

- Favorito		
18. Añadir producto a la lista de favoritos de un usuario		
19. Obtener los productos favoritos de un usuario		
20. Eliminar favorito		
21. Otros		
21.1. Enlace Ver favoritos siempre visible para usuarios autenticados		
21.2. Nombre favorito es un enlace a ver detalles		
Parte optativa: AJAX		
22. ¿En dónde se aplica?		
23. Justificación/conocimiento modo uso		
24. Librería usada		
Parte optativa: Cacheado búsquedas		
25. Creación caché (global.asax): ICacheManager cacheManager = CacheFactory. GetCacheManager();		
26. Servicio.Find() - Antes de realizar llamada contra el DAO se comprueba caché object o1 = cacheManager.GetData("query");		
27. Actualización caché tras una nueva consulta		
28. Actualización caché tras consulta que ya estaba dada de alta (para actualizar resultado) cacheManager.Add("testkey2", "Some Text");		
29. Configuración en EL5.0 (expiración, numMax elementos)		
Parte optativa: etiquetado de comentarios		
30. Añadir una o varias etiquetas sobre un comentario		
30.1. Obtener las etiquetas existentes		
31. Obtener los comentarios asociados a una etiqueta		
32. Visualización comentarios incluye visualización etiquetas		
33. Es posible añadir/eliminar etiquetas en bloque		
33.1. Eliminar etiquetas: muestra etiquetas previamente añadidas		
34. Nube de etiquetas		
a) Etiqueta es un enlace a los comentarios asociados a la etiqueta		
b) Se tiene en cuenta el número de comentarios asociados a cada etiqueta para tamaño		

Usabilidad	
35. ¿Cómo es el grado de usabilidad de la aplicación? [¿Qué sucede cuando se añade un nuevo comentario? ¿Se informa al usuario que la operación fue realizada correctamente?]	regular, bueno, muy bueno
Aspecto visual	
36. [mejora] ¿Cómo es el aspecto de la aplicación (al margen de la usabilidad)? [¿Se usan hojas de estilo?, ¿se muestran iconos en lugar de enlaces?,...]	regular, bueno, muy bueno
37. Paginación ObjectDataSource / Next-Previous	

Visualización (aspecto general)		
38. No hay código HTML en controlador		
39. No usan tag script en capa vista		
Internacionalización		
40. Empleo de Ficheros de Recursos Locales		
41. Empleo de Ficheros de Recursos Globales		
42. En base a qué se selecciona el idioma (perfil de usuario, preferencias navegador, etc.) 42.1. Páginas derivan de SpecificCulturePage sólo si es necesario		
Web.config		
* Seguridad: Control acceso a página añadir valoraciones		
* ¿Aplicación funciona sin Cookies?		
Conocimiento acerca de las opciones de configuración		
¿Qué fachadas tiene la aplicación? (User, Group, Comment, Favorite, Recommendation, Tag (Opc. 2), Event)		
Ver SessionManager. ¿Se han incluido nuevos métodos?		
Calidad Memoria: [ ] Muy Mala, [ ] Mala, [ ] Normal, [ ] Buena, [ ] Muy Buena **Impresión Global: [ ] S, [ ] A, [ ] Not, [ ] Sob, [ ] MH**		

Anexo II

Plantilla de evaluación unificada de la asignatura

Convocatoria:	MaD	
Grupo:	EXAMEN	Nota
Alumno 1:		
Alumno 2:		
Alumno 3:		

Parte básica: 7 puntos		
	A1	
	A2	
	A3	

Diseño: 1,5 puntos				
□ Malo (0) Fallos muy graves en el diseño. No ha corregido ningún fallo de la primera iteración. No se siguen las indicaciones dadas en clase. No se sigue el MVC Faltan casos de uso	□ Regular (0,40) Fallos graves en el diseño. No ha corregido todos los fallos de la primera iteración. No se siguen las indicaciones dadas en clase. No se sigue el patrón MVC Faltan casos de uso	□ Normal (0,80) No hay fallos en el diseño o estos son leves. Se siguen las indicaciones dadas en clase Está la mayor parte de los casos de uso	□ Bueno (1,2) El diseño es bueno. Se siguen las indicaciones dadas en clase. Se incluyen algunas mejoras de estructuración Están todos los casos de uso	□ Muy Bueno (1,5) El diseño es muy bueno. Se siguen las indicaciones dadas en clase. Se incluyen la mayoría de las mejoras de estructuración, posibles en la práctica. Las decisiones de creación están fuertemente argumentadas
Notas:				

Implementación: 1 punto				
□ Mala (0) La implementación contiene **errores graves**. No funciona adecuadamente. No se ha seguido las indicaciones dadas en clase. La sintaxis de métodos o atributos no es correcta. No se documentan excepciones	□ Regular (0,25) La implementación contiene **numerosos errores**. No funciona adecuadamente en algunos casos. No se ha seguido las indicaciones dadas en clase. La sintaxis de métodos o atributos no es correcta. No se documentan excepciones	□ Normal (0,5) La implementación **no contiene errores importantes**. Funciona adecuadamente. Se ha seguido las indicaciones dadas en clase. La sintaxis de métodos o atributos es correcta (al menos en la mayor parte de los casos). Se documentan excepciones	□ Buena (0,75) La implementación **no contiene errores**. Funciona adecuadamente. Se ha seguido las indicaciones dadas en clase. La sintaxis de métodos o atributos es correcta. Se documentan excepciones. Se siguen buenas prácticas de implementación	□ Muy Buena (1) La implementación **no contiene errores**. Funciona adecuadamente. Se ha seguido las indicaciones dadas en clase. La sintaxis de métodos o atributos es correcta. Se documentan excepciones. Se siguen buenas prácticas de implementación. Se realizan mejoras. Se tiene en cuenta cuestiones de rendimiento
Notas:				

Defensa: 1,5 puntos					
A1	☐ Mala (0)	☐ Regular (0,40)	☐ Normal (0,80)	☐ Buena (1,2)	☐ Muy Buena (1,5)
A2	☐ Mala (0)	☐ Regular (0,40)	☐ Normal (0,80)	☐ Buena (1,2)	☐ Muy Buena (1,5)
A3	☐ Mala (0)	☐ Regular (0,40)	☐ Normal (0,80)	☐ Buena (1,2)	☐ Muy Buena (1,5)
	El alumno tiene un conocimiento muy bajo de los conceptos utilizados en la práctica. No sabe responder a las preguntas formuladas	El alumno tiene un conocimiento bajo de los conceptos utilizados en la práctica. Responde sólo algunas preguntas	El alumno tiene un conocimiento aceptable de los conceptos utilizados en la práctica. Responde a las preguntas formuladas	El alumno tiene un conocimiento bueno de los conceptos utilizados en la práctica. Responde con claridad a las preguntas formuladas. Las explicaciones son fundamentadas	El alumno tiene un conocimiento muy bueno de los conceptos utilizados en la práctica. Responde con claridad a las preguntas formuladas. Las explicaciones son fundamentadas. Demuestra un conocimiento muy bueno de los conceptos vistos en la asignatura
Notas:					

Memoria: 1 punto				
☐ Mala (0) La memoria no cumple con lo solicitado. Faltan muchos apartados. La información es redundante o no es de calidad. Los diagramas no son representativos. La memoria no aporta nada para la corrección de la práctica	☐ Regular (0,25) La memoria no cumple con lo solicitado. Faltan algunos apartados. La información es redundante o no es de calidad. Los diagramas no son representativos	☐ Normal (0,5) La memoria cumple con lo solicitado, incluyendo los siguientes apartados: Arquitectura global, Modelo, Interfaz gráfica, un apartado para cada parte adicional, problemas conocidos	☐ Buena (0,75) La memoria cumple con lo solicitado. La redacción es buena. Facilita la corrección de la práctica	☐ Muy Buena (1) La memoria cumple con lo solicitado. La redacción es muy buena. Facilita la corrección de la práctica. Todos los aspectos importantes quedan reflejados en la memoria
Notas:				
Calidad de las pruebas: 0,5 puntos				
☐ Mala (0)	☐ Regular (0,2)	☐ Normal (0,3)	☐ Buena (0,4)	☐ Muy Buena (0,5)
Notas:				
Usabilidad/Navegabilidad: 0,5 puntos				
☐ Mala (0)	☐ Regular (0,2)	☐ Normal (0,3)	☐ Buena (0,4)	☐ Muy Buena (0,5)
Notas:				
Mejoras/Optimizaciones: 0,5 puntos				
☐ Mala (0)	☐ Regular (0,2)	☐ Normal (0,3)	☐ Buena (0,4)	☐ Muy Buena (0,5)
Notas:				
Funcionamiento general de la práctica (percepción global): 0,5 puntos				
☐ Mala (0)	☐ Regular (0,2)	☐ Normal (0,3)	☐ Buena (0,4)	☐ Muy Buena (0,5)
Notas:				

Partes opcionales (3 puntos)

<table>
<tr><td colspan="4">Etiquetado comentarios (opcional): Puntuación: 1,5 puntos</td><td>A1:</td></tr>
<tr><td colspan="4" rowspan="2"></td><td>A2</td></tr>
<tr><td>A3</td></tr>
<tr><td>☐ Malo (0)</td><td>☐ Regular (0,40)</td><td>☐ Normal (0,80)</td><td>☐ Bueno (1,20)</td><td>☐ Muy Bueno (1,5)</td></tr>
<tr><td colspan="5">Notas:</td></tr>
<tr><td colspan="4">Cacheado de búsquedas (opcional): Puntuación: 0,75 puntos</td><td>A1:</td></tr>
<tr><td colspan="4" rowspan="2"></td><td>A2</td></tr>
<tr><td>A3</td></tr>
<tr><td>☐ Malo (0)</td><td>☐ Regular (0,20)</td><td>☐ Normal (0,40)</td><td>☐ Bueno (0,60)</td><td>☐ Muy Bueno (0,75)</td></tr>
<tr><td colspan="5">Notas:</td></tr>
<tr><td colspan="4">AJAX (opcional): Puntuación: 0,75 puntos</td><td>A1:</td></tr>
<tr><td colspan="4" rowspan="2"></td><td>A2</td></tr>
<tr><td>A3</td></tr>
<tr><td>☐ Malo (0)</td><td>☐ Regular (0,20)</td><td>☐ Normal (0,40)</td><td>☐ Bueno (0,60)</td><td>☐ Muy Bueno (0,75)</td></tr>
<tr><td colspan="5">Notas:</td></tr>
</table>