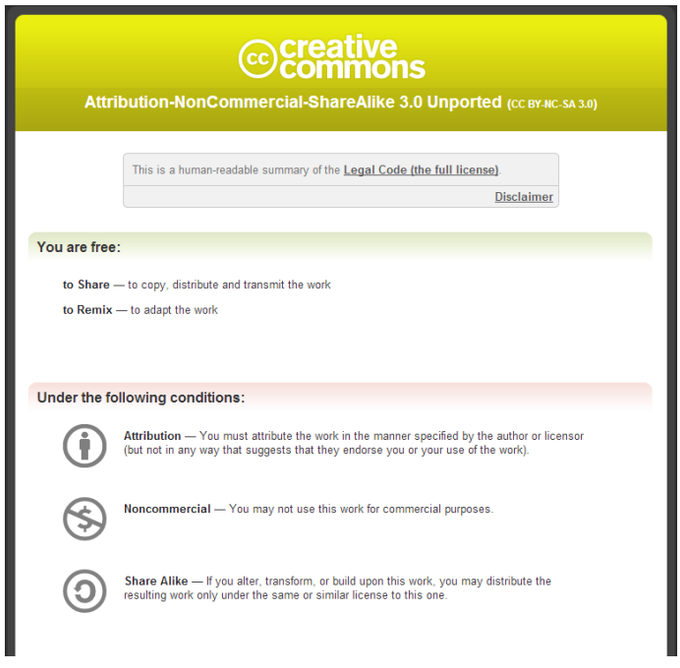
Aplicación Web para Formación Online

Memoria de Proyecto Final de Grado
Grado Multimedia
Ingeniería Web

Autor: Andrés Molina Orero
Consultor: Ignasi Lorente Purchades
Profesor: Carlos Casado Martínez

20 de Junio de 2013

Copyright



Dedicatoria

A Valeria. Sin su apoyo y paciencia no habría sido posible llegar hasta aquí.

Resumen

A lo largo de mis estudios, cada vez más me sentí atraído por el desarrollo puro como área de referencia. Esto me llevó a escoger el itinerario de ingeniería web pues las asignaturas que lo forman ensañan las técnicas propias de la ingeniería del software que permiten desarrollar aplicaciones tanto de escritorio como web de la forma más óptima. Es por esto que este proyecto representa la puesta en escena de todo lo visto al lo largo de mis estudios, pero con mayor énfasis en los conocimientos propios esas asignaturas como son el análisis y diseño orientado a objetos, la programación orientada a objetos, diagramación UML, aplicación de patrones etc; todos ellos encaminados hacia la obtención de software eficiente.

En un primer momento decidí usar Java como lenguaje de programación ya que está considerado como el paradigma en lenguajes basados en programación orientada a objetos. No obstante, después de recibir consejo sobre la dificultad de implementar una aplicación web en un framework tipo Hibernate, y teniendo conocimiento de que actualmente existen otros frameworks basado en PHP que soportan perfectamente la POO y están específicamente dirigidos al mundo web, opté por un cambio de tecnología, selecciónando entre los diferentes framwork Symfony2.

El desarrollo del mismo permitirá a los usuarios poder realizar seguimiento de cursos mediante la visualización de videos, previo contrato de tiempos determinados de acceso.

El proyecto tiene como objetivo desarrollar una aplicación web desde la que los usuarios podrán contratar los servicios de acceso que les permitan hacer un seguimiento de cualquier curso de cualquier categoría tecnológica. En Glosario se detallan los conceptos Categoría, Curso y Tema.

Palabras clave: Aplicación web, aplicación de escritorio, Symfony2, Yaml, Doctrine, Twig, Bundle, bases de datos, MySQL, ingeniería del software, análisis y diseño orientado a objetos, programación orientada a objetos, aplicación de patrones, UML

Convenciones

Categoría	Nombre de una tecnología, que por su amplitud forma un grupo propio.
Curso	Contenidos de formación sobre una determinada tecnología.
Tema	Unidad de información en forma de video que muestra parte del contenido de un curso
Symfony2	Framework de PHP especialmente dirigido a aplicaciones web y que permite la programación orientada a objetos.
Patrón MVC	Patrón Modelo - Vista - Controlador en el que se base el framework.
YAML	Modelo de anotación en la configuración de algunos archivos en la arquitectura de Symfony.
Doctrine	Conjunto de librerías que permite el mapeado entidad/tabla (ORM) así como la persistencia de los datos.
Twig	Conjunto de librerías que permiten el desarrollo de plantillas para el renderizado de las vistas.
Form	Conjunto de librería que permiten la generación de formularios vinculados a una entidad.
Mapeado	Relación entre entidad (objeto) y su correspondiente tabla(s) para la persistencia en base de datos.
Firewalls	Configuración de seguridad destinada a solicitar las credenciales al usuario que intenta acceder a una ruta protegida.
Acces_control	Definición de rutas protegidas.
Providers	Sistema empleado en la provisión de usuarios para realizar la autenticación de los mismos.
Encoders	Sistema de encriptación empleado en la contraseña de los usuarios.
Patter	Patrón de la ruta a la que accede el usuario.

Contenidos

1. Introducción 10
   1.1. Justificación y contexto 10
   1.2. Estructura de la memoria 10
2. Descripción 11
3. Objetivos 13
   3.1. Objetivos principales 13
   3.2. Objetivos secundarios 13
4. Metodología 14
   4.1. Beneficios del desarrollo iterativo 14
   4.2. Buenas prácticas del UP 14
5. Planificación 15
   5.1. Tecnología 15
   5.2. Evaluación del riesgo 15
   5.3. Conclusión 16
   5.4. Calendario 16
   5.5. Documento de requisitos 16
   5.6. Caso de uso CU01 Contratar acceso 18
   5.7. Caso de uso CU02 Registro 19
   5.8. Caso de uso CU03 Confirmar Pago 20
   5.9. Caso de uso CU04 Login 21
   5.10. Caso de uso CU05 Renovar suscripción 22
   5.11. Caso de uso CU06 Cancelar Suscripción 23
   5.12. Caso de uso CU07 Consultar estado cuenta 24
   5.13. Caso de uso CU08 Cambiar password 25
   5.14. Caso de Uso 09 Consultar temas 26
6. Arquitectura y diseño 27
   6.1. Perfiles de usuario 27
   6.2. Contenidos. Pantallas CU01, CU02 y CU03 27
   6.3. Driagramas UML 31

7. Desarrollo 33
7.1. Plataforma de desarrollo 33
7.2. Ventajas de Symfony 36
7.3. APIs utilizadas 37
7.4. Test 45
7.5. Bugs 47
8. Proyecciones de futuro 53
9. Conclusiones 54
Anexo 1. Entregables 56
Anexo 2. Librerías 57
Anexo 3. Resumen 58
Anexo 4. Bibliografía 59

Imágenes y tablas

Glosario 6
Iteraciones en UP 14
Planificación 16
Pantalla Registro 27
Pantalla Login 27
Pantalla DatosyUsuarioContrato 28
Pantalla DatosPago 28
Pantalla RegistrarPago 29
Pantalla PagoCancelado 29
Pantalla UsuarioIncorrecto 30
Diagrama casos de uso actor principal: usuario 31
Diagrama de clases modelo dominio. Primera versión 31
Arquitectura Modelo - Vista - Controlador MVC 33
Estructura arquitectura Symfony2 34
Relación Objeto - Modelo relacional 34
Esquema de trabajo Symfony2 35
Estructura tablas base de datos 35
configuración de seguridad 37
Código formulario login 38
Caso 1 de autenticación 39
Caso 2 de autenticación 40
Caso 3 de autenticación 41
Caso 4 de autenticación 42
Código archive UsuarioType 43
Método que llama a la vista para generar el formulario 44
Método que llama a la vista para generar el formulario 44
Parte del controlador que envía el formulario si es válido 44
Aquí genera el formulario 44
Código para pruebas generar los cursos 45
Aquí genera el formulario 45
Código de pruebas para fechas caducadas 46
Ejemplo de código de una plantilla Twig 47
Parte del código de la entidad Usuario. 48
Detalle de la portada. Permite seleccionar contrato 49
Pantalla datos contrato y Registro 50
Pantalla datos contrato y usuario 50
Pantalla de login 51
Pantalla de cursos disponibles 51
Pantalla contrato caducado 52

1. Introducción

1.1. Justificación y contexto

Desde el año 2000 trabajo como formador en cursos para desempleados. Ahora, dados los recortes que se están produciendo en todos los sectores sociales, este tipo de formación también se está viendo muy afectado, de tal forma que en la actualidad apenas si se desarrollan cursos específicos para este colectivo.

Estas circunstancias me llevaron a pensar que es el momento oportuno de crear mi propia plataforma de formación online que me permita continuar con la labor de formador, ahora para todo tipo de usuarios y no solo desempleados.

Además esto coincide con la etapa final de mi carrera de Grado en Multimedia. Así pues no dudé en solicitar como TFG el desarrollo de la citada aplicación.

La idea fundamental es la creación de una plataforma que cubra ambos objetivos. Por una parte crear mi entorno de desarrollo profesional con esta plataforma y por otra practicar con los conocimientos adquiridos a lo largo de la carrera, pero especialmente con los del área de especialidad, ingeniería web.

El objetivo de mi proyecto es el de aplicar los conocimientos de análisis, diseño e implementación.

El desarrollo completo queda:

- Aplicación web que permita a la usuarios contratar el servicio de acceso durante un periodo determinado de tiempo. La aplicación se desarrolla con un Framework ágil específico para las aplicaciones web.

1.2. Estructura de la memoria

El resto de la memoria detalla los productos, los objetivos buscados, la metodología empleada así como su planificación. También se muestra otros apartados más técnicos referidos a la arquitectura y desarrollo.

Cierra la misma las conclusiones, así como una serie de anexos de información adicional

2. Descripción

Se trata de construir una plataforma que permita a cualquier usuario poder hacer el seguimiento de cursos referidos a diferentes tecnologías. Los temas de estos cursos son capturas de video mediante la correspondiente herramienta informática como Camtasia. Estos estarán formados por varios apartados, cada uno de ellos desarrollado en varios videos. Los temas se agrupan por categorías tecnológicas, por ejemplo:

Categoría: Microsoft.NET

Tema: Acceso a datos

Video 1. Establecimiento de conexiones.

Video 2. Ejecución de comandos

Video 3. DataReader

Tema: DataSets

Video 1. Ejecutar un Dataset

Video 2. Leer un Dataset

.

.

.

Los conceptos de Categoría, Curso y Tema se definen en el apartado Convenciones.

Este proyecto tiene como objetivo llevar a cabo el análisis, diseño e implementación de un sistema basado en dos productos utilizando para ello las técnicas aprendidas a lo largo de la carrera con especial interés en el área de la ingeniería del software como son el análisis y diseño orientado a objetos, programación eficiente basada en POO y la aplicación de patrones

Se trata de, mediante el uso de una determinada tecnología que lo permita, aplicar en el desarrollo del software los principios de diseño como son:

- Bajo acoplamiento

  Aplicar técnicas dirigidas a minimizar el acoplamiento, es decir la dependencia entre los elementos de la aplicación como clases, paquetes, etc.

- Alta cohesión

  Conseguir el mayor grado posible entre las diferentes responsabilidades de una clase. Lograr la mayor relación de estas responsabilidades

- Abierto - cerrado

  La entidad software debe estar abierta a la extensión, pero cerrada a la modificación. Para añadir nuevas responsabilidades al sistema iremos añadiendo nuevas clases, pero no modificando las que ya hemos desarrollado

- No repetición

  Conseguir mediante este principio la no duplicación de responsabilidades en diversas partes del sistema. Esto nos permite conseguir aplicaciones más fáciles de mantener ya que ante un cambio o error podemos identificar fácilmente cual es el componente afectado.

- Sustitución de Liskov

  Este principio nos indica que en la jerarquía tiene que respetarse que las instancias de las subclases puedan ser reemplazadas por instancias de las superclases.

- Segregación de interfaces

  Separar la interfaz de las clases en subconjuntos de operaciones con el objetivo de evitar el acoplamiento de esas clases.

- Inversión de dependencias

  Maximizar la reutilización de las clases evitando el acoplamiento con respecto a las clases de más bajo nivel.

Aplicando estas técnicas se desarrollan los siguientes productos:

- Aplicación web que permite a los usuarios contratar periodos de tiempo durante los cuales podrá acceder y ver cualquier curso disponible en la plataforma de cualquier categoría

3. Objetivos

3.1. Objetivos principales

- Poner en práctica conocimientos adquiridos a lo largo de la carrera, especialmente aquellos que pertenecen al área de especialidad Ingeniería Web.
- Obtener un alto conocimiento sobre la arquitectura de aplicaciones web mediante un framework específico.
- Obtener un alto conocimiento sobre las APIs que forman el framework seleccionado.
- Conocer la instalación de todos los componentes para el stack necesario según la opción tecnológica elegida.

3.2. Objetivos secundarios

- Aprender otras herramientas informáticas fundamentales, como MagicDraw para la diagramación UML.
- Diferenciar de forma clara las características técnicas de las posibles soluciones de implementación.

4. Metodología

Este proyecto se realiza siguiendo las bases del Proceso Unificado (UP). Es decir, está organizado en un inicio que estará destinado al desarrollo de casos de uso, aplicación de patrones, creación de diagramas tanto de modelo de negocio como de diseño etc. Una vez desarrollado el modelado de los objetos realizar dos iteraciones. En la primera de ellas se implementa el 40% de los casos de uso. En la segundo el 60% restante más las pruebas de test y actualización de la memoria

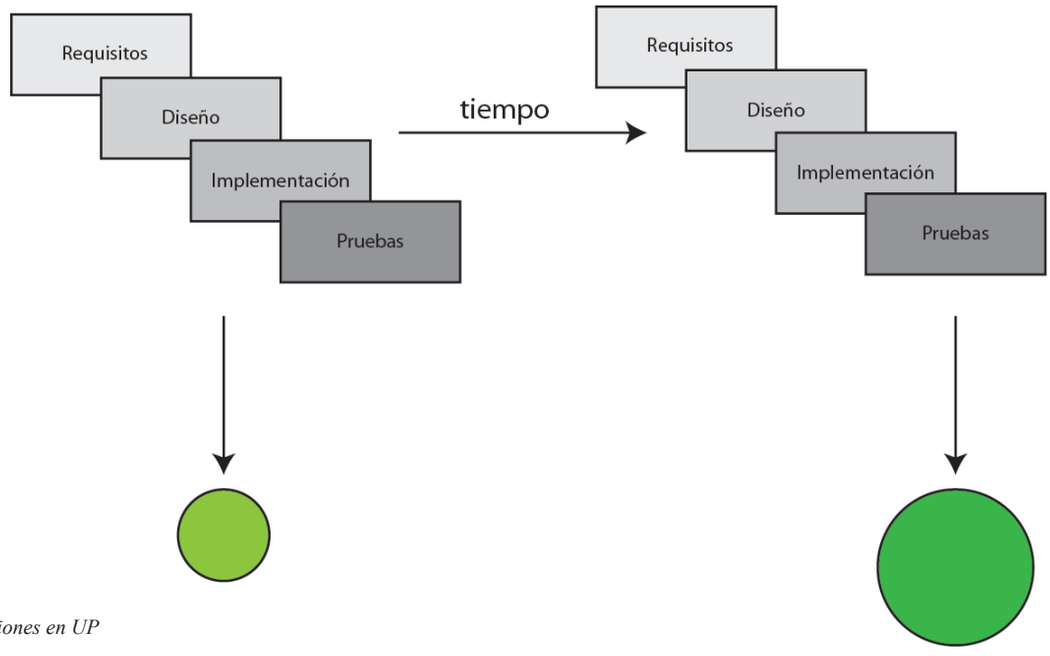
Este método de trabajo en ciclos de vida iterativo se basa en la ampliación y refinamiento sucesivos mediante múltiples iteraciones, con retroalimentación cíclica y adaptación como elementos que terminan convergiendo en el sistema adecuado.

4.1. Beneficios del desarrollo iterativo

- Mitigación tan pronto como sea posible de riegos altos (técnicos, requisitos, objetivos, usabilidad etc.)
- Progreso visible en las primeras etapas.
- Retroalimentación y adaptación tempranas.
- El conocimiento de cada iteración se puede utilizar en la mejora del propio proceso de desarrollo

4.2. Buenas prácticas del UP

- Abordar cuestiones de alto riesgo y valiosas en las primeras iteraciones.
- Construir en las primeras iteraciones una arquitectura que constituya un núcleo central consistente.
- Involucrar continuamente a los usuarios para evaluación, retroalimentación y requisitos.



Iteraciones en UP

5. Planificación

En un principio, la idea para el proyecto era la de desarrollar tres productos diferentes. Por un lado una aplicación de escritorio que permitiría al administrador gestionar los cursos, añadiendo categorías, temas, etc. Por otro una aplicación web que es el apartado principal del proyecto ya que permite a los usuarios contratar periodos de acceso al campus para poder hacer el seguimiento de los cursos. Finalmente también consistía en desarrollar una aplicación para dispositivos móviles.

Teniendo en cuenta la poca experiencia sobre el desarrollo de aplicaciones para dispositivos móviles y la magnitud del proyecto, se llega a la conclusión de que es muy posible que exceda las posibilidades reales de llevar el mismo a buen puerto. Así se decide unificar las tecnologías a una sola, aplicación web, que por su uso más intensivo a lo largo de los estudios es mas familiar y con menos costes de aplicación en tiempo. Esta decisión se detalla aún más en los tres puntos siguientes.

5.1. Tecnología

En principio estaba previsto desarrollar el proyecto en Java, ya que a lo largo de las asignaturas del área ingeniería web en las que tuvimos la oportunidad de aprender técnicas de ingeniería del software usamos este entorno como paradigma del mismo.

Pero, una vez recibido consejo sobre la posible dificultad que tendría implementar la aplicación por ejemplo con Hibernate y teniendo en cuenta que existen frameworks basados en PHP (tecnología a la que estoy más acostumbrado) y que llevan a cabo perfectamente todo lo relacionado con POO, decidí cambiar la tecnología por uno de estos framework, en concreto Symfony2.

5.2. Evaluación del riesgo

Aunque la tecnología PHP es más conocida para mi, el hecho de usarla como parte de un framework lleva consigo el dominio de otras herramientas clave y que es la primera vez que se toma contacto con ellas. Por ejemplo Doctrine, Security, Form etc. He incluso, el de otras herramientas que no son específicas de Symfony2 que permiten la gestión de dependencias, la publicación de las aplicaciones etc. Herramientas como Git, Composer, Capifony.

También son varias disciplinas conocidas las que se deben actualiza y revisar los conocimientos. Entre otras

- Pasar de XHTM a HTML5.
- Comprender cómo implementa PHP la POO.
- La propia filosofía del framework.
- El uso de base de datos desde un ORM como Doctrine

5.3. Conclusión

Asumir las carencias anteriormente mencionadas llevan a considerar necesario dos cambios importantes para el proyecto.

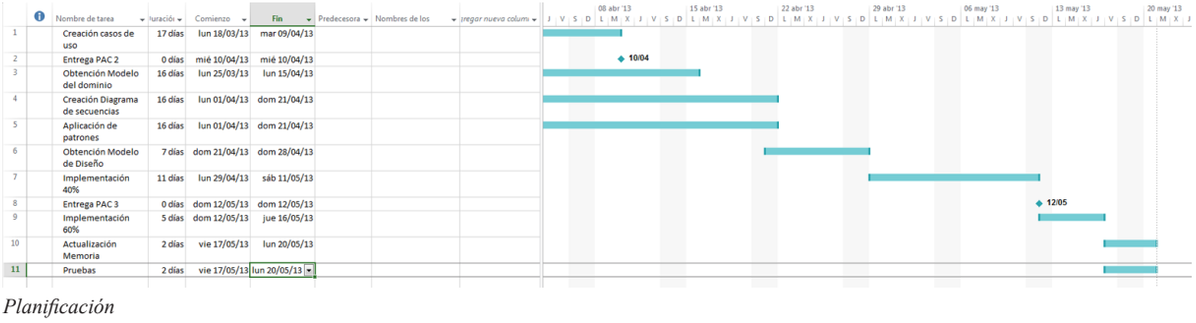
El primero en lo concerniente al alcance del mismo. La amplitud del proyecto era claramente excesiva. Agradezco en este sentido la intervención de mi consultor Ignasi Lorente que pronto me hizo ver esto. Para facilitar la viabilidad del proyecto se reestructura implementado la aplicación web para los usuarios (es el apartado principal) y dejar para ampliaciones futuras tanto la parte de administrado (gestión de cursos) como la de implementar una aplicación app para dispositivos móviles.

El segundo la tecnología a emplear. Si bien en un principio la decisión era la de usar el entorno Java, al detectar que no es la solución ideal, y comprobar sin embargo que ahora PHP si cuenta con varios framework que permiten implementar todas las ventajas que se deseaban llevar a cabo con Java, cambiar a esta tecnología para el desarrollo del proyecto y de forma más concreta con el framework Symfony2 al entender que este reúne lo mejor de otros framework.

5.4. Calendario

Según lo especificado en el apartado anterior, se sigue un proceso iterativo. En concreto se hacen 3 iteraciones de 4 semanas cada una de ellas. Cada iteración cuenta con sus apartados de requisitos, diseño, implementación y pruebas.

Además existe una primera etapa de documentación y puesta en marcha y una etapa final de pruebas y actualización de la memoria. La planificación temporal es la siguiente



	Nombre de tarea	uración	Comienzo	Fin	Predecesora	Nombres de los	¡regar nueva column
1	Creación casos de uso	17 días	lun 18/03/13	mar 09/04/13			
2	Entrega PAC 2	0 días	mié 10/04/13	mié 10/04/13			
3	Obtención Modelo del dominio	16 días	lun 25/03/13	lun 15/04/13			
4	Creación Diagrama de secuencias	16 días	lun 01/04/13	dom 21/04/13			
5	Aplicación de patrones	16 días	lun 01/04/13	dom 21/04/13			
6	Obtención Modelo de Diseño	7 días	dom 21/04/13	dom 28/04/13			
7	Implementación 40%	11 días	lun 29/04/13	sáb 11/05/13			
8	Entrega PAC 3	0 días	dom 12/05/13	dom 12/05/13			
9	Implementación 60%	5 días	dom 12/05/13	jue 16/05/13			
10	Actualización Memoria	2 días	vie 17/05/13	lun 20/05/13			
11	Pruebas	2 días	vie 17/05/13	lun 20/05/13			

Planificación

5.5. Documento de requisitos

La plataforma permitirá a los usuarios poder contratar espacios de tiempo (1 mes, 6 meses, 1 año) de acceso a la misma. Durante este tiempo, el usuario podrá visualizar cualquier curso de cualquier categoría sin ninguna restricción.

Los cursos se organizan por categoría -> curso -> tema (cada tema es un video). La definición se especifica más en el apartado Convenciones. Por ejemplo

- Categoría: Java
    - Curso: Introducción a Java
        - Tema: Introducción a las aplicaciones en Java
        - Tema: Introducción a las clases y objetos

        - Tema: Instrucciones de control
        - Tema: Métodos. Un análisis más detallado
        - Tema: Arreglos
    - Curso: Programación Orientada a Objetos con Java
        - Tema: Herencia
        - Tema: Polimorfismo
        - Tema: Interfaces
        - Tema: Clases abstractas
- Categoría: UML
    - Curso: Casos de uso
        - Requisitos
        - Actores principales y secundarios
        - Objetivos de los actores
        - Diagramas de casos de uso
        - Documento especificación complementaria
        - Glosario
    - Curso: Modelo de dominio
        - Identificación de las clases conceptuales
        - Guías para el modelo de negocio
        - Notación UML

5.5.1 Especificaciones de usuario

- Consultar los cursos de una categoría. El usuario podrá acceder a los detalles de un curso como su título, la duración del video y una descripción sobre el mismo.
- Consultar los temas de un curso. Al seleccionar un tema, el usuario accede a una lista con todos los cursos referentes a ese tema.
- Contratar un espacio de tiempo de acceso a los cursos. El usuario podrá contratar el acceso a la plataforma durante un rango de tiempo determinado. Durante este tiempo podrá acceder a todos los temas de cualquier curso y categoría.
- Visualizar los videos de los temas durante el espacio de tiempo contratado.

- Ver el estado de su cuenta.
- Renovar la suscripción.
- Cancelar la suscripción.
- Cambiar la contraseña.

5.5.2 Como sistema

- Evitar que los videos sobre cada tema puedan ser descargados.
- Controlar que el acceso de los usuarios es correcto, es decir, se produce dentro del rango de fechas contratadas

5.6. Caso de uso CU01 Contratar acceso

Identificador	CU01
Nombre	Contratar acceso
Resumen	Permite al usuario contratar el acceso al campus durante un rango de tiempo determinado
Actor	Usuario
Precondiciones	No hay ninguna sesión activa
Postcondiciones	El usuario ha seleccionado un tipo de contrato, se ha registrado y aceptado el pago
Flujo normal	1. El usuario accede al sistema por primera vez, o cuando finaliza sesión. 2. El sistema muestra los distintos tipos de contratos disponibles. 3. El usuario selecciona un tipo de contrato 4. El sistema muestra los datos del contrato seleccionado y un formulario para que el usuario se registre 5. El usuario introduce sus datos y se registra. 6. El sistema muestra los datos del contrato y del usuario y solicita los datos de la tarjeta. 7. El sistema acepta el pago y envía al usuario a login para acceder al campus
Flujos alternativos	El usuario pude cancelar la operación en los puntos 4 y 5. El sistema regresa a portada. Los datos de tarjeta no son aceptados. Regresa a portada

Inclusiones	CU02 Registro. CU03 Confirmar Pago.
Extensiones	Ninguna.

5.7. Caso de uso CU02 Registro

Identificador	CU02
Nombre	Registro
Resumen	Permite al usuario registrarse en la plataforma para poder hacer un contrato de tiempo de acceso
Actor	Usuario
Precondiciones	El usuario ha seleccionado un tipo de contrato de acceso
Postcondiciones	El usuario queda registrado una vez aprobado el pago
Flujo normal	1. El sistema muestra los datos del contrato seleccionado y un formulario para realizar el registro 2. El usuario introduce los datos solicitados 3. El sistema prepara la información para grabarla una vez el pago se haya confirmado.
Flujos alternativos	El usuario pude cancelar la operación en el punto 2. El sistema regresa a portada.
Inclusiones	Ninguna.
Extensiones	Ninguna.

5.8. Caso de uso CU03 Confirmar Pago

Identificador	CU03
Nombre	Confirmar Pago.
Resumen	Permite al usuario confirmar el pago del contrato seleccionado.
Actor	Usuario
Precondiciones	El usuario ha seleccionado un tipo de contrato. El usuario ha rellenado el formulario de registro
Postcondiciones	El usuario queda grabado en el sistema para acceder al campus
Flujo normal	1. El sistema muestra un formulario con los datos de la tarjeta. 2. El usuario introduce los datos. 3. El sistema conecta con la pasarela bancaria para aceptar el pago. 4. El sistema envía al usuario a login para acceder al campus.
Flujos alternativos	El usuario pude cancelar la operación en los puntos 1 y 2. El sistema regresa a portada. Los datos de tarjeta no son aceptados. Se muestra un mensaje al usuario y se regresa a portada
Inclusiones	Ninguna.
Extensiones	Ninguna.

5.9. Caso de uso CU04 Login

Identificador	CU04
Nombre	Login
Resumen	Permite al usuario autenticarse ante el sistema para iniciar sesión.
Actor	Usuario
Precondiciones	No hay ninguna sesión activa
Postcondiciones	El usuario accede al campus.
Flujo normal	1. El sistema muestra formulario con usuario / password 2. El usuario introduce sus datos. 3. El sistema verifica que los datos son correctos 4. El sistema verifica que la fecha de fin de acceso al campus es correcta 5. El usuario accede al campus.
Flujos alternativos	Los datos introducidos no son correctos. Se muestra mensaje para que lo intente de nuevo. La fecha de acceso está caducada. El usuario accede una página que le permite renovar el contrato
Inclusiones	Ninguna.
Extensiones	Ninguna.

5.10. Caso de uso CU05 Renovar suscripción

Identificador	CU05
Nombre	Renovar suscripción
Resumen	Permite al usuario realizar un nuevo contrato de acceso al campus.
Actor	Usuario
Precondiciones	El usuario se ha autenticado. La fecha de acceso al campus ha caducado.
Postcondiciones	El usuario tiene de nuevo acceso al campus
Flujo normal	1. Se ejecuta CU04 Login 2. El sistema muestra información sobre la caducidad del contrato actual 3. El sistema muestra los diferentes contratos disponibles. 5. El usuario selecciona un tipo de contrato 6. Se ejecuta el caso de uso CU03 Confirmar Pago. 7. El sistema acepta el pago y envía al usuario al campus
Flujos alternativos	El usuario pude cancelar la operación en los puntos 4 y 5. El sistema regresa a portada. El usuario cancela la sesión y se regresa a portada
Inclusiones	CU04 Login. CU03 Confirmar Pago
Extensiones	Ninguna

5.11. Caso de uso CU06 Cancelar Suscripción

Identificador	CU06
Nombre	Cancelar Suscripción
Resumen	Permite al usuario cancelar el contrato de acceso.
Actor	Usuario
Precondiciones	El usuario se ha autenticado. La fecha de acceso al campus está vigente
Postcondiciones	El usuario no tiene acceso al campus
Flujo normal	1. Se ejecuta CU04 Login 2. El sistema muestra información sobre la cuenta del usuario. 3. El usuario indica que desea cancelar el contrato. 5. El usuario confirma la cancelación
Flujos alternativos	El usuario pude cancelar la operación de cancelación y regresar al campus.
Inclusiones	CU04 Login.
Extensiones	Ninguna

5.12. Caso de uso CU07 Consultar estado cuenta

Identificador	CU07
Nombre	Consultar Estado Cuenta
Resumen	Permite al usuario conocer las fechas de acceso al campus de su contrato
Actor	Usuario
Precondiciones	El usuario se ha autenticado.
Postcondiciones	Ninguna
Flujo normal	1. Se ejecuta CU04 Login 2. El sistema muestra información sobre la caducidad del contrato actual
Flujos alternativos	
Inclusiones	CU04 Login.
Extensiones	Ninguna

5.13. Caso de uso CU08 Cambiar password

Identificador	CU08
Nombre	Cambiar Password
Resumen	Permite al usuario cambiar la contraseña de acceso al campus
Actor	Usuario
Precondiciones	El usuario se ha autenticado.
Postcondiciones	El usuario accede al campus con la nueva contraseña
Flujo normal	1. Se ejecuta CU04 Login 2. El usuario indica que desea cambiar la contraseña. 3. El sistema pide la contraseña actual y la nueva contraseña. 5. El usuario confirma el cambio. 6. El sistema registra el cambio.
Flujos alternativos	El usuario pude cancelar la operación de cambio en el punto 3.
Inclusiones	CU04 Login.
Extensiones	Ninguna

5.14. Caso de Uso 09 Consultar temas

Identificador	CU09
Nombre	Consultar Temas
Resumen	Permite al usuario consultar los temas de un curso
Actor	Usuario
Precondiciones	Ninguna
Postcondiciones	El usuario accede a un listado con todos los temas de los cursos disponibles en el campus
Flujo normal	1. El usuario indica que desea ver los cursos disponible. 2. El sistema muestra los cursos disponibles, y por cada curso los temas que lo forman.
Flujos alternativos	
Inclusiones	Ninguna.
Extensiones	Ninguna.

6. Arquitectura y diseño

6.1. Perfiles de usuario

Usuario anónimo: Accede a cualquier parte abierta de la aplicación.

Usuario registrado: ROLE_USUR

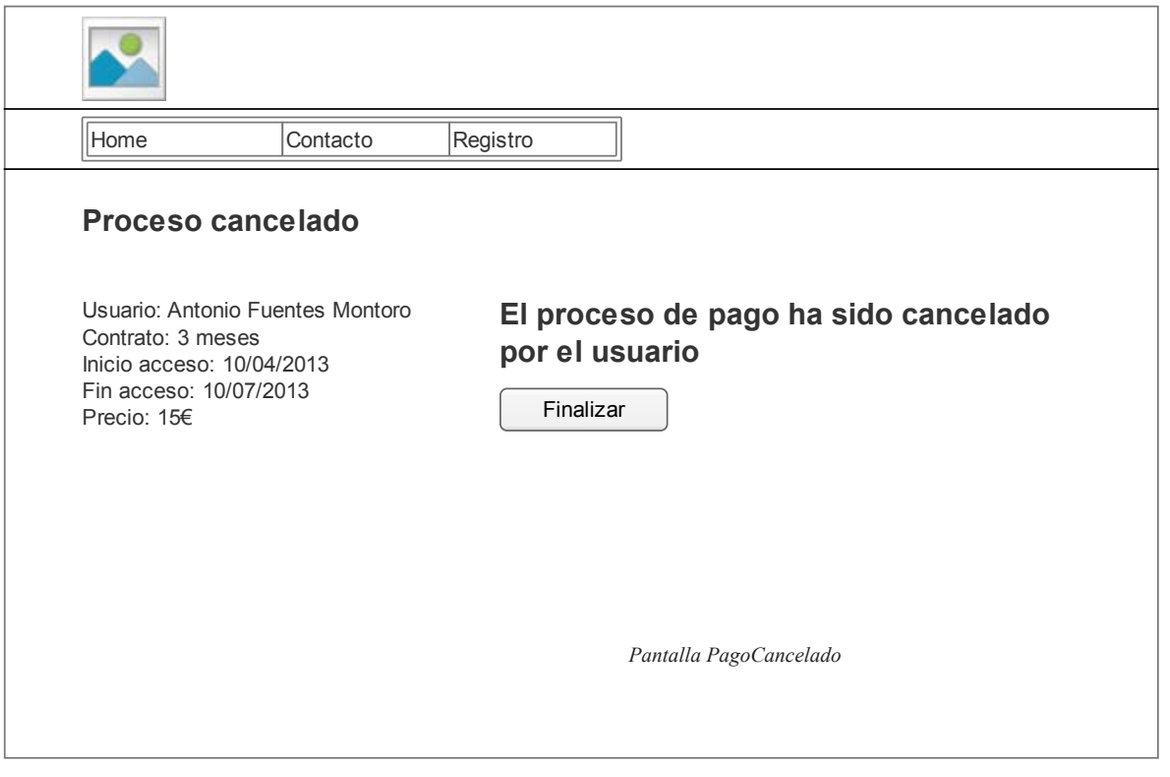
Administrador: ROLE_ADMIN

6.2. Contenidos. Pantallas CU01, CU02 y CU03

Pantalla DatosyUsuarioContrato

Pantalla DatosPago

Pantalla RegistrarPago



Pantalla PagoCancelado

Pantalla UsuarioIncorrecto

6.3. Driagramas UML

6.3.1 Diagrama casos de uso

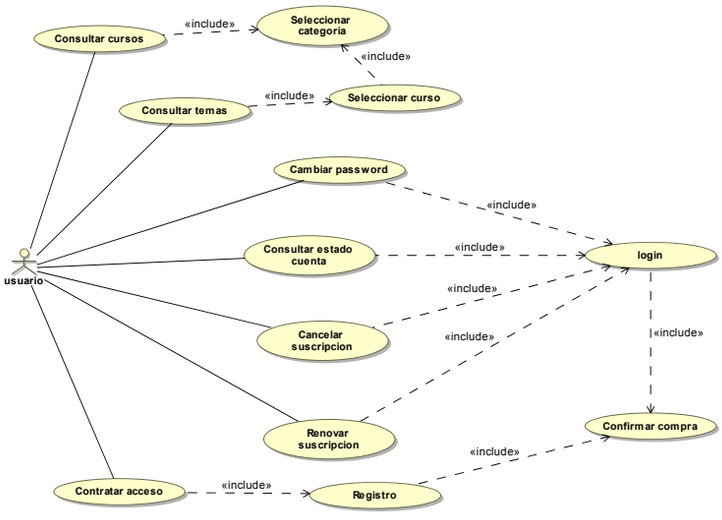


Diagrama casos de uso actor principal: usuario

6.3.2 Diagrama de clases modelo de dominio.

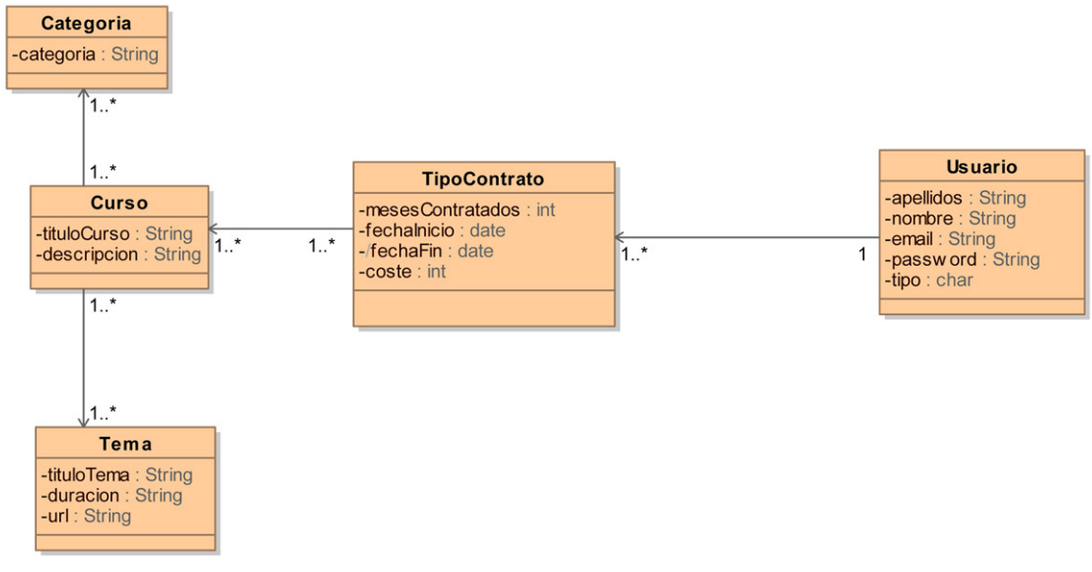
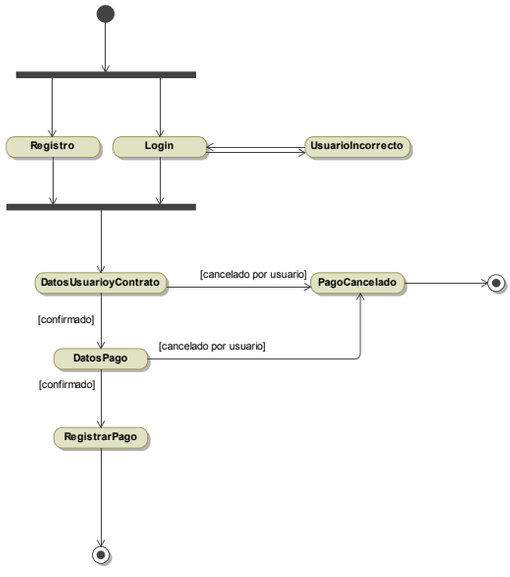


Diagrama de clases modelo dominio. Primera versión

6.3.3 Diagrama estados casos de uso CU01, CU02 y CU03



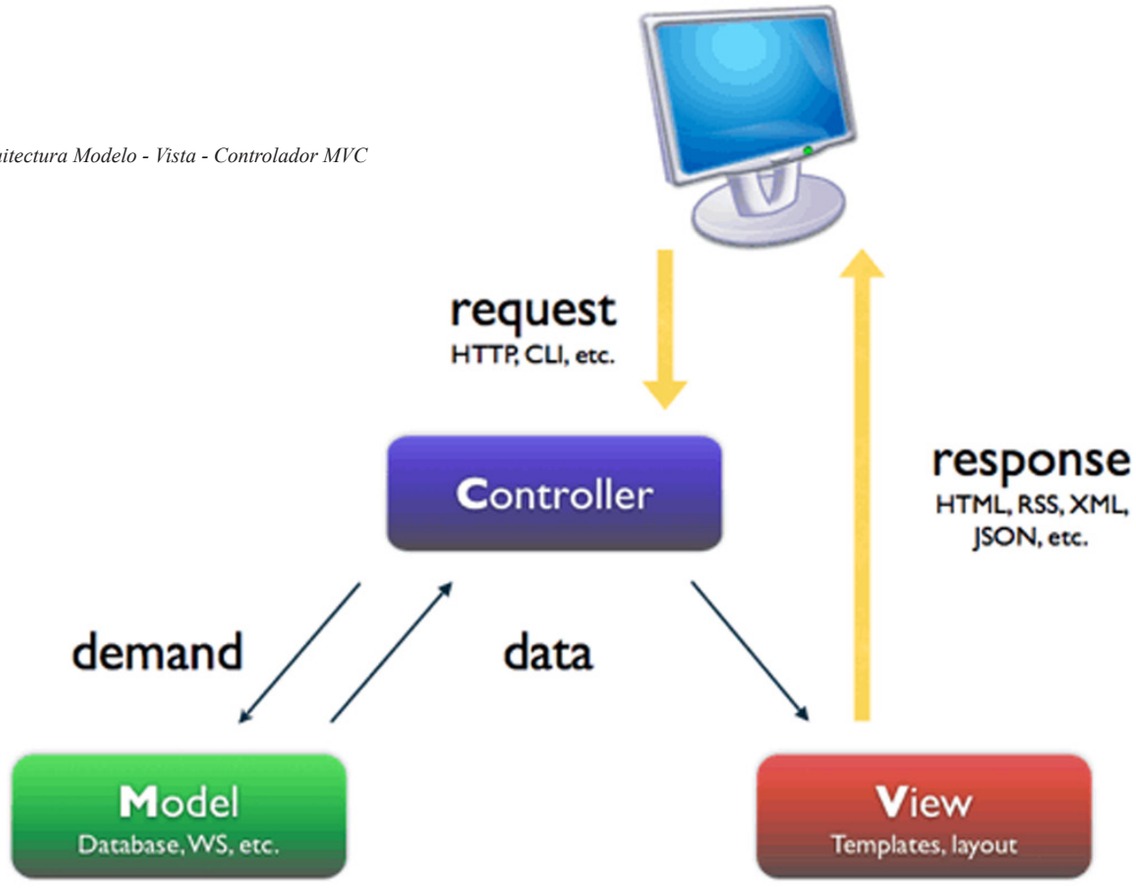
7. Desarrollo

7.1. Plataforma de desarrollo

Después de estudiar detenidamente las propiedades de algunos frameworks basados en PHP, finalmente he optado por desarrollar en Symfony2. Los motivos que me llevan a ello es el pleno soporte de las técnicas de POO.

Symfony2 es un framework basado en la arquitectura Modelo - Vista - Controlador.

Arquitectura Modelo - Vista - Controlador MVC

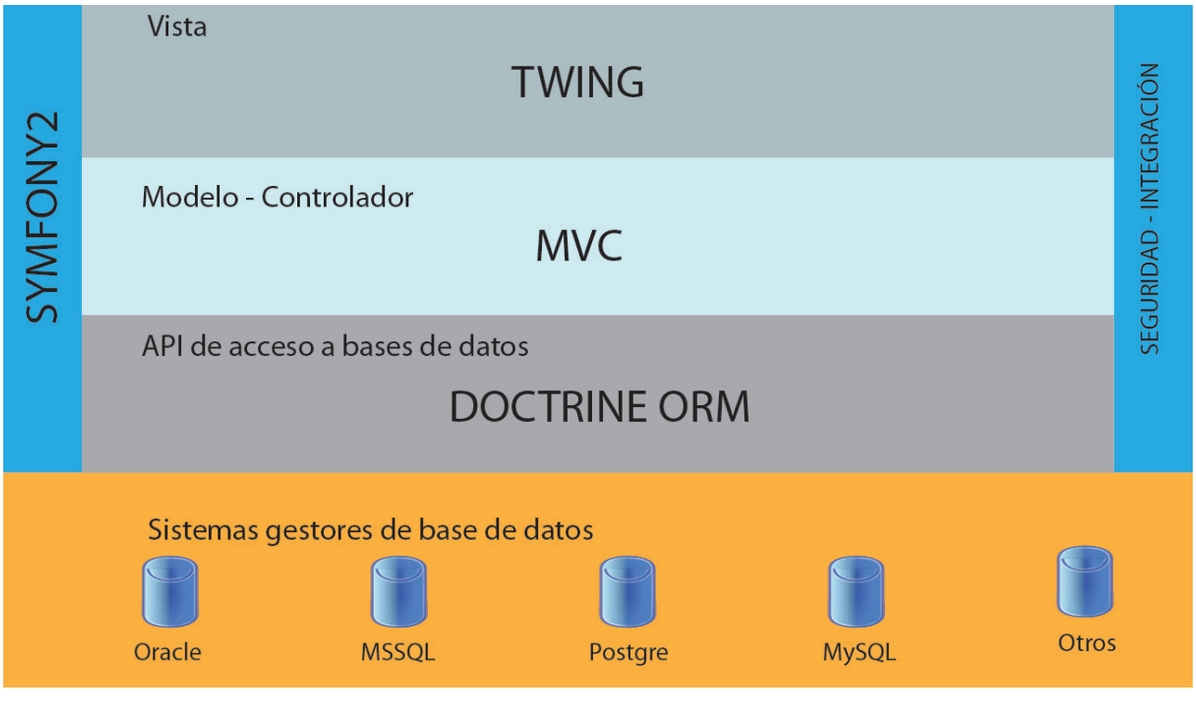


La arquitectura interior de Symfony2 (se muestra en la imagen anterior) está formada por los siguientes apartados.

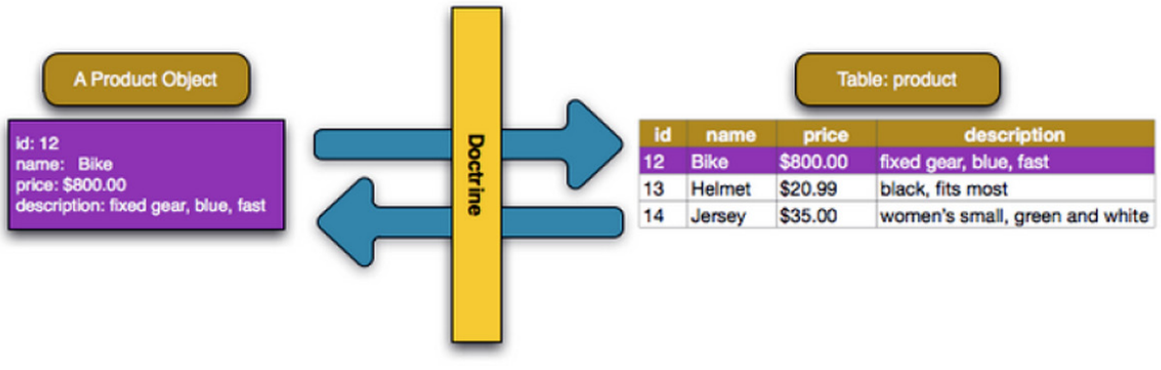
- En la parte de presentación, para las vistas, usa Twig que permite crear plantillas de forma fácil e intuitiva facilitando el trabajo a maquetadores y diseñadores frontend. Una de las ventajas de Twig es la herencia que implementan las plantillas. Esto permite a los diseñadores reutilizar diseños de forma fácil. Symfony incorpora una primera plantilla base de la que pueden heredar las demás, y que marca la estructura de las páginas. Está implementada bajo el estándar HTML5 y CSS3

- Como API de acceso a datos usa Doctrine. Este es un ORM que permite la integración entre un modelo relacional y orientado a objetos. Permite el acceso a cualquier tipo de base de datos y cuenta con un potente mapeador entre las tablas relacionales y los objetos de las entidades. Cuenta con su propio lenguaje de consulta muy próximo a SQL pero mucho más cómodo para implementar de cara al desarrollador.

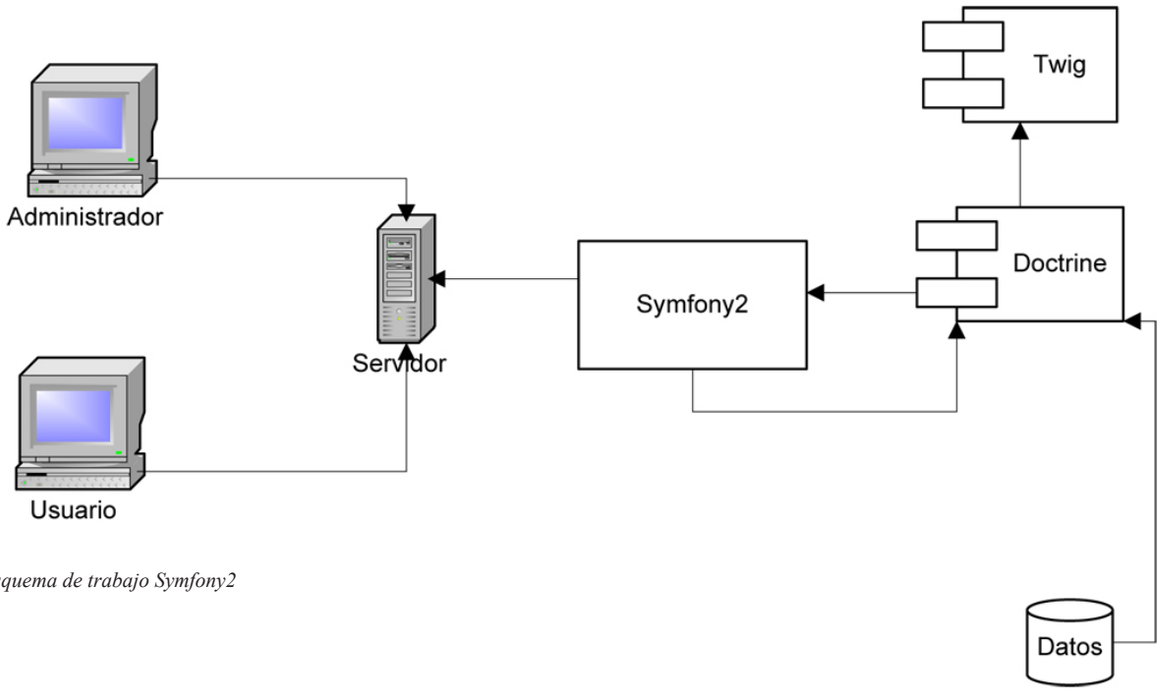
  Otro aspecto interesante en lo que a la creación de las entidades se refiere es que desde órdenes de consola podemos crear la base de datos, así como las tablas. Además también podemos indicar en cada campo el comportamiento del mismo a la hora de validar la información que contiene cada propiedad. Esto es importante para facilitar entre otras cosas la validación de formularios de forma automática.



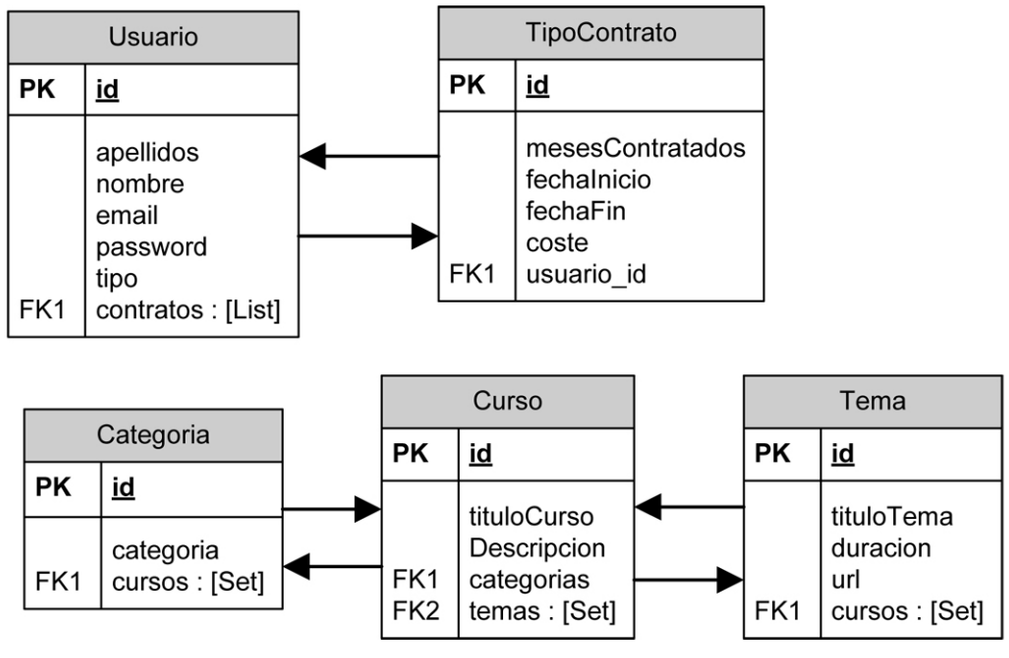
Estructura arquitectura Symfony2



Relación Objeto - Modelo relacional



Esquema de trabajo Symfony2



Estructura tablas base de datos

7.2. Ventajas de Symfony

Symfony2 ofrece una amplia variedad de ventajas para los desarrolladores. A destacar

- Algunos frameworks ofrecen un pila full-stack monolíticos como Cake php. Otros están basados en componentes como ZF. Otros, dirigidos a pequeñas aplicaciones ofrecen micro framework como Fuel.

  Symfony2 es todo lo anterior. Ofrece todas las estructuras mencionadas. En principio, el proyecto nace con 21 librerías propias. A estas se añaden otras librerías externas como Assetic, Doctrine etc. Y para integrarlo todo, se crean los bundles.
- Incorpora las mejores ideas de otros frameworks como Spring, Hibernate, Rails, Django.
- Los archivos de configuración ofrecen alta libertad de creación ya que puede estar escritos en diferentes formatos como Yaml, XML, PHP.
- Lo mismo ocurre con las plantillas. Cuenta con una poderosa herramienta Twig, pero aquellos que deseen seguir trabajando en PHP no tendrán problemas.
- Cuenta con un acceso a todo tipo de bases de datos.
- Tiene un alto rendimiento, pues toda la configuración se traduce a PHP y es mantenida en memoria caché.
- Una consulta a la base de datos solo se ejecuta una vez. Si la misma consulta vuelve a repetirse, no se vuelve a ejecutar sino que los datos son otra vez entregados gracias a un reverse proxy intermedio (Varnish) entre el usuario y la base de datos.
- Una ventaja muy interesante radica en la forma en la que se crean los formularios. Estos se llevan a cabo mediante archivos de tipo “Type” al cual se referencia una entidad existente. De esta forma los datos introducidos por el usuario en el formulario, una vez la vista lo ha renderizado son validados de forma automática, sin otro esfuerzo por parte del desarrollador.

Otras herramientas utilizadas en el proyecto:

- Axure RP Pro 6.5: prototipado
- Microsoft Proyect 2012: gestión del proyecto.
- Adobe Photoshop CS5 e illustrator CS5: recursos gráficos.
- MagicDraw: diagramación UML.
- Adobe Indesing CS5: maquetación.

7.3. APIs utilizadas

- Doctrine.

  Se encarga del mapeado entre las entidades y las tablas relacionales de la base de datos. Puede generar de forma automática tanto la propia base de datos como las tablas teniendo en cuenta las asociaciones implementadas entre las entidades.

  También permite realizar el logeado de forma rápida y fácil para el programador.

  Cuando el usuario intenta acceder a una ruta protegida por el archivo de seguridad es dirigido al formulario de login, que tiene unas características determinadas.

  Antes de ello, el archivo de seguridad habrá sido debidamente configurado.

File - C:\wamp\www\Proyecto\app\config\security.yml

```
jms_security_extra:
    secure_all_services: false
    expressions: true

security:
    encoders:
        Tfg\ProyectoBundle\Entity\Usuario: plaintext

    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]

    providers:

        main:

            entity: {class:Tfg\ProyectoBundle\Entity\Usuario, property: mail}



    firewalls:
        dev:
            pattern:  ^/(_(profiler|wdt)|css|images|js)/
            security: false



        secured_area:
            pattern:    ^/
            anonymous: ~
            form_login:
                check_path: /login_check
                login_path: /login
            logout:
                path:   /logout
                target: /

            #http_basic:
            #    realm: "Secured Demo Area"

    access_control:
        #- { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY,
requires_channel: https }
        #- { path: ^/_internal/secure, roles: IS_AUTHENTICATED_ANONYMOUSLY, ip
: 127.0.0.1 }
       - { path: ^/mostrar, roles: ROLE_USER }
```

Datos para login

configuración de seguridad

Como se aprecia en la imagen anterior, en el área de seguridad tenemos que configurar el tipo de login que deseamos, en este caso form_login, así como las rutas check_path y login_path. La primera de ellas es la que se encarga de llevar a cabo realmente el proceso de login, es decir, buscar entre los usuarios cargados en el apartado providers alguno que tenga las credenciales correctas. La segunda ruta es el formulario de login.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Resources\views\Default\login.html.twig

```
1  {# src/Tfg/ProyectoBundle/Resources/views/Default/login.html.twig #}
2
3  {% extends '::frontend.html.twig' %}
4
5  {% block title %} Autentificación {% endblock %}
6
7  {% block article %}
8
9      <div id="imagenesPago">
10
11         <figure>
12             <img src="{{ asset('bundles/tfgproyecto/images/monitor.jpg') }}">
13         </figure>
14
15     </div>
16
17     <div id="login">
18
19         <h2> Formulario de acceso al campus </h2>
20
21         {% if error %}
22
23             <h2> {{ error.message }}</h2>
24
25         {% endif %}
26
27         <form action="{{ path('login_check') }}" method="post"/>
28
29         <label for="_username">Usuario: (es el email) </label>
30         <input type="text" id="_username" name="_username" value=""/>
31
32         <label for="_password"> Password </label>
33         <input type="text" id="_password" name="_password"/>
34
35         <label for="login"></label>
36
37         <input type="submit" name="login">
38
39         </form>
40
41     </div>
42
43
44
45 {% endblock %}
46
47 {% block aside %}
48
49     <h3> Artículos técnicos de interés </h3>
50
51 {% endblock %}
```

Nombres
de los campos
_username
_password

Código formulario login

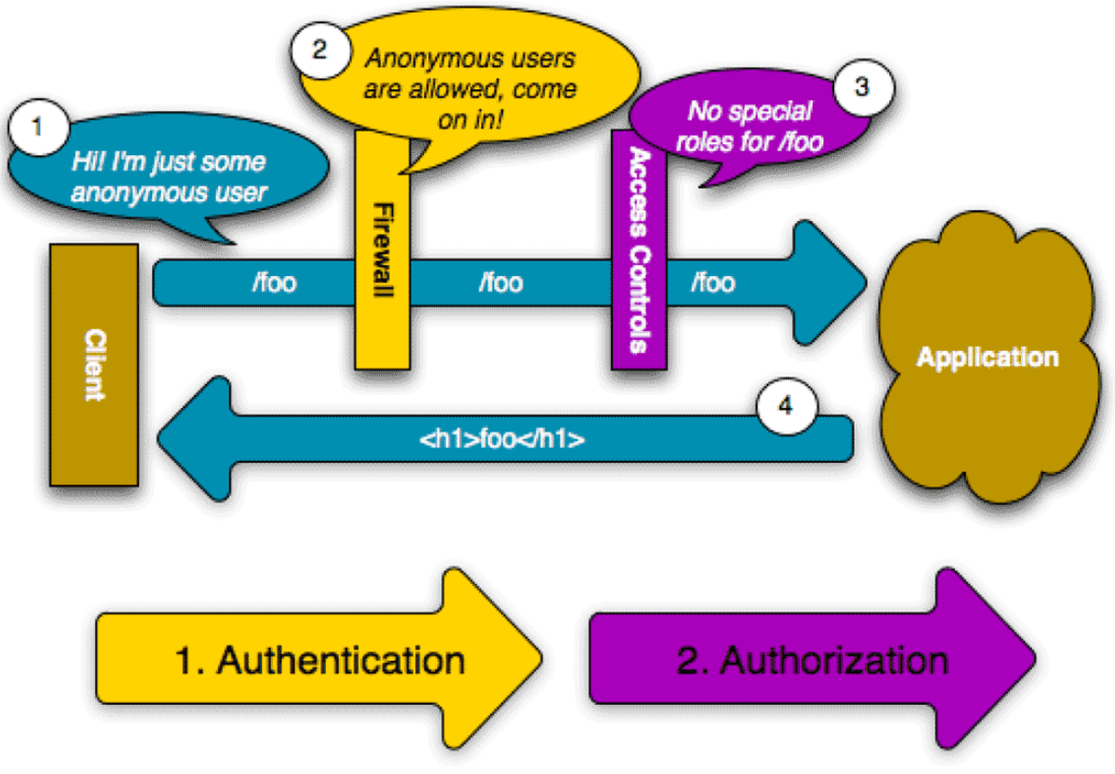
- Security

  Este API se encarga de controlar el acceso de los usuarios. En el archivo de seguridad anteriormente mostrado se configura también las distintas rutas protegidas, así como el rol que debe tener el usuario que accede a ellas. Podemos ver que existen diferentes tipos de usuarios. Nosotros como desarrolladores podemos configurar los que deseemos, pero de entrada podemos ver un par de cosas interesantes.

  En el patrón de seguridad podemos ver: ^\

  Esto indicaría que todas las páginas de la aplicación están protegidas. Sin embargo, para no pedir aún las credenciales al usuario (aún no es necesario) también vemos que se ha configurado un usuario anónimo. anonymus

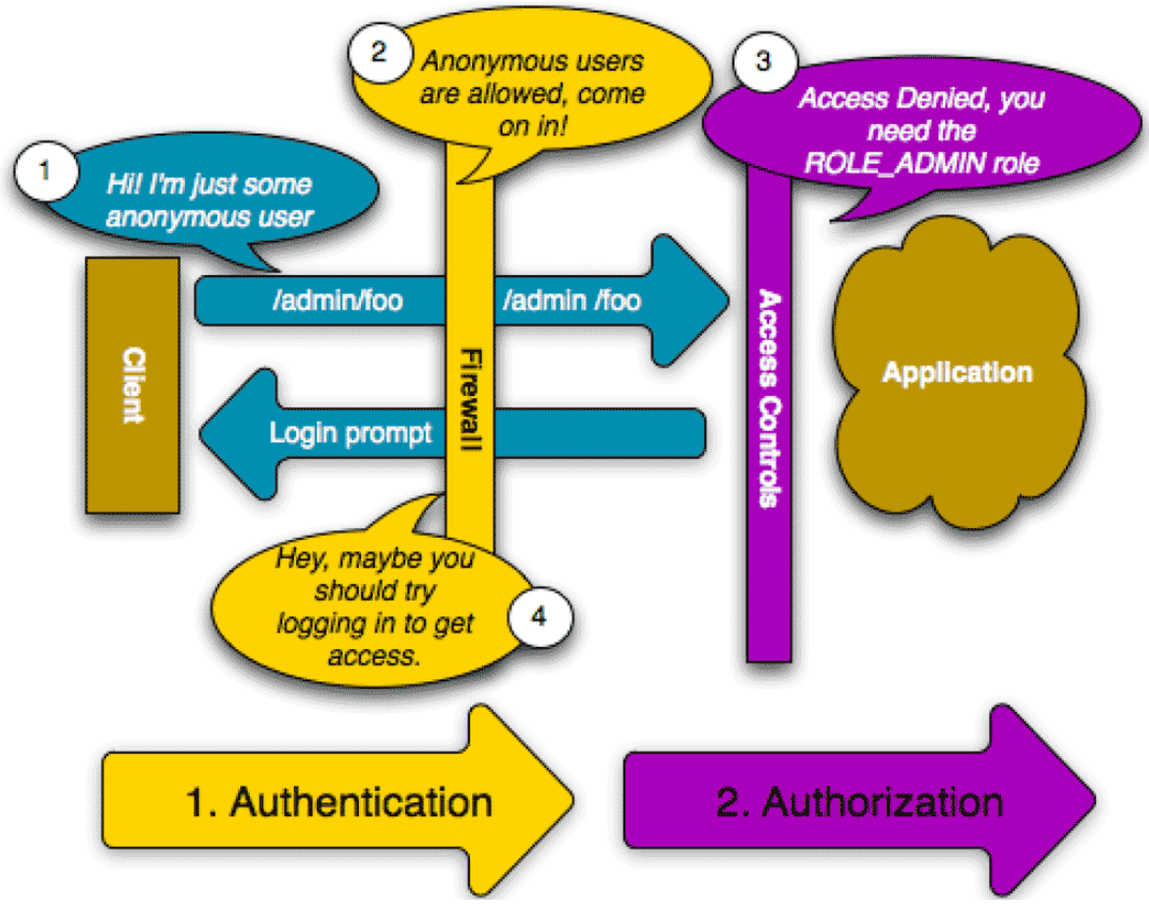
  El proceso de seguridad de acceso se lleva a cabo en dos pasos diferentes y es el siguiente.



Un usuario anónimo desea acceder a la página con ruta /foo. En primer lugar, el firewall lo deja pasar ya que en el archivo de configuración de seguridad no hemos definido ninguna restricción para esa ruta. Una vez que hemos pasado el firewall, el control de acceso comprueba si tenemos el rol adecuado para acceder. En este caso, tampoco existe un rol definido para esta ruta. Por ello la aplicación muestra la página solicitada sin ningún problema.

En la imagen siguiente se muestra otro caso. Un usuario anónimo intenta acceder a una ruta que está protegida y que necesita que el rol que intenta acceder sea ROLE_ADMIN

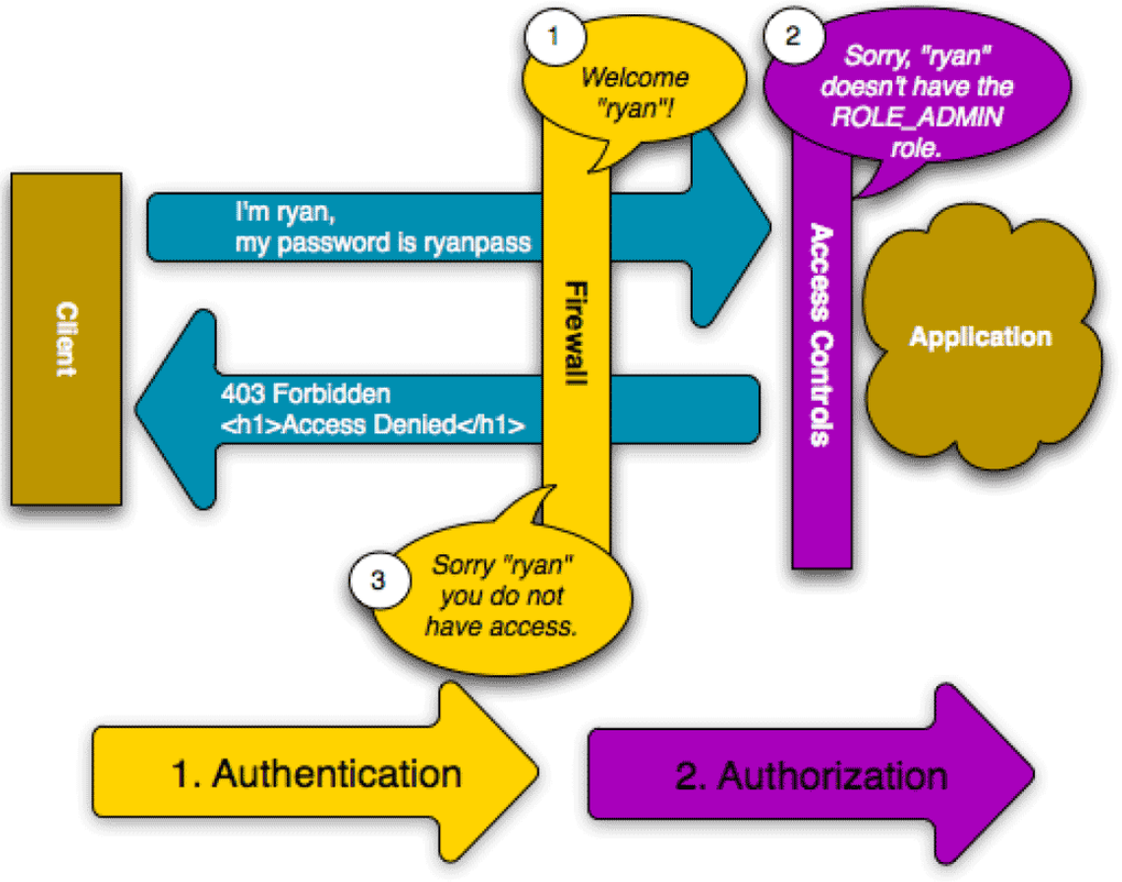
Caso 1 de autenticación



Caso 2 de autenticación

Ahora, el firewall le deja paso en primera instancia, pero es en el control de acceso el que indica que para acceder a la ruta indicada necesita acreditarse. Así pues es enviado al formulario de login.

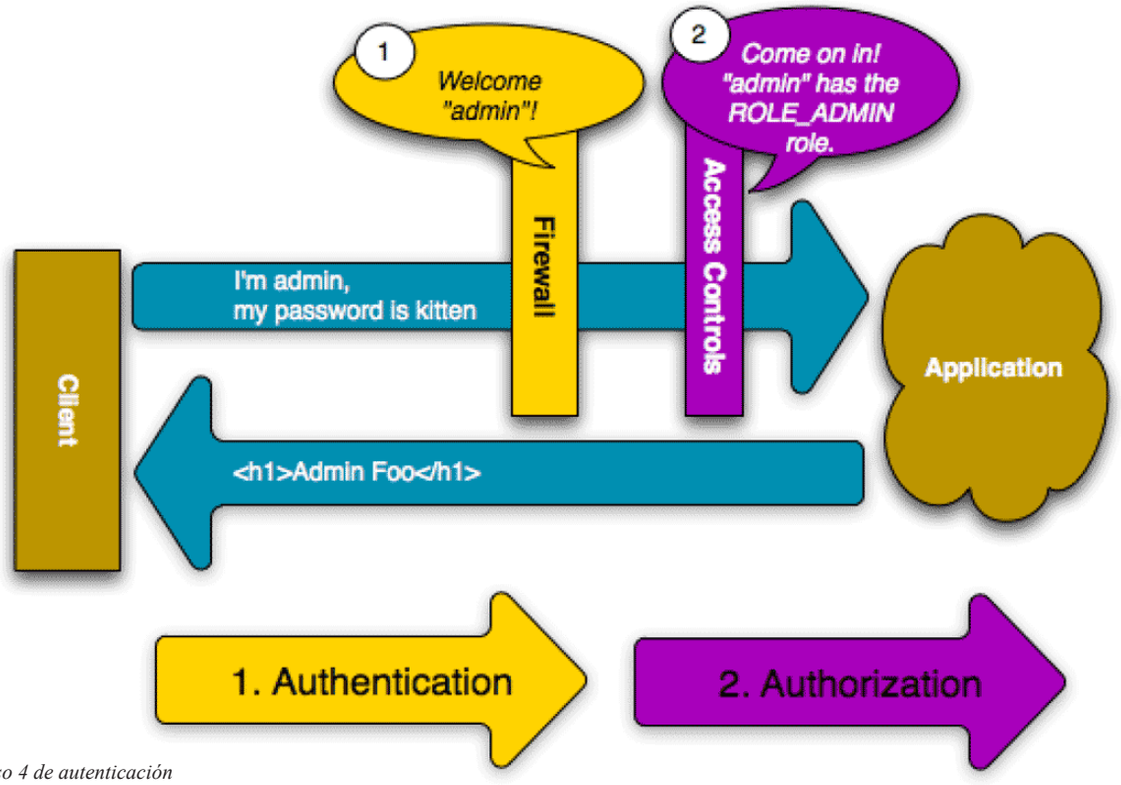
Por último en otras dos imágenes se muestra el proceso con otros dos casos. Un usuario registrado desea acceder a una ruta de administradores y un administrador que desea acceder a una ruta también de administrador.



Caso 3 de autenticación

En este caso, un usuario desea acceder a una ruta que está protegida mediante ROLE_ ADMIN es decir, sólo los usuarios que tengan dicho rol podrán acceder, aunque se hayan autentificado correctamente.

Este es el caso. Un usuario se autentifica correctamente, es decir su usuario y contraseña son válidos. El firewall se deja pasar, pero es el control de acceso el que al revisar su rol detecta que no es ROLE_ADMIN. En este caso el usuario ya no es enviado a login (ya se autenticó), sino que directamente se le niega el acceso.



Caso 4 de autenticación

En este último ejemplo, un usuario administrador, con rol (lógicamente) de administrado intenta acceder a una ruta protegida con ROLE_ADMIN. En primer lugar el firewall no actúa ya que se ha autenticado correctamente. Posteriormente el control de acceso revisa su rol. Como si es ROLE_ADMIN la aplicación devuelve la página solicitada sin mayor problema.

- Form

  Gracias a esta API la generación y validación de formularios se realiza de forma cómoda para el programador, evitando la programación manual de la validación de los campos, así como la persistencia en la base de datos de la información introducida por el usuario en el mismo.

  El procedimiento es el siguiente:

  1. Crear un archivo tipo Type con los campos que se mostrarán.
  2. Crear un controlador que se encargue de la validación y persistencia.
  3. Crear una plantilla que renderice la vista del formulario.

  En las siguientes imágenes se muestra el formulario de registro de la aplicación, el controlador encargado de su validación-persistencia, así como la plantilla que lo renderiza.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Form\UsuarioType.php

```php
<?php
/**
 * Created by JetBrains PhpStorm.
 * User: Pc
 * Date: 12/06/13
 * Time: 12:51
 * To change this template use File | Settings | File Templates.
 */

namespace Tfg\ProyectoBundle\Form;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilder;
use Symfony\Component\Form\FormBuilderInterface;


class UsuarioType extends AbstractType{

   public function buildForm(FormBuilderInterface $builder, array $opciones){

        $builder->add('apellidos');
        $builder->add('nombre');
        $builder->add('mail');
        $builder->add('password');

    }


    public function getName(){

        return 'usuario';
    }

}
```

Código archive UsuarioType

Como se puede observar, los archivos para formularios extienden la clase AbstracType. La Interface de esta implementa dos métodos builderForm() y getName(). El primero es el que estructura el formulario mediante add(). Cada uno de ellos es un campo del mismo.

El segundo método retorna el nombre para el formulario.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Controller\DefaultController.php

```
116             $form->bind($request);
117             if($form->isValid()){
118
119                 $em = $this->getDoctrine()->getManager();
120                 $em->persist($usuario);
121                 $em->flush();
122
123                 // return $this->redirect($this->generateUrl('pago'));
124                 return $this->render('TfgProyectoBundle:Default:pago.html.twig',array('usuario'=>$usuario));
125             }
126
127         }
128
129
130
131       return $this->render('TfgProyectoBundle:Default:creaContrato.html.twig',
132           array('tipoAcceso' => $contratoSeleccionado, 'formulario'=>$form->createView(), 'cliente'=>$usuario));
133
134
135     }
136
137
```

Parte del controlador que envía el formulario si es válido

Método que llama a la vista para generar el formulario

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Resources\views\Default\creaContrato.html.twig

```
35      <div id="formuRegistro">
36
37          <h3> Formulario de Registro </h3>
38
39          <form action="{{ path('contrato',{'id':tipoAcceso.id}) }} " method="po
    st">
40
41              {{ form_widget(formulario) }}
42
43              <input type="submit" />
44
45          </form>
46      </div>
```

Aquí genera el formulario

7.4. Test

La carga de datos sobre cursos, que en su momento será la parte de administrador a desarrollar próximamente se realiza mediante una clase que crea los objetos correspondientes, las asociaciones entre ellos y la persistencia en la base de datos.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Controller\PruebaCursoController.php

```php
<?php
/**
 * Created by JetBrains PhpStorm.
 * User: Pc
 * Date: 14/05/13
 * Time: 18:29
 * To change this template use File | Settings | File Templates.
 */

namespace Tfg\ProyectoBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

use Tfg\ProyectoBundle\Entity\Categoria;
use Tfg\ProyectoBundle\Entity\Tema;
use Tfg\ProyectoBundle\Entity\Curso;
use Tfg\ProyectoBundle\Entity\Usuario;
use Tfg\ProyectoBundle\Entity\TipoContrato;

class PruebaCursoController extends Controller
{
    public function crearAction(){

        $cat = new Categoria();
        $cat->setCategoria("Microsoft");



        $tema = new Tema();
        $tema->setDuracion("8 minutos");
        $tema->setTema("Fórmulas y monedas");
        $tema->setUrl("imagen1.jpg");

        $tema2= new Tema();
        $tema2->setTema("Enlazando celdas");
        $tema2->setDuracion("6 minutos");
        $tema2->setUrl("imagen2.jpg");


        $curso = new Curso();
        $curso->setTituloCurso("Exel");
        $curso->setDescripcion("Aprenda a manejar Exel a nivel profesional");
        $curso->addCategoria($cat);
        $curso->addTema($tema);
        $curso->addTema($tema2);


        $em = $this->getDoctrine()-> getManager();
        $em->persist($curso);
        $em->flush();

        return $this->render('TfgProyectoBundle:Default:cursos_prueba.html.twig
', array('curso'=>$curso));

    }


}
```

Aquí genera el formulario

Código para pruebas generar los cursos

En cuanto a contratos y usuarios se han realizado diferentes pruebas con todo tipo de combinaciones. Los resultados obtenidos son positivos. Para testear la caducidad de los contratos para realizar las pruebas, se ha modificado el método que calcula la fecha de finalización de acceso, atrasando la misma lo suficiente como para asegurar que los contratos están caducados.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Controller\DefaultController.php

```
137     public function verCursosAction(){
138
139
140
141         $user = $this->get('security.context')->getToken()->getUser();
142
143
144         $contratos = $user -> getContratos();
145         $indice = count($contratos);
146         $contrato = $contratos[$indice-1];
147
148         $hoy = new\DateTime();
149         $final = $contrato->getFechaFinal();
150
151         // La siguiente instrucción es para comprobar si funciona con fechas
    caducadas
152         //$final = date_modify($final, '- 10 months');
153
154         if($final<$hoy){
155
156             return $this->render('TfgProyectoBundle:Default:caducado.html.twig
    ',
157                 array('usuario'=>$user, 'contrato'=>$contrato));
158
159         }
160
161         $em = $this->getDoctrine()->getEntityManager();
162         $cursos = $em ->getRepository('TfgProyectoBundle:Curso')->findAll();
163
164         return $this->render('TfgProyectoBundle:Default:verCursos.html.twig',
165             array('cursos'=>$cursos, 'usuario'=>$user, 'contrato'=>$contrato))
166     }
167
168     public function abreAction($titulo){
169         $dire="bundles/tfgproyecto/videos/" . $titulo;
170
171         return $this->render('TfgProyectoBundle:Default:video.html.twig', arra
    y('titulo'=>$dire));
172     }
```

Código de pruebas para fechas caducadas

7.5. Bugs

- Errores en al implementación de las plantillas Twig. Estos ocurren al intentar colocar elementos de escritura dinámica fuera de bloques creados en las plantillas super de las que heredan aquellas que provocan el error.

  La solución es disponer de varias plantillas diferentes de tercer nivel (tercera herencia) que puedan aportar diferentes estructuras a las páginas, pero que eviten colocar contenido en secciones que no se encuentren dentro de un bloque editable.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Resources\views\Default\verCursos.html.twig

```
 1 {% extends '::usuarios.html.twig' %}
 2
 3 {% block title %} Pago tarjeta {% endblock %}
 4
 5 {% block article %}
 6     <p> Hola {{ usuario.nombre }}. Tu contrato finaliza el: {{ contrato.fechaFi
   nal |date('d/m/Y') }}</p>
 7     <h1> Listado de cursos disponibles </h1>
 8
 9
10     <div id="imagenesPago">
11
12         <figure>
13             <img src="{{ asset('bundles/tfgproyecto/images/clips_final.gif') }}
14         </figure>
15
16     </div>
17
18     <div id='cursos'>
19
20         {% for curso in cursos %}
21
22             <h2> Curso:  {{ curso.tituloCurso }}</h2>
23
24             {% for tem in curso.temas %}
25
26                 <h3> Tema: <a href="{{ path('abre',{'titulo':tem.url}) }}" targ
   et="_blank"> {{ tem.tema }}  </a> </h3>
27
28             {% endfor %}
29
30         {% endfor %}
31
32     </div>
33
34
35
36 {% endblock %}
37
38 {% block aside %}
39
40     <a href="{{ path('logout') }}"> Cerrar Sesión </a>
41
42 {% endblock %}
43
44
45
```

Los apartados block son heredados de otras plantillas. Error al intentar mostrar algo fuera de ello o no ajustarse a la estructura del block

Ejemplo de código de una plantilla Twig

- Errores al insertar datos en la base de datos mediante el ORM Doctrine. Los errores se dieron en una asociación ManyToMany en la que el sistema no encontraba la parte inversa de la misma.

  La solución es utilizar el atributo cascade en la definición de las asociaciones en las entidades.

File - C:\wamp\www\Proyecto\src\Tfg\ProyectoBundle\Entity\Usuario.php

```
42      /**
43       * @ORM\Column(type="string")
44       */
45      protected $password;
46
47
48      /**
49       * @ORM\Column(type="string")
50       */
51      protected $salt="";
52
53
54      /**
55       * @ORM\Column(type="string", length=1)
56       */
57      protected $tipo;
58
59
60      /**
61       * @ORM\OneToMany(targetEntity="TipoContrato", mappedBy="usuario",cascade={"persist", "remove"})
62       */
63      protected $contratos;
64
65      /**
66       * Constructor
67       */
68      public function __construct()
69      {
70          $this->contratos = new \Doctrine\Common\Collections\ArrayCollection();
71      }
72
73      /**
74       * Get id
75       *
76       * @return integer
77       */
78      public function getId()
79      {
80          return $this->id;
81      }
82
```

Parte del código de la entidad Usuario.

- Problemas en la puesta en producción de la aplicación.

  Según la documentación oficial de Symphony, el despliegue de las aplicaciónes no se debe realizar desde FTP sino con herramientas específicas que permitan asegurar una correcta puesta en producción. Estas herramientas son: Git y Capifony (necesita RubyGems).

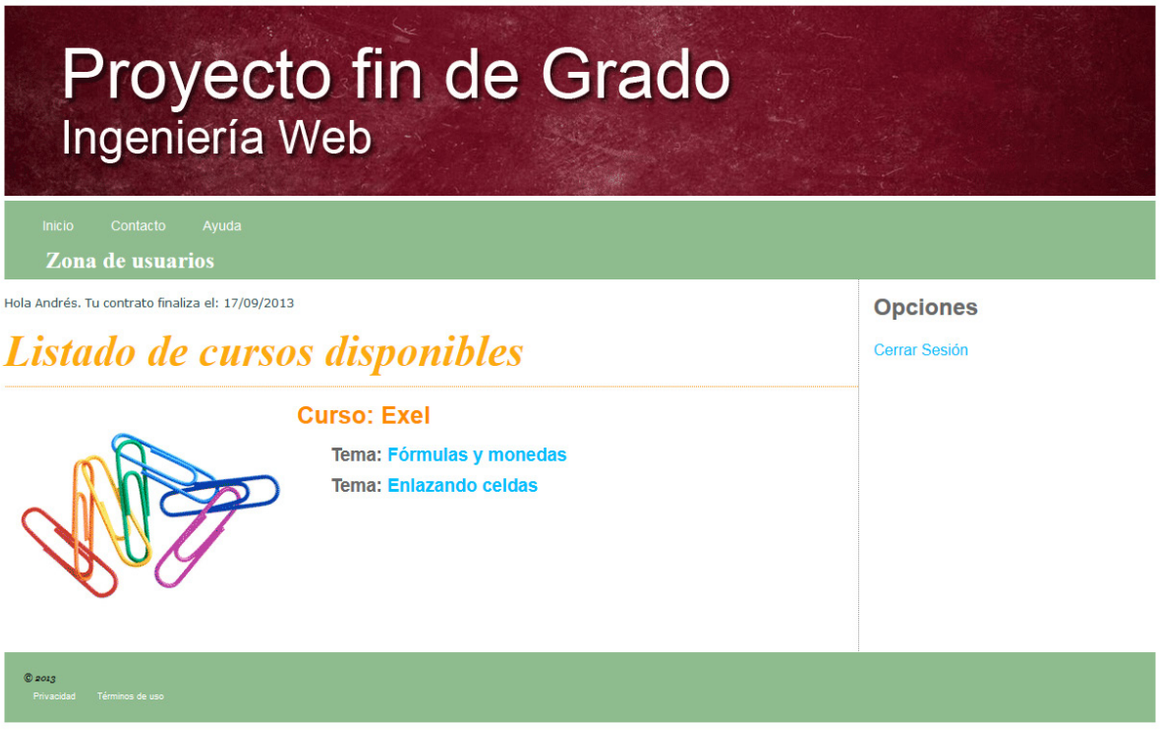
Muestro a continuación las pantallas más importantes de la aplicación

Detalle de la portada. Permite seleccionar contrato

Pantalla datos contrato y Registro

Pantalla datos contrato y usuario.

Pantalla de login



Pantalla de cursos disponibles

Pantalla contrato caducado

8. Proyecciones de futuro

- Implementar sistema de aviso a usuarios para que renueven su inscripción cuando esta esté próxima a expirar.
- Implementar sistemas de pago.
- Implementar adquirir y descargar tanto cursos completos como temas individuales.
- Implementar sistema de puntuación de los cursos por parte del usuario.
- Implementar la aplicación web de gestión del sitio para el administrador. Esto le permitirá añadir cursos, ofertas, espacios de tiempo para contratar etc.
- App para el seguimiento de los cursos desde dispositivos móviles.

9. Conclusiones

Es un salto cualitativo importante afrontar un proyecto de desarrollo completo partiendo de unos orígenes que hacen imprescindible adaptar los conocimientos en prácticamente todos los apartados tecnológicos a utilizar. En este caso en concreto, adaptado desde el principio a utilizar lenguaje PHP con HTML embebido, sin orientación a objetos, ha provocado la revisión completa de conocimientos entre otros:

- Pasar de XHTML - CSS a HTML 5 - CSS3. Esto no ha llevado mayor problema. La adaptación ha resultado fácil. Esta adaptación ha sido imprescindible para poder implementar las vistas como plantillas Twig.

- Pasar de un desarrollo de programación estructural de PHP, a un uso de la programación utilizando las técnicas de Programación Orientada a Objetos. Esta adaptación tampoco ha resultado difícil ya que los conceptos estaban bien asimilados gracias a las asignaturas propias del área elegida de especialización: Ingeniería Web. Utilizar estas técnicas es, en realidad, el principal objetivo del proyecto.

- Pasar de utilizar la programación PHP embebida en archivos HTML a una separación según el modelo MVC: Modelo, Vista, Controlador. Este ha sido posiblemente la parte más difícil, pues ha supuesto conocer en buen grado algún framework capaz de soportar la programación de aplicaciones web.

  Si bien en un primer momento la decisión fue utilizar Java para la implementación del proyecto, dada su excelencia en cuanto a la POO, se descartó esta teniendo en cuenta que se trataba de la primera aproximación a un framework de forma práctica y real. La curva de aprendizaje de Hibernate, por ejemplo era demasiado prolongada en el tiempo. Además de que, quizás Java se adapte mejor para otro tipo de aplicaciones informáticas.

  Teniendo en cuenta además de la existencia de frameworks basados en PHP mucho mejor adaptados a desarrollo web, se tomó la decisión de cambiar la tecnología a uno de estos frameworks. Después de estudiar pro y contras de varios de ellos fue Symfony2 el elegido.

  Han sido varias la dificultades desde el punto de vista de la adaptación. En primer lugar, entender el propio funcionamiento del framework. Cómo se establecen las rutas, como actúa el controlador, cómo se generan la vistas, etc. No es sólo entender la filosofía de Symfony2, sino no provocar errores en la implementación.

  En este apartado han sido tres apartados los de mayor dificultad de aprendizaje.

  - El ORM Doctrine. Hasta el proyecto, siempre, el acceso a datos se ha implementado desde PHP con órdenes directas SQL a base de datos MySQL. Ha sido la primera vez en utilizar un ORM. Esto ha supuesto una mayor dedicación en la curva de adaptación que en otras facetas. Especialmente en la realización de las asociaciones entre entidades.

  - La creación y uso de formularios. Hasta el proyecto, los formularios siempre fueron implementados mediante archivos HTML con llamadas a otros archivos que se encargaban del tratamiento del mismo. Por primera vez los formularios se han generado utilizando archivos especiales de tipo “Type” vinculados a alguna

entidad a la que pasar los datos introducidos, validarlos y persistirlos en la base de datos. Aunque muy potente, este apartado también ha necesitado alguna dedicación extra de tiempo de adaptación.

- La seguridad. Hasta el momento, el acceso a páginas para protegerlas e iniciar sesión se había implementado mediante programación PHP con variables y arrays globales. Ha sido también la primera vez que la seguridad se realiza desde un archivo específico, dónde se indica las rutas protegidas que deben provocar la autenticación del usuario, así como los roles de acceso a determinadas partes de la aplicación.
- Y finalmente la fase de despliegue que puede ser muy problemática ya que existen muchas dependencias entre las partes del proyecto y enrutamientos que cambian al cambiar de servidor. Se recomienda usar herramientas específicas como Capifony.

Una vez llevado a cabo la adaptación de conocimiento, el proyecto ha permitido la aplicación práctica y funcional de implementar todos estos apartados mencionados, y hacerlo utilizando la POO, que era el objetivo principal.

Por todo ello, la percepción al finalizar el proyecto es de satisfacción, pues ahora tengo el conocimiento global de funcionamiento de un framework, lo que me permitirá adaptar rápidamente estos conocimientos a otros diferentes a Symfony2. Además, este es un potente framework muy dirigido al desarrollo web lo que facilita enormemente (una vez conocido) el desarrollo de aplicaciones.

Por supuesto que habría sido mejor llegar más lejos en la implementación real de la aplicación. Esto se hará sin duda una vez se dominan los principios tecnológicos de Symfony2. La aplicación seguirá avanzando hasta completar la implementación integra de la misma.

Anexo 1. Entregables

- El código de la aplicación.

  Se entrega toda la carpeta del proyecto con toda la estructura creada por Symfony2 para el desarrollo del mismo. Parte de esa estructura está destinada a resolver el tema de dependencias, especialmente de las librería utilizadas por el framework.

- La base de datos.

  Tanto el archivo de la base de datos física, como información sobre su acceso

- Presentación PowerPoint.
- Video presentación.

Anexo 2. Librerías

- Componentes
  - ClassLoader: Carga las clases del proyecto.
  - HttpFoundation: Capa orientada a objetos según especificaciones HTTP. Define objetos Request (cookies, attributes, files, etc) y Response (content, status, headers).
  - DependencyInjection: Estandariza la forma en la que se crean los objetos.
  - Console.
  - Config.

- Librería externas
  - Twig: Sistema de plantillas para PHP con sintaxis orientada a plantillas.
  - Doctrine: conjunto de librería que proporciona servicios de mapeado entre entidades / tablas, además de persistencia.

- Bundles
  - FrameworkBundle.
  - WebProfailerBundle.
  - TwinBundle.
  - SecurityBundle.

Anexo 3. Resumen

La implementación del proyecto ha hecho posible conocer el funcionamiento de un framework basado en el patrón MVC. Ha sido importante conocer su forma de trabajo, sus ventajas en la producción de aplicaciones web. Esto ha llevado consigo una actualización importante de algunos conocimientos adquiridos en varias asignaturas, especialmente aquellos con los que más prácticas he realizado como HTML, CSS, PHP.

Por otra parte también ha sido necesario alcanzar nuevos conocimientos, en este caso los referidos a Symfony2. Librerías como Doctrine, Twig, Form, Secrity que hacen posible que el desarrollador se pueda centrar en el modelo, dejando al framework tareas repetitivas como puede ser la validación de formularios. También la persistencia en bases de datos. Tratado todo ellos desde una programación orientada a objetos.

El resultado ha sido fructífero. Ahora queda profundizar en los conocimientos adquiridos, ampliando la aplicación realizada.

Anexo 4. Bibliografía

- P.J. Deitel, H. M. Deitel. "Java. Cómo programar". 7ª Edición. Editorial Pearson Educación. Año 2008.
- Laurent Debrauwer. "Patrones de diseño para C#". Ediciones ENI. Año 2012.
- Craig Larman. "UML y Patrones". 2ª Edición. Prentice Hall. Año 2003.
- Varios autores. "Programación orientada a objetos". Editorial UOC. Año 2007.
- Laurent Debrauwer. "UML 2". Segunda edición. Ediciones ENI. Año 2009.
- Manual de Symfony2. Release 2.0.2. [PDF].
- Doctrine 2 ORM Documentation. Release 2.1. [PDF].
- Programación Orientada a Objetos para PHP 5. [PDF].
- Symfony. Framework PHP orientado a objetos. [PDF].
- Manual de Twig. Release 1.2.0. [PDF].
- Desarrollo Ágil con Symfony 2.1. [PDF].
- API Symfony2. [WEB]. http://api.symfony.com/2.1/index.html
- Sitio Oficial de Symfony2. [WEB]. http://symfony.com/
- Instalando Symfony2 en Wamp. [WEB]. http://blog.bloque9.com/index.php/item/87-instalando-symfony-en-wamp-php-5-3-8
- El Componente YAML. [WEB]. http://gitnacho.github.io/symfony-docs-es/components/yaml.html
- Twig. [WEB]. http://gitnacho.github.io/Twig/
- Bases de datos y Doctrine. [WEB]. http://gitnacho.github.io/symfony-docs-es/book/doctrine.html
- Utilizando Doctrine como ORM en PHP. [WEB]. http://web.ontuts.com/tutoriales/utilizando-doctrine-como-orm-en-php/
- Mapeo de relaciones uno a muchos [WEB]. http://parasitovirtual.wordpress.com/2011/03/09/mapeo-de-relaciones-uno-a-muchos-one-to-many-en-doctrine-2/
- Creación de formularios en Symfony2. [WEB]. http://symfony.com/legacy/doc/forms/1_2/es/01-Form-Creation
- Creación de formularios. [WEB]. https://gist.github.com/bschussek/883293
- RegistrationFormType::buildForm() not compatible. [WEB]. http://stackoverflow.com/

questions/12359307/registrationformtypebuildform-not-compatible

- Creando un formulario sencillo. [WEB]. http://librosweb.es/symfony_2_1/capitulo_12/_creando_un_formulario_sencillo.html
- Creando un login simple con Symfony2. [WEB]. http://roberto.costumero.es/2011/08/15/creando-un-login-simple-con-symfony2/