UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Aplicación Web para la evaluación de prácticas de laboratorio en tiempo real

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Raúl Fernández Cabrera

Director: José Vicente Busquets Mataix

Septiembre de 2012

Resumen

El proyecto a desarrollar consiste en un sistema de evaluación de prácticas de laboratorio que permita a los alumnos realizar las prácticas creadas por los profesores y ser evaluados por el sistema en tiempo real sin intervención humana. Se desarrollará sobre el sistema operativo Windows XP y podrá ser usado a priori desde cualquier sistema operativo con navegador de internet. Para su implementación se utilizará exclusivamente tecnologías de uso libre para evitar el pago de licencias de software, aplicándose esta filosofía tanto en el framework sobre el que se desarrolla la aplicación como en los requisitos del mismo, así como en las aplicaciones usadas para elaborar el presente documento y los gráficos que contiene.

Palabras clave: symfony, evaluación, prácticas, PHP, WAMP, framework, aplicación web, docencia en red.

ÍNDICE DE CONTENIDOS

1. ÍNDICE DE FIGURAS

2. INTRODUCCIÓN

El objetivo de este proyecto es abordar la implementación de un sistema de evaluación de prácticas de laboratorio en tiempo real basándose en el framework Symfony.

Symfony es un potente marco de trabajo que permite acelerar la creación de aplicaciones web basadas en bases de datos, automatizando las tareas comunes en este tipo de aplicaciones, como son la creación de bases de datos, carga de datos iniciales de la aplicación, carga de datos de pruebas, creación de formularios e incluso validación de datos básica.

El uso de Symfony implica de forma inherente usar una serie de aplicaciones que el framework necesita para funcionar:

- Un servidor web. Para el presente proyecto se usa Apache, debido a su condición de gratuito y a haber sido utilizado con anterioridad.

- PHP. Un leguaje de programación del lado de servidor que se usará para desarrollar las funcionalidades requeridas, de uso gratuito.

- Un Sistema de Gestión de Bases de Datos relacionales. Este requisito es necesario si la aplicación va a hacer uso de una base de datos, algo que es muy habitual. La aplicación a desarrollar efectivamente hará uso de una base de datos y para ello se usará MySQL, debido a ser software gratuito y a estar incluido en el paquete WAMP.

- Doctrine. Symfony hace uso de la metodología ORM para interactuar con los datos de la base de datos como si fueran objetos, permitiendo reemplazar las sentencias SQL clásicas por instrucciones más intuitivas.

- Un entorno de desarrollo. Para escribir el código de la aplicación se puede usar un editor de texto cualquiera, en este caso se ha elegido Notepad++ por su enorme versatilidad a la hora de colorear el código en función del lenguaje de programación usado, identarlo, y por supuesto por ser de uso libre.

De la misma manera, para la elaboración de la memoria se ha utilizado íntegramente software de uso libre:

- Procesador de textos. Se ha elegido OpenOffice, completa suite ofimática.

- Generación de material gráfico. Para crear el modelo UML, casos de uso, esquemas y diagramas de flujo de datos se ha usado StarUML.

- Gimp para la edición gráfica del material generado en StarUML, elegir el tipo de archivo a guardar y reducir su tamaño.

3. ANÁLISIS

3.a ANÁLISIS DE REQUISITOS

El sistema a implementar debe permitir a los alumnos autentificados resolver prácticas desde un cliente web en un laboratorio de prácticas y obtener la nota correspondiente en tiempo real sin necesidad de esperar a que un profesor realice evaluación alguna, facilitándo de este modo el trabajo de los profesores y mejorando la eficacia de las prácticas. Los alumnos podrán además consultar las notas obtenidas en las prácticas que hayan finalizado. Un alumno no debe poder resolver una práctica más de una vez para evitar que averigue los resultados correctos de las preguntas por el método de prueba y error. El sistema registrará la fecha y hora en que el alumno realizó la práctica.

Los profesores que utilicen el sistema podrán crear nuevos alumnos en el sistema y modificar los datos de los existentes, no pudiendo sin embargo realizar estas acciones con otros profesores. Los profesores podrán obtener listados de las notas obtenidas por cada alumno, así como de las notas obtenidas en cada práctica.

El sistema permitirá a los profesores la creación y configuración de prácticas, permitiendo introducir el título y descripción de la práctica y configurarla con un número indeterminado de cuestiones. Cuando los alumnos resuelvan las prácticas el sistema calculará la nota de manera dinámica independientemente de este número. Los profesores podrán modificar las prácticas ya creadas añadiendo o eliminando cuestiones, además de los otros datos inherentes a la práctica. El sistema establecerá automáticamente al profesor que cree o modifique una práctica como responsable de la práctica, y guardará la fecha y hora de creación y modificación. A diferencia de los alumnos, los profesores deben ver en las prácticas las cuestiones de las que constan así como las respuestas correctas. Los alumnos solo verán las cuestiones cuando se dispongan a resolver la práctica.

Los profesores podrán crear nuevas cuestiones, así como modificar las existentes y agregarlas a las prácticas. El sistema contemplará cuestiones de tres tipos:

- Cuestiones de tipo test: Deben tener enunciado y cuatro posibles respuestas. Solo una de ellas será válida.

- Cuestiones de respuesta múltiple: Dispondrán de enunciado y cuatro posibles respuestas que pueden ser válidas o no. Se considerá que la cuestión se ha respondido correctamente si el alumno acierta las cuatro respuestas.

- Cuestiones de respuesta corta: Dispondrán de enunciado y una sola respuesta de tipo numérico. El sistema permitirá a los profesores configurar estás cuestiones con un margen de error dentro del cual se considerará la cuestión como respondida correctamente.

Debe existir en el sistema un usuario administrador, que podrá crear profesores y modificar los datos de los existentes, así como realizar el resto de funciones propias de un profesor. Así pues, la aplicación contemplará tres tipos de usuario y realizará su autenticación de manera segura, nunca

almacenando las contraseñas en texto plano.

Por último, el sistema notificará apropiadamente a los usuarios de los inicios y cierres de sesión, así como de los eventos que requieran notificación para cumplir los estándares de usabilidad.

3.b MODELADO DE LA APLICACIÓN

Conforme a los requisitos recogidos en el apartado anterior se desarrollará la aplicación tomando como base el siguiente modelo UML:

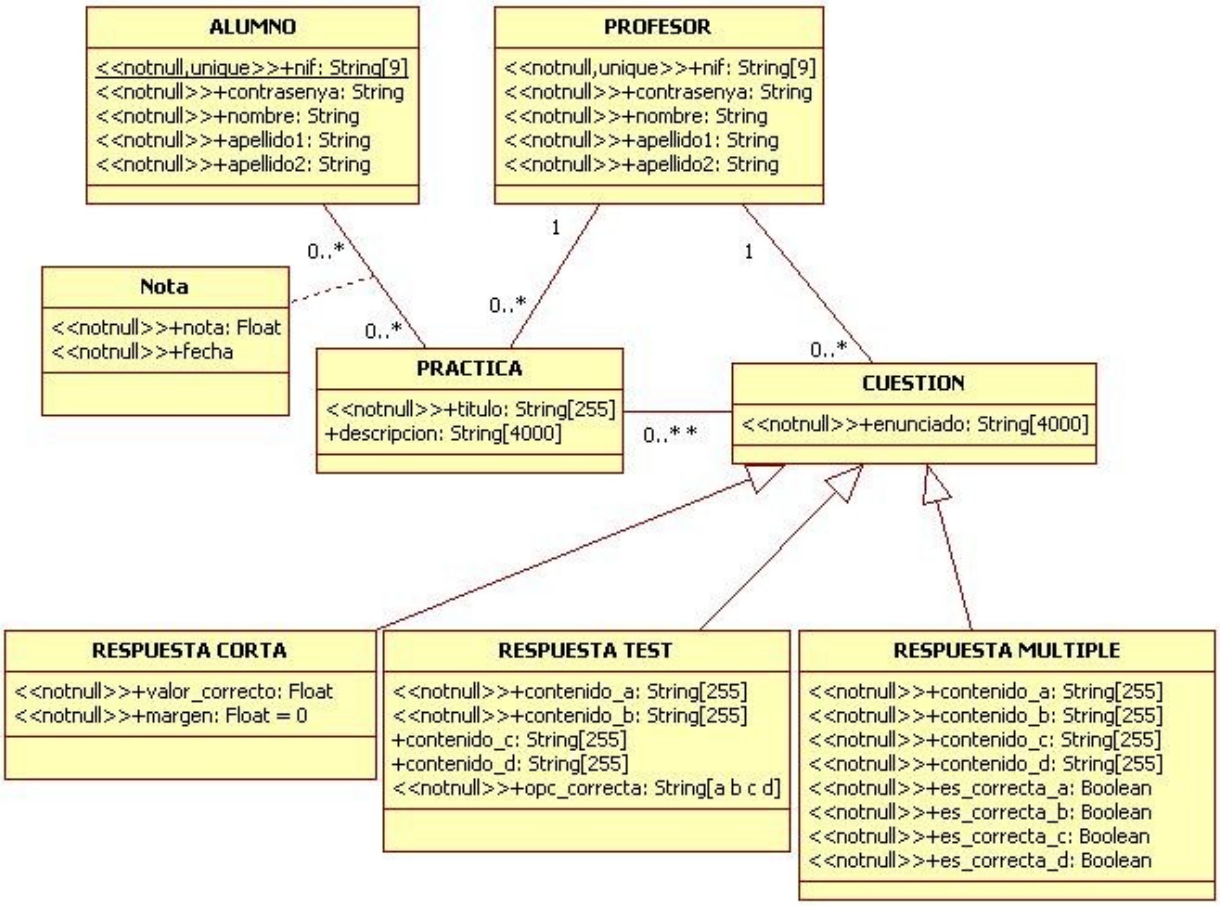

Ilustración 1: Modelo UML de la aplicación

3.c CASOS DE USO

Para formar una idea global de la aplicación, se extraen de los requisitos los casos de uso siguientes:

3.c.i Caso de uso del usuario al iniciar sesión

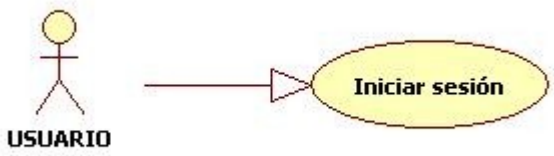

Ilustración 2: Caso de uso 1: Iniciar sesión

3.c.ii Casos de uso del usuario Alumno

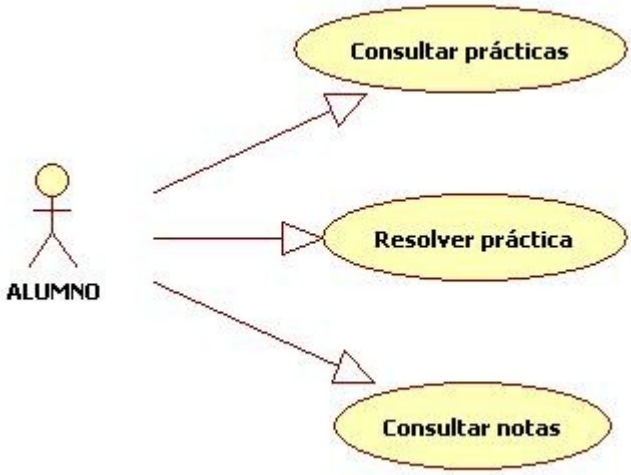

Ilustración 3: Casos de uso 2, 3 y 4: Usuario Alumno

3.c.iii Casos de uso del usuario Profesor

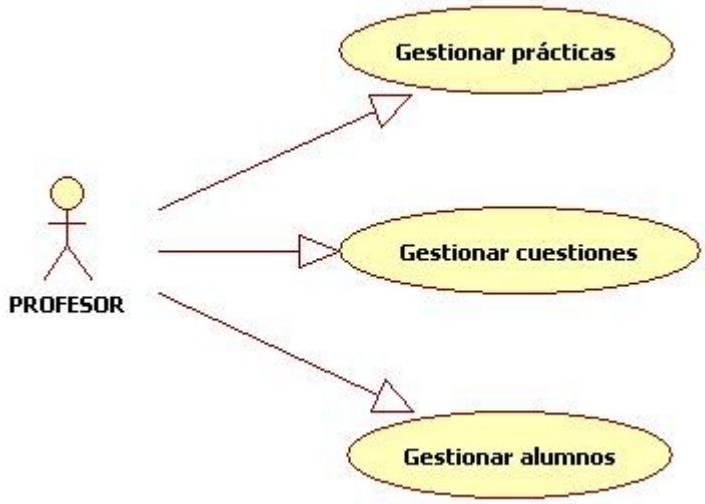

Ilustración 4: Casos de uso 5, 6 y 7: Usuario Profesor

3.c.iv Caso de uso del usuario Administrador

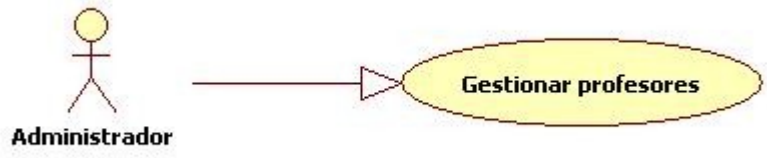

Ilustración 5: Caso de uso 8: Usuario Administrador

3.d DIAGRAMAS DE FLUJO DE DATOS

Profundizando un poco más en los requisitos, a partir de los casos de uso derivan diagramas de flujo de datos útiles para conocer en más detalle el funcionamiento de la aplicación.

3.d.i Diagrama de flujo de datos del usuario al iniciar sesión

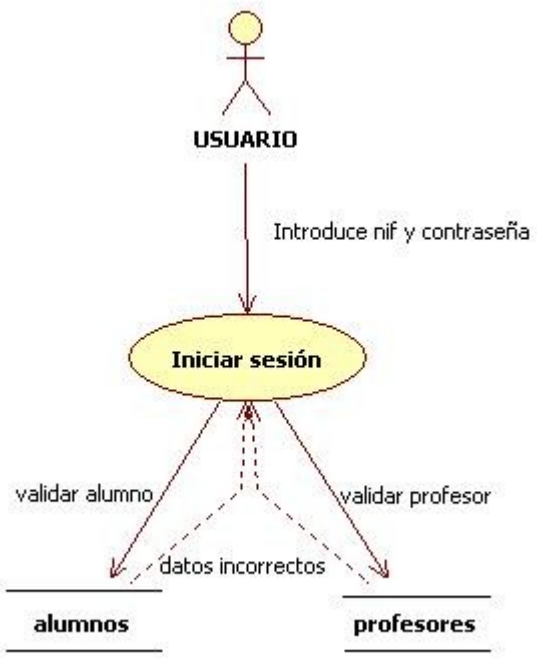

Ilustración 6: Diagrama de flujo de datos 1
Usuario inicia sesión

3.d.ii Diagramas de flujo de datos del usuario Alumno

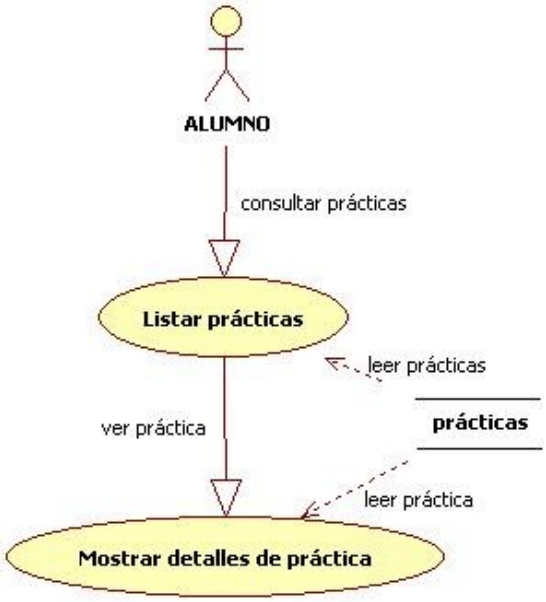

Ilustración 7: DFD2: Alumno consulta práctica

Ilustración 9: Diagrama de flujo de datos 4: Alumno consulta notas

Ilustración 8: DFD 3: Alumno resuelve práctica

3.d.iii Diagramas de flujo de datos del usuario Profesor

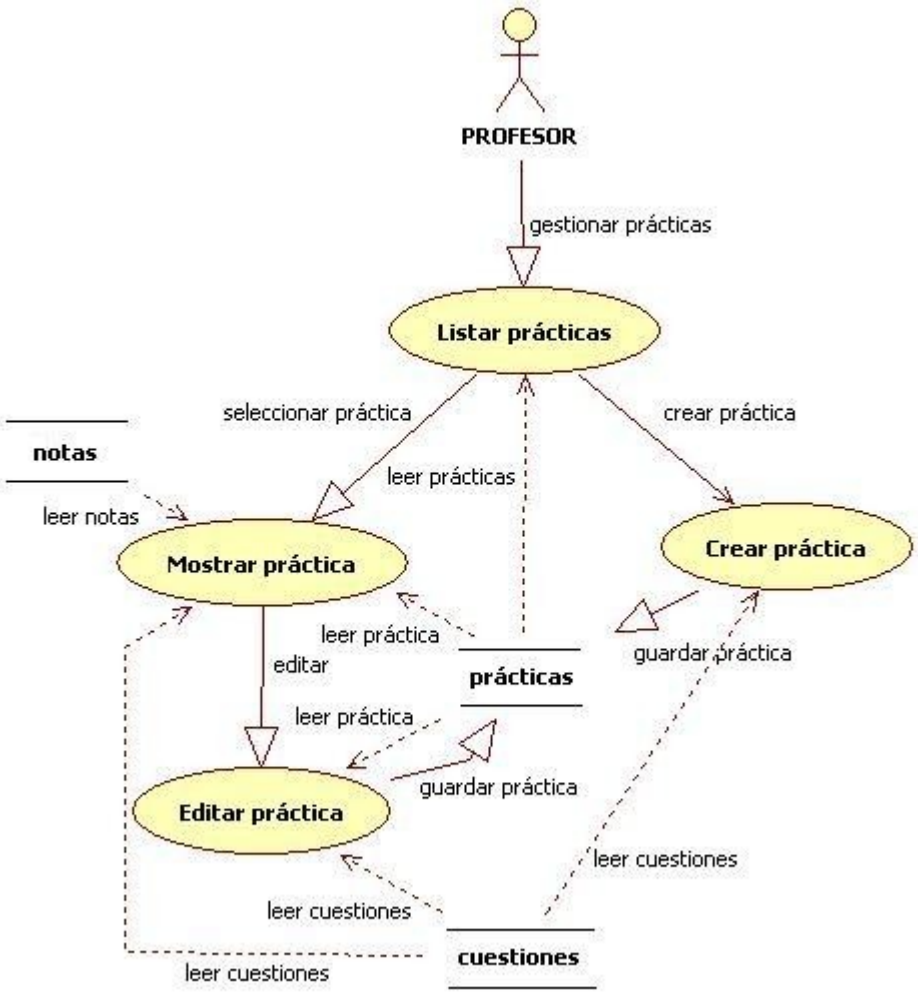

Ilustración 10: Diagrama de flujo de datos 5: Gestión de prácticas

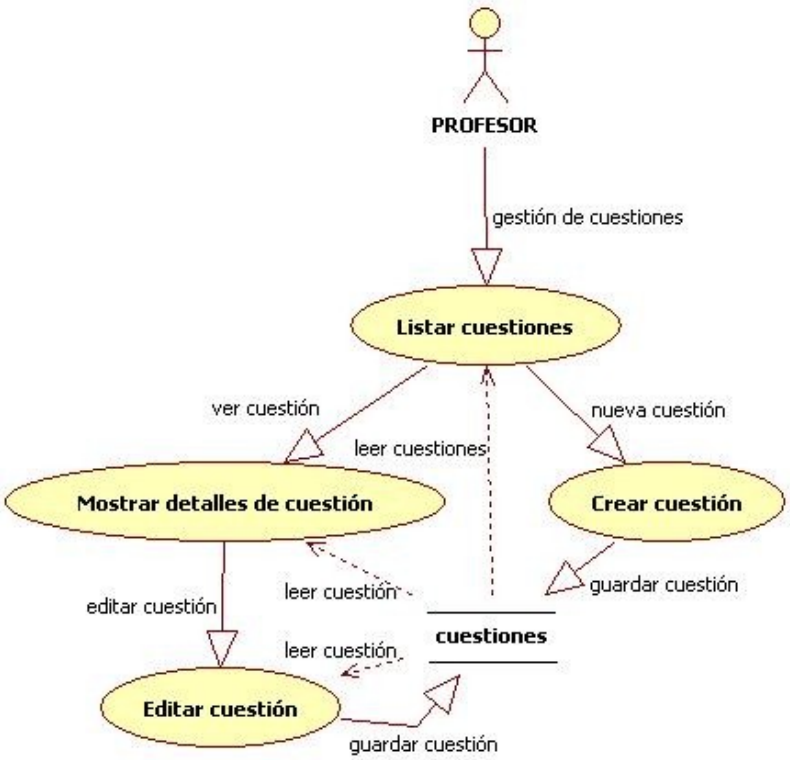

Ilustración 11: Diagrama de flujo de datos 6: Gestión de cuestiones

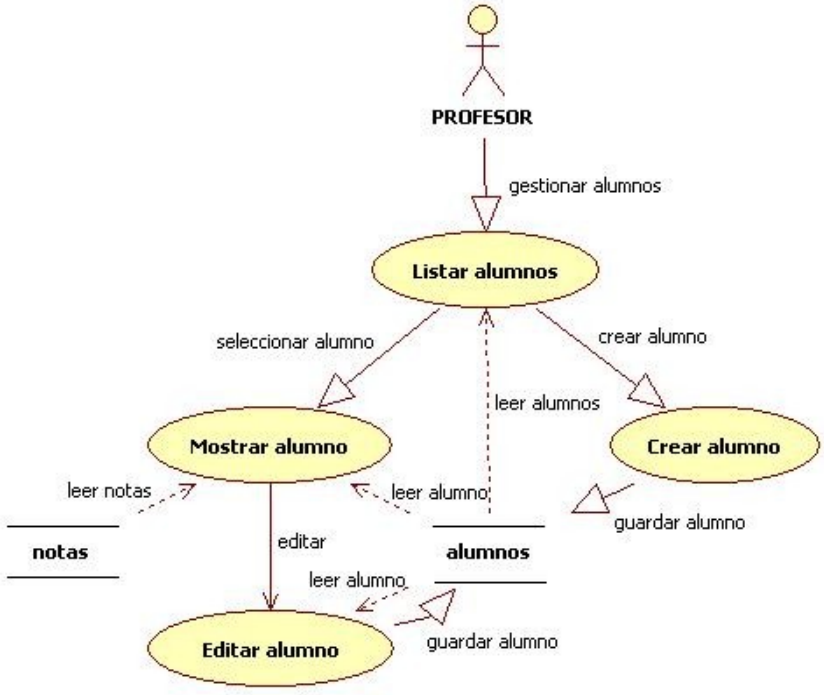

Ilustración 12: Diagrama de flujo de datos 7: Gestión de alumnos

3.d.iv Diagrama de flujo de datos del usuario Administrador

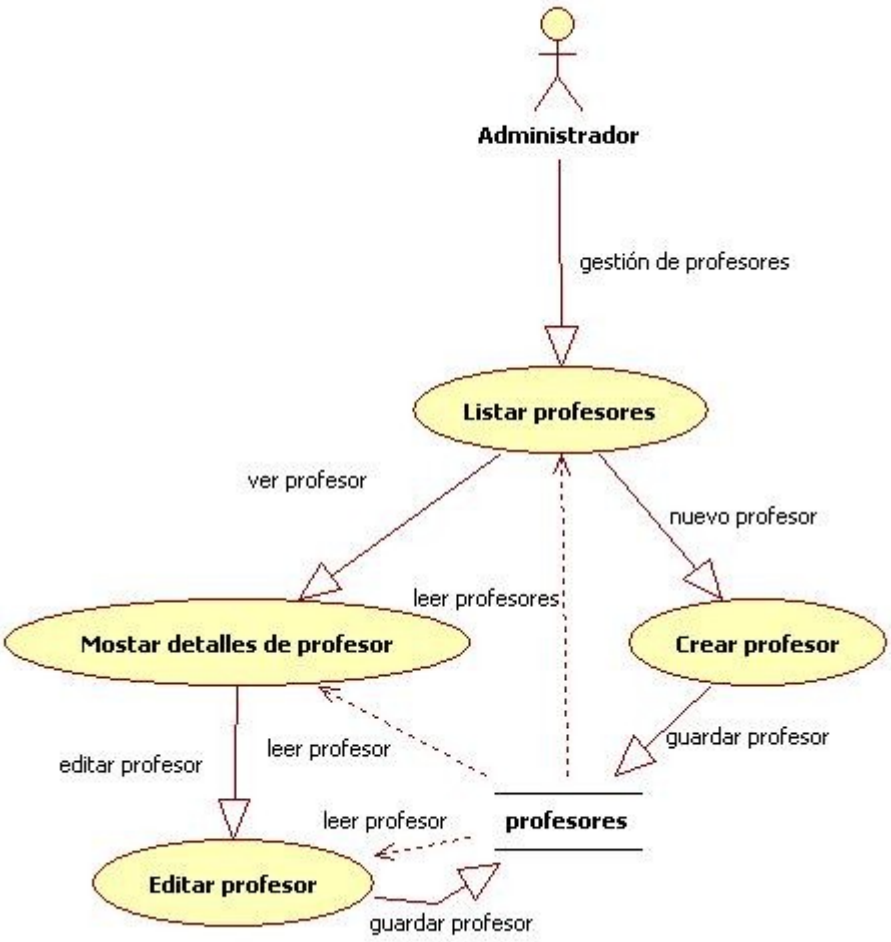

Ilustración 13: Diagrama de flujo de datos 8: Gestión de profesores

4. PUESTA EN MARCHA DE SYMFONY

4.a INSTALACIÓN DE SYMFONY Y SUS PREREQUISITOS

Para poder utilizarSymfony, previamente debe haberse instalado un servidor web y PHP. Si además se va a utilizar una base de datos será necesario un sistema de gestión de bases de datos.

El paquete de software Wamp (acrónimo de Windows-Apache-Mysql-PHP) incluye el servidor web Apache, el sistema de gestión de bases de datos relacionales Mysql y el lenguaje de programación web del lado de servidor PHP en un solo instalador. Son programas de código libre y por lo tanto no suponen coste alguno. Wamp está disponible de forma gratuita en la página web oficial del proyecto, y usándolo se acelera la adecuación del equipo de trabajo a los prerequisitos de Symfony. Para instalar Wamp se requieren privilegios de administrador local.

Una vez instalado Wamp, puede ejecutarse desde "*Inicio - Todos los programas – WampServer - start WampServer*".

Se puede comprobar que la instalación se ha efectuado de manera correcta accediendo a localhost desde un navegador web.

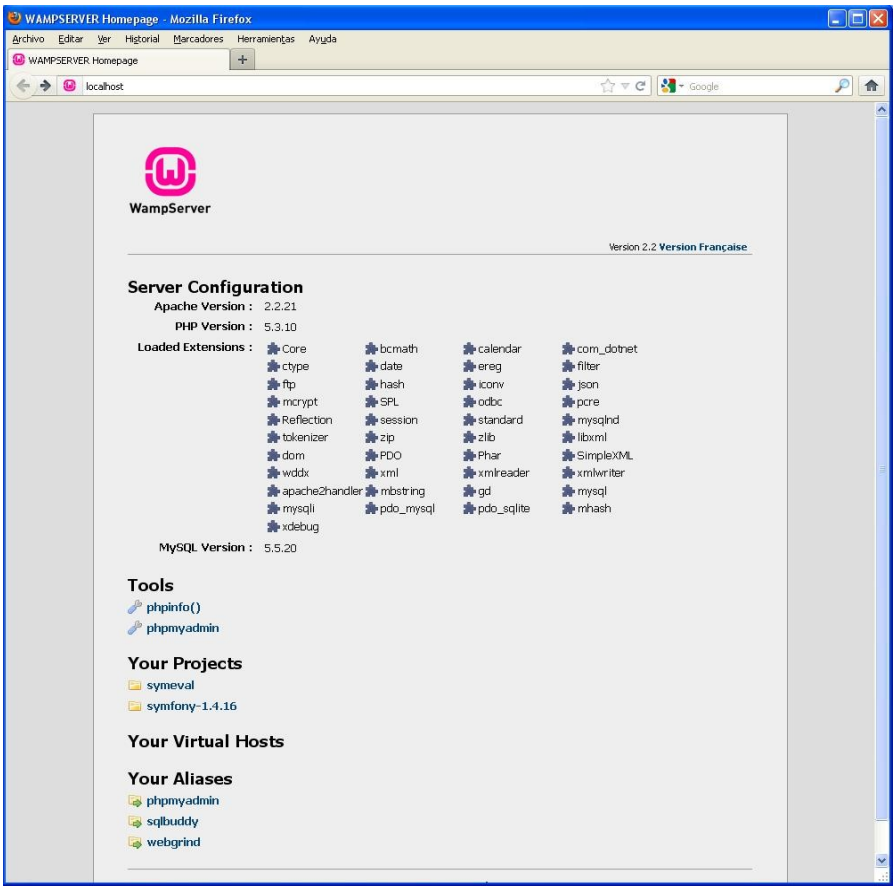

Ilustración 14: Página de inicio de WAMP

Se descomprimirá el archivo *symfony-1.4.16.zip* en C:\wamp\www\symfony-1.4.16\ obteniendo de este modo los componentes necesarios para usar el marco de trabajo. Antes de poder usarlo es necesario configurar el sistema operativo para que funcionen correctamente PHP y MySQL.

4.b CONFIGURACIÓN DE PREREQUISITOS DE SYMFONY

Para poder ejecutar los potentes comandos de Symfony se requiere PHP, y para ejecutar PHP es necesario indicar al sistema operativo dónde se encuentra su ejecutable. Lo mismo se aplica para el gestor de bases de datos. El sistema operativo dispone de una variable de entorno llamada *Path* que se debe modificar a tal efecto. Se puede acceder a esta variable desde "*Inicio - Mi Pc – (botón derecho) Propiedades - Pestaña Opciones Avanzadas – Botón Variables de entorno*". Para modificarla se requiere un usuario con privilegios de administrador, y para que los cambios sean efectivos es preciso reiniciar el equipo.

Se modificará la variable *Path* añadiendo las rutas de los dos ejecutables mencionados:
C:\wamp\bin\php\php5.3.8
C:\wamp\bin\mysql\mysql5.5.16\bin

Los números de versión pueden variar dependiendo del número de versión de Wamp usada. En este

caso se ha utilizado la versión **2.2a**

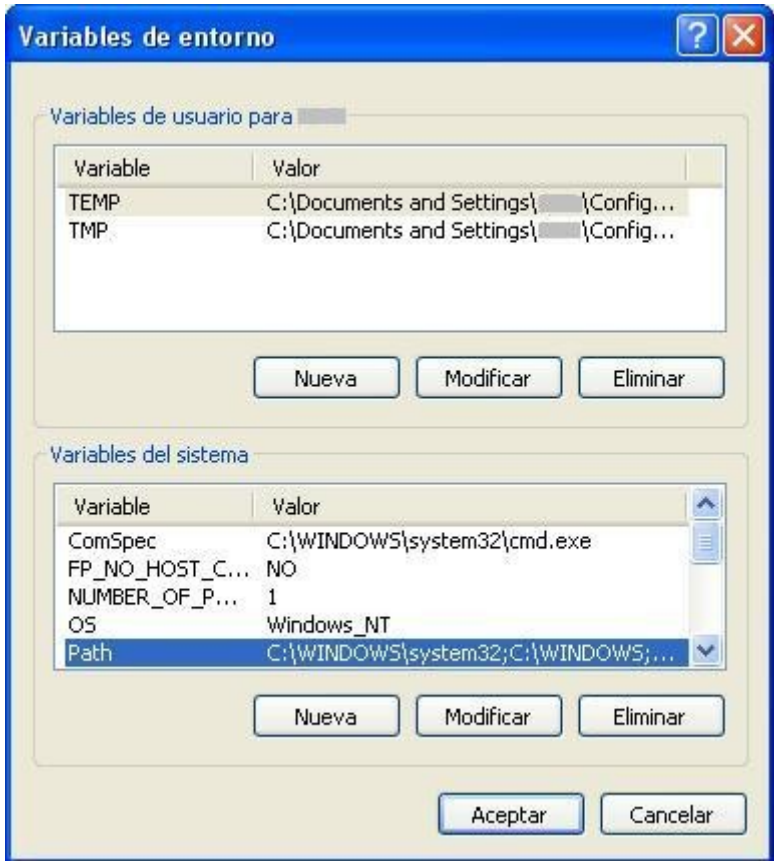

Ilustración 15: Variables de entorno

4.c CREACIÓN DEL PROYECTO

Para crear un proyecto en Symfony se utiliza la linea de comandos para ejecutar las instrucciones precisas, cada una de las cuales lleva a cabo una serie de tareas que de tener que realizarse de forma manual llevarían mucho más tiempo.

En primer lugar se iniciará una linea de comandos pulsando *Inicio/Ejecutar*, tecleando *cmd* y pulsando A*ceptar*. Utilizando los comandos apropiados habrá que situarse en la ruta de instalación de Wamp, crear un directorio para el nuevo proyecto y situarse en él.

```
cd c:\wamp\www
md symeval
cd symeval
```

Se creará el proyecto con el comando generate:project de Symfony:

```
php c:\wamp\www\symfony-1.4.16\data\bin\symfony generate:project symeval
```

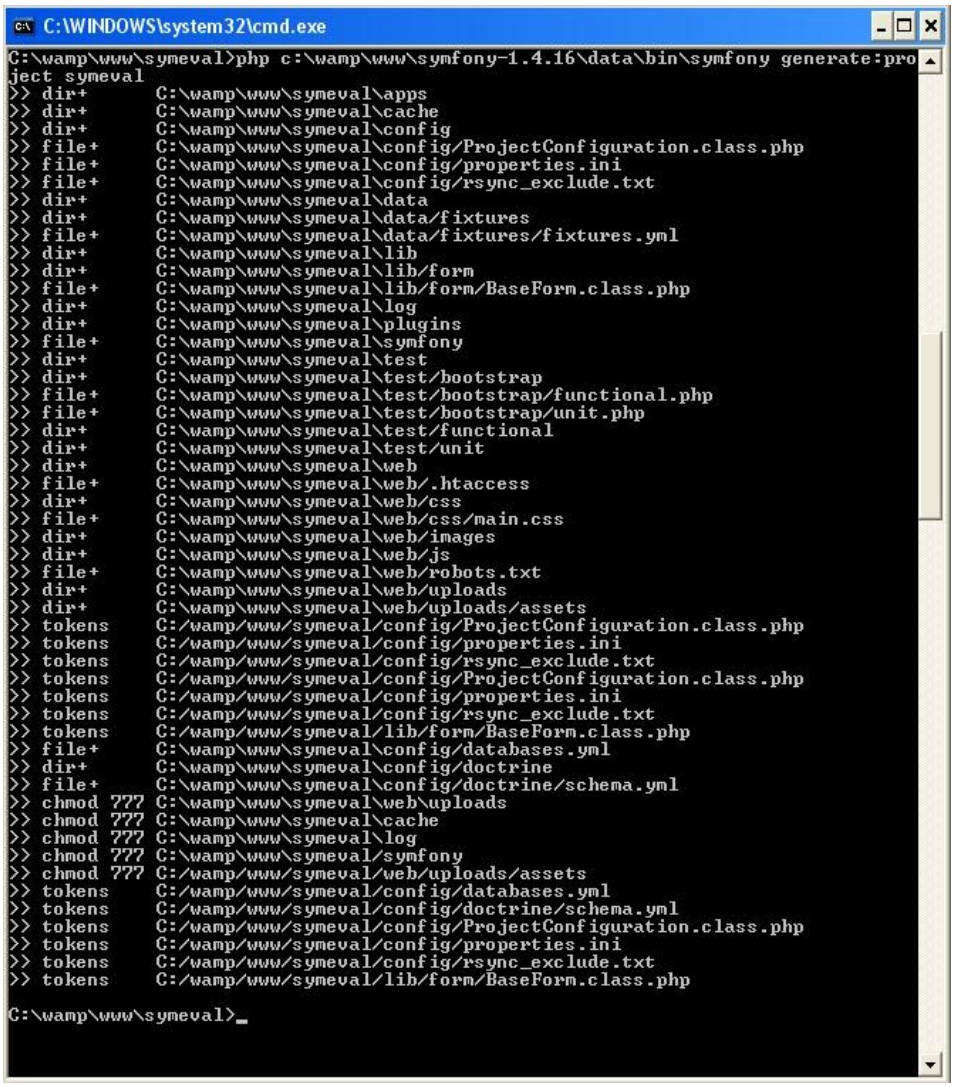

Ilustración 16: Symfony ejecutando comando de generación de proyecto

De esta manera se crea con un solo comando la estructura de directorios básica del proyecto.

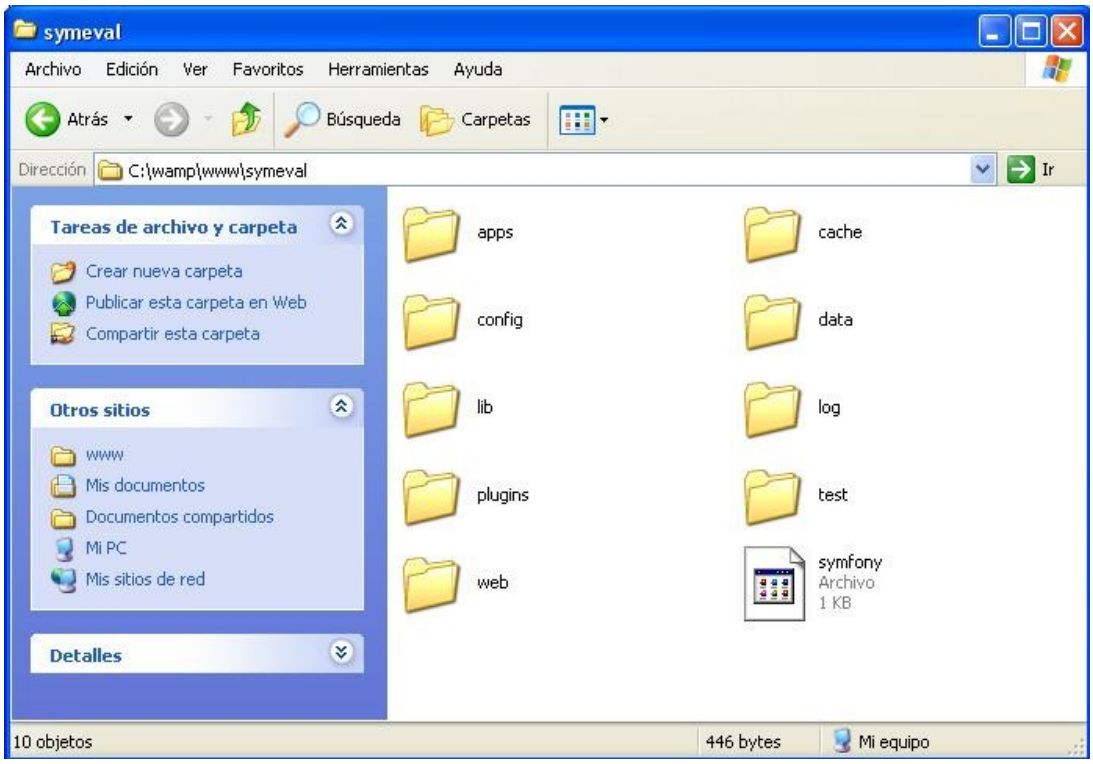

Ilustración 17: Estructura básica del proyecto

* No se puede crear un proyecto de esta manera en la versión sandbox de Symfony. Ver capítulo 5.b

Para crear el front-end del proyecto usaremos:

php symfony generate:app symeval

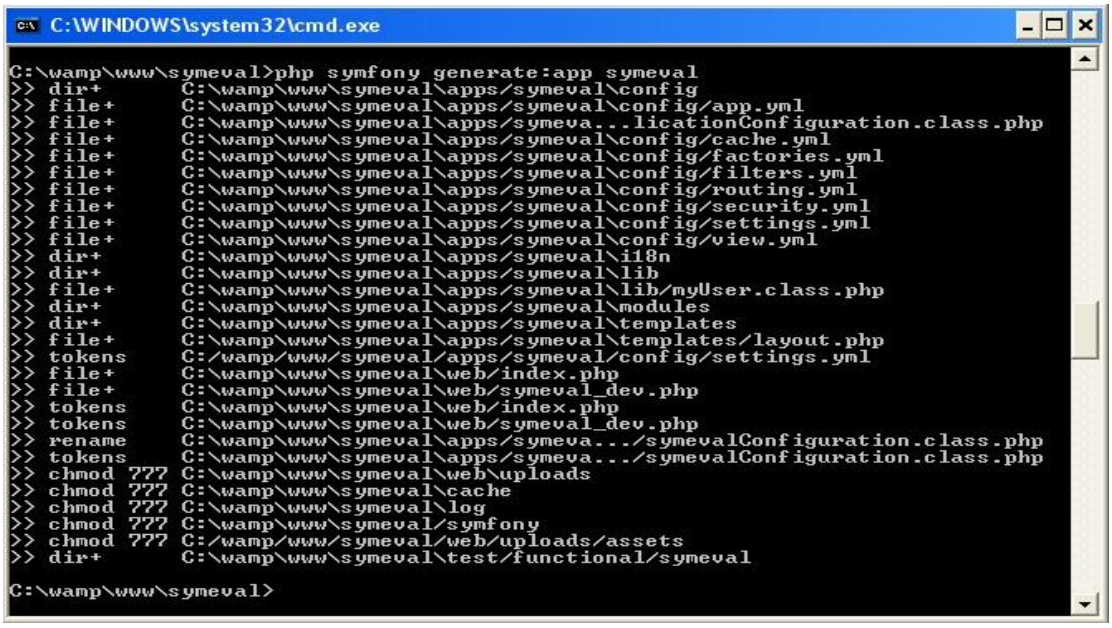

Ilustración 18: Symfony ejecutando comando para crear el front-end

Con este comando Symfony crea los archivos necesarios para el front-end como se puede comprobar en la ruta de la aplicación. El archivo index.php es el controlador frontal de la nueva aplicación. Al ser la primera aplicación creada, Symfony genera un archivo llamado index.php en vez de symeval.php. Si después se creara una nueva aplicación llamada back-end, el controlador frontal creado se llamaría back-end.php.

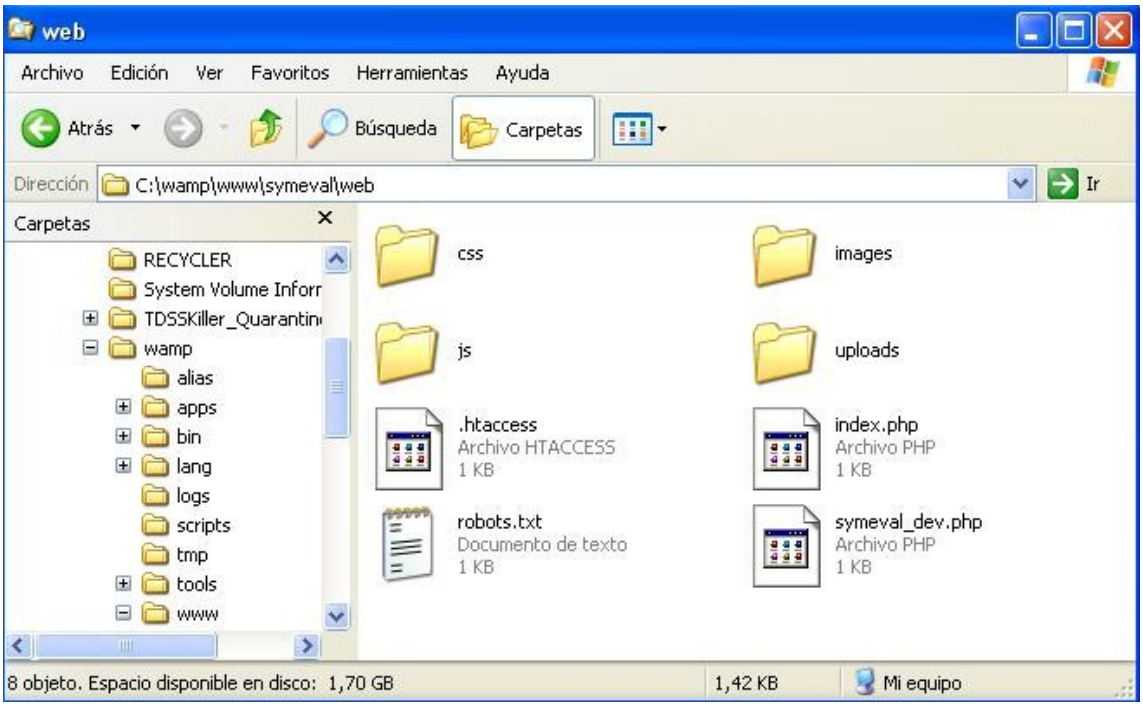

Ilustración 19: Archivos del front-end

En este punto la aplicación es accesible sin posibilidad de mostrar imágenes en la url localhost/symeval/web. Para mostrarlas es necesario configurar el servidor web para el proyecto como se verá en el siguiente punto.

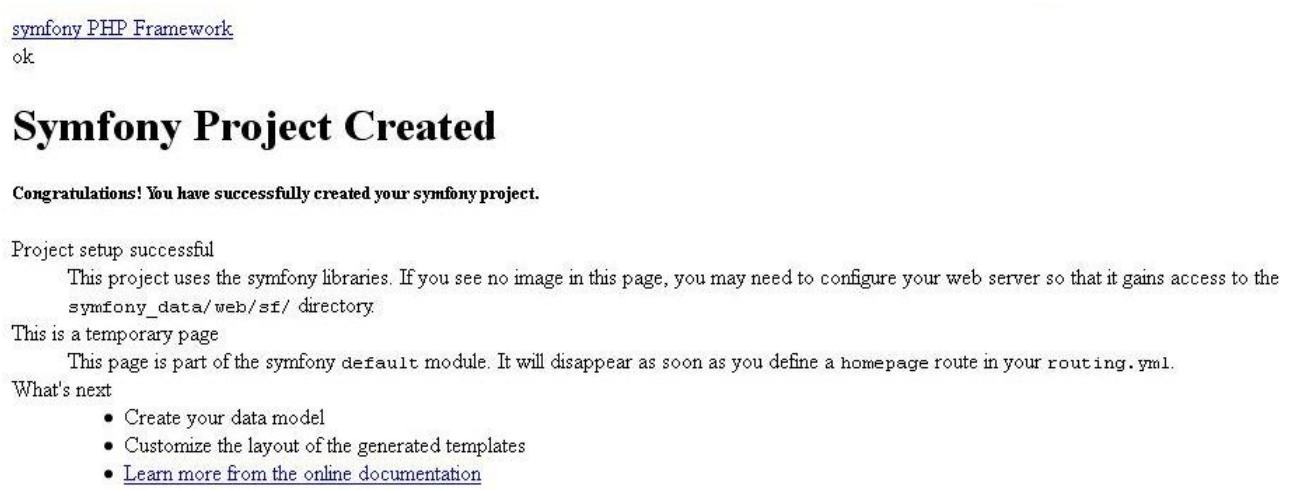

Ilustración 20: Página de inicio de proyecto Symfony antes de configurar el servidor web

4.d CONFIGURACIÓN DEL SERVIDOR WEB PARA EL PROYECTO

En este punto se explicará como crear un Virtual Host para el proyecto. De esta manera se podrá acceder al proyecto sin necesidad de especificar la ruta en la que se encuentra dentro de la estructura de directorios del servidor web. Podría hacerse usando para ello un número de puerto o un nombre. Es más conveniente elegir la opción del nombre, ya que en un entorno de desarrollo con un número elevado de proyectos es más eficaz y facil de recordar gestionarlos por sus nombres que por números de puerto. Asimismo, se configurará la ruta desde la que se obtienen las imágenes del proyecto. Repitiendo el proceso aquí explicado con direcciones IP internas diferentes se podrán gestionar de manera sencilla tantos proyectos como sean necesarios

Se modificará el fichero de configuración httpd.conf del servidor web, que se encuentra localizado en C:\wamp\bin\apache\Apache2.2.21\conf\, añadiendo:

```
<VirtualHost 127.0.0.2:80>
  ServerName symeval.localhost

  DocumentRoot "c:\wamp\www\symeval\web"
  DirectoryIndex index.php
  <Directory "c:\wamp\www\symeval\web">
    AllowOverride All
    Allow from All
  </Directory>

  Alias /sf c:\wamp\www\symfony-1.4.16\data\web\sf
  <Directory "c:\wamp\www\symfony-1.4.16\data\web\sf">
    AllowOverride All
    Allow from All
  </Directory>
</VirtualHost>
```

La finalidad del alias es permitir que Apache pueda encontrar las imágenes, hojas de estilos y archivos de JavaScript utilizados en la barra de depuración, en el generador automático de aplicaciones de gestión, en las páginas propias de Symfony y en las utilidades de Ajax. La alternativa a crear este alias podría ser la creación de un enlace simbólico (symlink) o copiar directamente los contenidos del directorio c:\wamp\www\symfony-1.4.16\data\web\sf al directorio C:\wamp\www\symeval\web\sf pero esto no se recomienda ya que de cambiar los archivos originales habría que volver a hacer la copia, y además repetir esta operativa para cada uno de los proyectos a los que se desearan propagar la actualización de archivos.

Para que sean efectivos los cambios del fichero de configuración es necesario reiniciar el servidor web. Esto se puede llevar a cabo por linea de comando o haciendo click en el icono de Wampserver en la barra de tareas y seleccionando Apache/Service/Reiniciar Servicio.

Ilustración 21: Reinicio de servidor web

En caso de existir algún error en el fichero el servidor web no podrá iniciarse, mostrándose el icono de Wamp en color naranja en lugar del verde habitual. 

Nótese que las rutas del fichero de configuración de Apache no pueden acabar con barra invertida. De ser necesario se pueden consultar los logs de accesos y errores del servidor web en C:\wamp\logs\ o desde el icono de Wampserver como se puede apreciar en la ilustración nº 21.

Es necesario además modificar el fichero hosts del sistema operativo. Este fichero se encuentra en C:\Windows\System32\drivers\etc\ y a través de él se establecen correspondencias entre direcciones URL y direcciones IP.
Se añadirá la entrada "127.0.0.2 symeval.localhost".

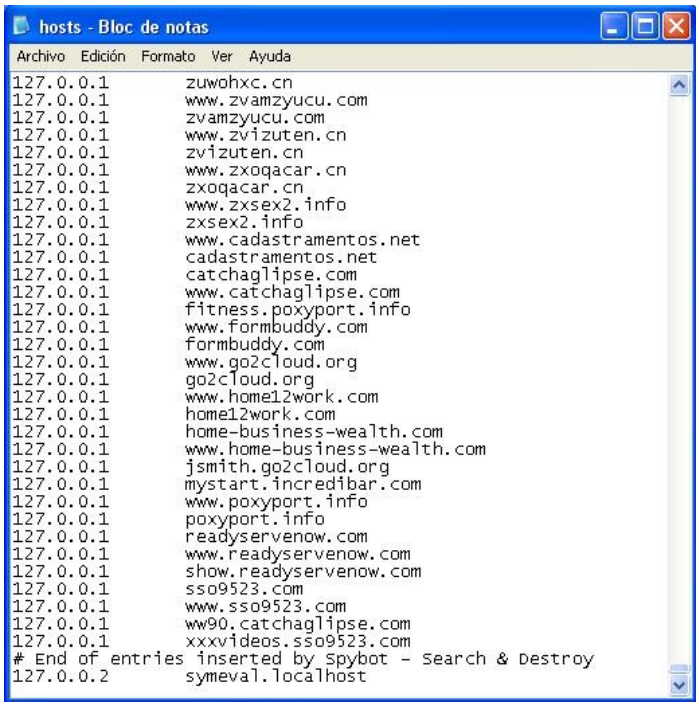

Ilustración 22: Fichero hosts

Se trata de un fichero de solo lectura por lo que para poder modificarlo es preciso cambiar primero sus permisos, y después de modificarlo restaurar sus permisos originales. Esto se puede conseguir por linea de comando con la instrucción *attrib -r hosts* o accediendo a las propiedades del archivo seleccionandolo con el botón derecho del ratón y desmarcando la propiedad "Sólo lectura".

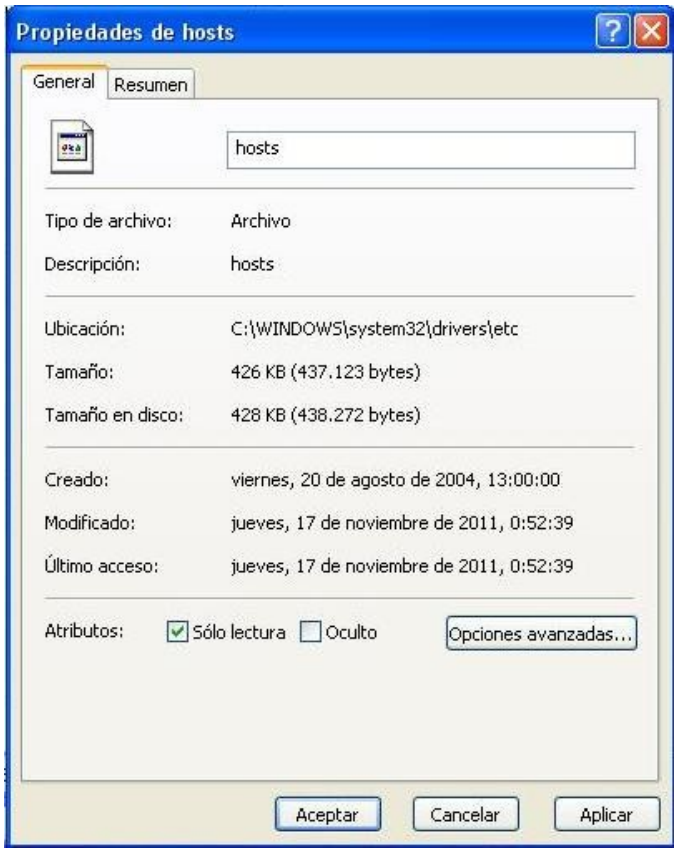

Ilustración 23: Permisos del fichero hosts

El fichero host es un objetivo para determinados malwares que pretenden redirigir la navegación web del usuario a páginas maliciosas, debido a ello puede (y debe) encontrarse protegido por programas de seguridad como Antivirus u otros antimalwares. Si se da el caso será necesario desactivar la protección o protecciones residentes existentes para poder modificarlo. Es importante proceder inmediatamente después de la modificación del fichero hosts a restaurar sus permisos originales y reactivar el o los programas de protección desactivados.

Ilustración 24: Protección residente

Una vez hecho esto la página de inicio del proyecto es accesible con imágenes y plantillas que se pueden mantener actualizadas de manera no manual. La URL de acceso al proyecto es ahora http://symeval.localhost, no siendo necesario indicar la ruta en la que se encuentra el proyecto dentro de la estructura de directorios del servidor web y fácil de recordar al tratarse del nombre del proyecto.

Ilustración 25: Página de inicio de Symfony con servidor web configurado

Los textos de la página se muestran en inglés, en el capítulo 6 se hablará de la personalización de las plantillas donde localizar el idioma, así como de cambiar esta página de inicio por defecto por la que corresponde a la aplicación desarrollada.

A través de la url http://symeval.localhost se accede al archivo index.php tal y como está indicado en el fichero de configuración de Apache. Sin embargo al ejecutar la aplicación en el entorno de desarrollo, es conveniente ejecutar el controlador frontal llamado symeval_dev.php, ya que ofrece información de depuración que resulta muy útil.
Por razones de seguridad el controlador frontal de desarrollo sólo se puede ejecutar por defecto desde localhost, de manera que se accederá a él desde la ruta http://localhost/symeval/web/symeval_dev.php

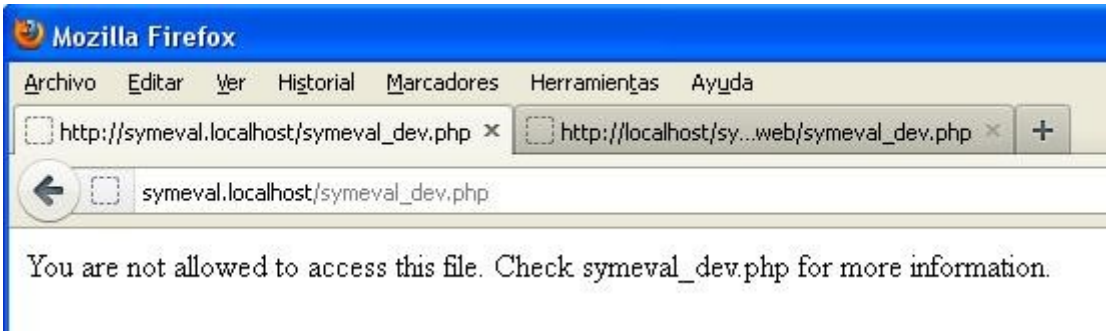

Ilustración 26: Restricción de acceso a controlador frontal de desarrollo de Symfony

Nótese que se ha elegido para el proyecto la IP interna 127.0.0.2. La dirección 127.0.0.1 ya está ocupada por la página de inicio de Wamp. De utilizarla para el proyecto Symeval la página de inicio de Wamp quedaría inaccesible. El objetivo de este proceder es poder llevar a cabo tantos proyectos diferentes como se desee en una misma máquina, cada uno alojado en su propia dirección IP interna y accesible tanto a través de esta dirección como de su nombre de proyecto, más fácil de recordar.

Asímismo en el caso de escribir las url hay que tener presente que no se debe acabar la ruta con "/" ya que daría error y no se mostraría la página deseada.

5. CREACIÓN DE LA BASE DE DATOS DE LA APLICACIÓN

Para interactuar con la BD de la aplicación se usará el ORM Doctrine que viene integrado con la versión de Symfony usada. Doctine es un mapeador relacional de objetos, un sistema que permite interactuar con los elementos de la BD como si fueran objetos. También dispone de instrucciones propias que automatizan tareas que sería más costoso realizar de otro modo.

5.a CREAR LA BD

Para hacer uso de esta funcionalidad Doctrine necesita conocer las tablas y relaciones de la DB, para crear las correspondientes clases con las instrucciones mencionadas. Esto se realiza a través del archivo de esquema localizado en C:\wamp\www\symeval\config\doctrine\schema.yml

Para crear la BD, desde una linea de comandos se ejecuta:

mysqladmin -uroot -create symeval

Con esto se obtiene en efecto una BD vacía, pero sin poder usarla en la aplicación. Para que Symfony quede configurado para usarla se ejecutará:

cd c:\wamp\www\symeval

php symfony configure:database "mysql:host=localhost;dbname=symeval" root

De no usar la metodología ORM ahora sería necesario crear los comandos SQL necesarios para crear las tablas de la base de datos o crearla a través de la interfaz de wamp, pero esto se realiza de manera más sencilla ejecutando tareas Doctrine que a su vez hacen uso de la definición realizada en el archivo de descripción de la base de datos (schema.yml)

php symfony doctrine:build-model
php symfony doctrine:build-sql
php symfony doctrine:insert-sql

Concretamente la tarea *build-model* crea archivos PHP en la ruta
C:\wamp\www\symeval\lib\model\doctrine

La tarea *build-sql* crea los comandos SQL en C:\wamp\www\symeval\data\sql

Al ejecutar la tarea *insert-sql* es cuando se crean las tablas en la BD

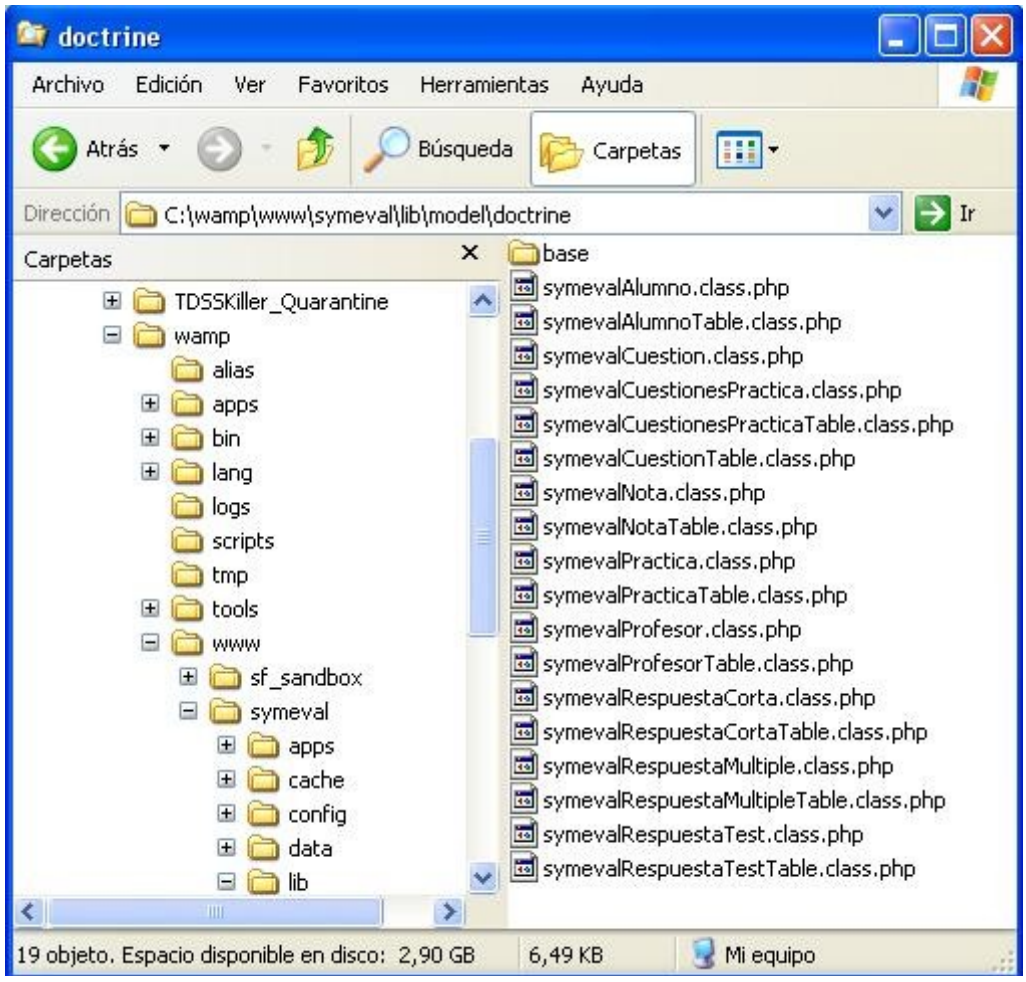

Ilustración 27: Archivos php generados por build-model

Se puede abreviar la ejecución de estas tareas aún más, ejecutando en su lugar una tarea que engloba a las tres anteriores:

php symfony doctrine:build --all –-no-confirmation

Ilustración 28: Ejecución de doctrine:build-all

En este punto es necesario liberar la caché de Symfony. Esto se debe a que Symfony carga automáticamente las clases php sin necesidad de usar instrucciones *require*, y para ello hay que realizar esta limpieza cada vez que se añade una nueva clase en el directorio *lib*, cosa que ha hecho la tarea *build-model.* Esto se realiza ejecutando:

```
php symfony cc
```

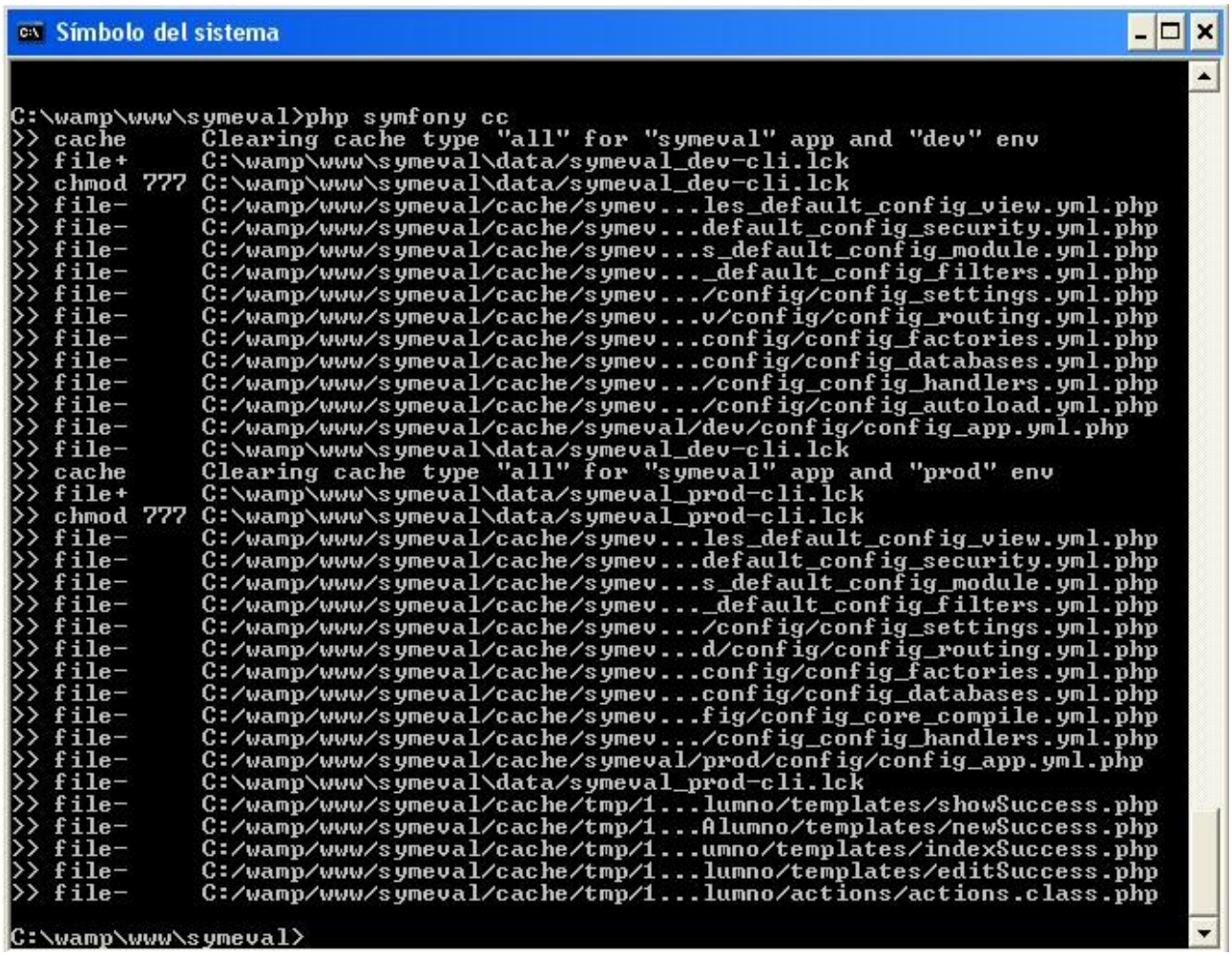

Ilustración 29: Limpiando la caché

5.b CREAR DATOS INICIALES EN LA BD

La BD que acaba de ser creada tiene tablas, pero no datos. Una vez más Doctrine acelerará la tarea de crearlos. En lugar de tener que crear sentencias sql o introducir datos a mano con una interfaz, se usará la tarea *data-load*, que introduce los datos en la BD a partir de los archivos *yml* presentes en la ruta C:\wamp\www\symeval\data\fixtures

La estructura de estos archivos se puede ver en el anexo IV del presente documento.

Una vez creados los archivos se ejecutará:

php symfony doctrine:data-load

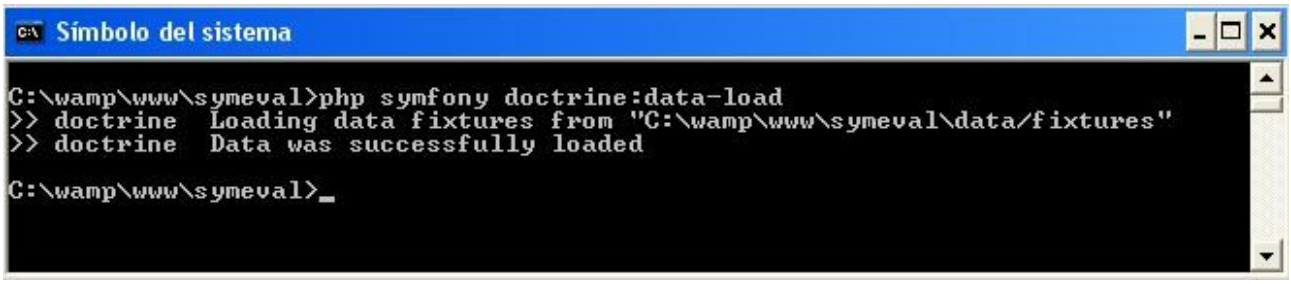

Ilustración 30: Introduciendo los datos iniciales

6. DESARROLLO DE LA APLICACIÓN

Symfony proporciona funcionalidad para crear páginas de listado de datos y formularios con capacidad de edición, borrado y entrada de registros en la BD creada en el capítulo anterior. Estos formularios poseen además capacidad de validación básica de datos. El alcance de estas validaciones es limitado, soportando la detección de datos requeridos que no se han introducido, tipos de datos incorrectos... sin embargo Symfony no válida en la versión usada la correción de un NIF, por ejemplo. Este tipo de validaciones de ser necesarias deben realizarse explícitamente.

6.a CREACIÓN DE LOS MÓDULOS Y FORMULARIOS

Para crear los primeros formularios de la aplicación se usará la instrucción doctrine:generate-module. De esta forma, para crear el módulo de los alumnos se ejecutará:

php symfony doctrine:generate-module --with-show --non-verbose-templates symeval alumno symevalAlumno

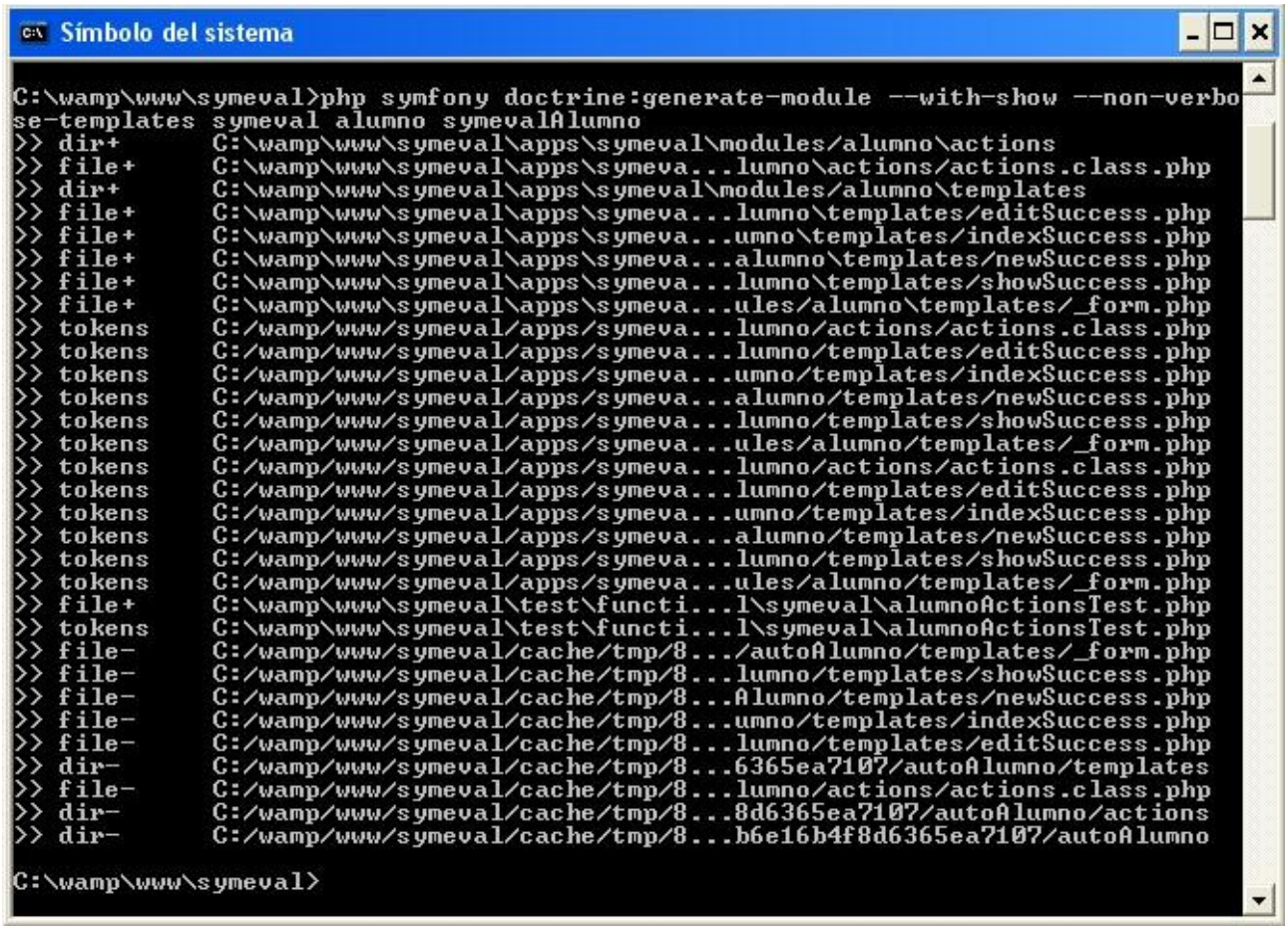

Ilustración 31: Generando módulo

Con esto, se dispone del módulo de los alumnos operativo. Es posible acceder a la página de listado de los alumnos en la url http://symeval.localhost/alumno y si se desea acceder a través del front-end de desarrollo usando la url http://localhost/symeval/web/symeval_dev.php/alumno

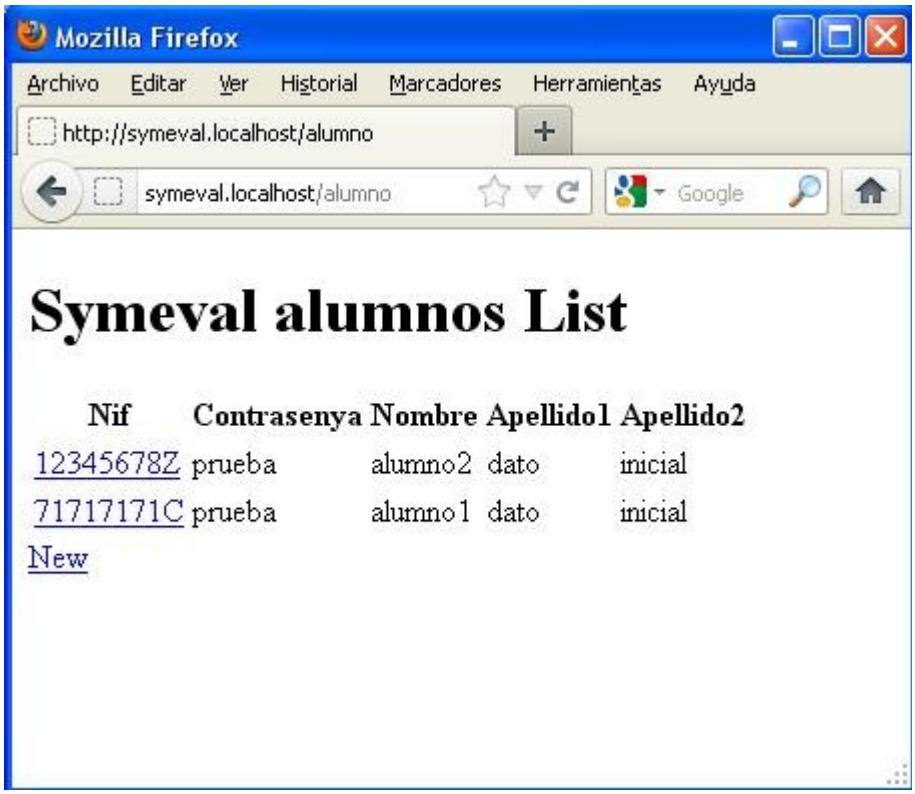

Ilustración 32: Módulo alumno

Desde esta página es posible crear nuevos alumnos en la BD, y acceder a los formularios de edición y borrado seleccionando alguno de los alumnos existentes. No obstante, tanto la página de listado como los formularios tienen un aspecto sencillo y palabras en inglés. En los apartados siguientes se realizarán las personalizaciones pertinentes.

Se repetirá la tarea *doctrine:generate-module* para el resto el resto de módulos/formularios que se deseen crear de este modo:

- php symfony doctrine:generate-module --with-show --non-verbose-templates symeval profesor symevalProfesor
- php symfony doctrine:generate-module --with-show --non-verbose-templates symeval practica symevalPractica
- php symfony doctrine:generate-module --with-show --non-verbose-templates symeval respuestaCorta symevalRespuestaCorta
- php symfony doctrine:generate-module --with-show --non-verbose-templates symeval respuestaTest symevalRespuestaTest

- php symfony doctrine:generate-module --with-show --non-verbose-templates symeval respuestaMultiple symevalRespuestaMultiple

6.b PERSONALIZACIÓN VISUAL

Para personalizar el aspecto visual de la aplicación es necesario crear y editar tanto código como imágenes. Para el código se ha usado Notepad++, para la creación de imágenes se recomienda The Gimp.

6.b.i Personalización de la plantilla global

Symfony utiliza en la aplicación una plantilla global llamada layout que se encuentra localizada en C:\wamp\www\symeval\apps\symeval\templates\layout.php

El resto de plantillas de la aplicación se decoran después que el contenido sea mostrado por la plantilla global. Se personalizará el aspecto global de la aplicación en esta plantilla, en la que se hará uso de imágenes y hojas de estilo.

Symfony crea directorios por defecto para almacenar las hojas de estilo (archivos .css) y las imágenes en C:\wamp\www\symeval\web\css y C:\wamp\www\symeval\web\images

Las hojas de estilo que serán llamadas desde la plantilla global (por defecto main.css)se definen en el archivo C:\wamp\www\symeval\apps\symeval\config\view.yml

Una vez creados estos contenidos el aspecto mejora considerablemente, no obstante la información del listado de alumnos sigue sin mostrarse de manera atractiva, y hay textos que se muestran en inglés. Este contendido depende de una plantilla local del módulo Alumno, cuya personalización se realiza en el siguiente apartado.

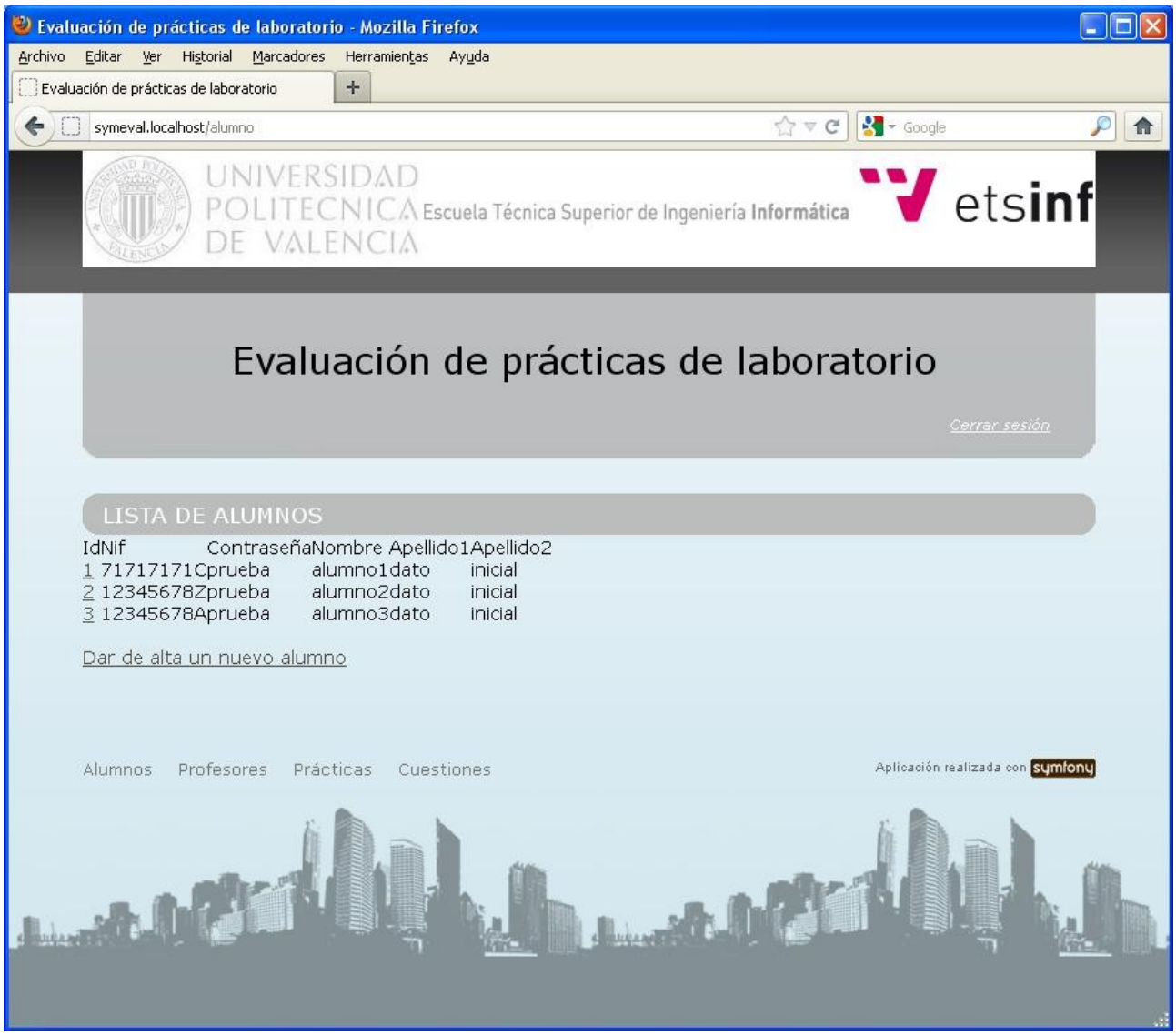

Ilustración 33: Personalización de la plantilla global

6.b.ii Personalización de plantillas locales

Las plantillas locales del módulo Alumno se encuentran localizadas en C:\wamp\www\symeval\apps\symeval\modules\alumno\templates\ y deben ser modificadas para mejorar el aspecto de la aplicación añadiendo el código necesario para referenciar a las hojas de estilo e imágenes que sean necesarias, que a su vez se crearán en los directorios *css* e *images* mencionados en el apartado anterior.

Las hojas de estilo que deben ser utilizadas desde una plantilla local se invocan desde el archivo *view.yml* de ese módulo localizado en C:\wamp\www\symeval\apps\symeval\modules\alumno\config\view.yml o desde la plantilla local del módulo correspondiente, que es la manera utilizada en este caso.

Así pues, se modificarán todas las plantillas, tanto para mejorar el aspecto de la aplicación como para mostrar solo los datos deseados y localizar el lenguaje**.**

indexSuccess.php es la plantilla del **listado de alumnos** y usará la hoja de estilos *alumnos.css*

showSuccess.php es la plantilla de **visualización de datos de un alumno**, desde la cual se puede eliminar o modificar sus datos y usará la misma hoja de estilos.

editSuccess.php es la plantilla de **edición de un alumno** y usará la misma hoja de estilos.

newSuccess.php es la plantilla de **alta de nuevo alumno** y usará la misma hoja de estilos.

form.php es una plantilla de formulario usada por las plantillas editSuccess.php y newSuccess.php

Ilustración 34: Personalización de plantillas locales

En este punto se encuentra personalizado visualmente y localizado el módulo Alumno de la aplicación. Para personalizar completamente toda la aplicación se repetirán las actuaciones recogidas en este apartado para el resto de módulos.

Nótese que aún así faltará personalizar el aspecto de parte de los formularios de edición y creación de datos de la aplicación. Esto se realizará en un apartado posterior al estar relacionado con las personalizaciones funcionales.

6.b.iii Personalización de plantillas de error

Symfony dispone de plantillas de error tanto en el entorno de desarrollo como en el de producción. Se usan para las páginas mostradas al producirse errores 404 (página no encontrada) o errores de autenticación, entre otros. Como el resto de plantillas, se encuentran en inglés y al menos en el entorno de producción es necesario mostrar los textos localizados al castellano, de modo que se crearán plantillas para el proyecto. También sería posible modificar las que lleva Symfony por defecto en C:\wamp\www\symfony-1.4.16\lib\controller\default, el inconveniente de hacer esto último es que las plantillas modificadas serían las mismas para todos los proyectos alojados en el servidor, y algunas de las configuraciones realizadas en apartados anteriores estaban enfocadas precisamente a permitir la coexistencia de varios proyectos en la misma máquina, por lo tanto se crearán plantillas nuevas especificas para el proyecto en C:\wamp\www\symeval\apps\symeval\modules\default

5.c PERSONALIZACIÓN FUNCIONAL

5.c.i Personalización de consultas

Al crear una nueva práctica o editar las existentes, el formulario muestra un listado con los identificadores númericos de profesores disponibles. No es eficaz seleccionar un dato a partir de su identificador.

Para que se muestre el nombre y apellidos o los datos que sean convenientes en vez de los identificadores se crearán métodos __toString() para las clases que lo necesiten, en este caso C:\wamp\www\symeval\lib\model\doctrine\symevalProfesor.class.php

Esta característica de Symfony es muy útil en muchas ocasiones, por ejemplo para mostrar los enunciados de las cuestiones al crear prácticas en lugar de los identificadores. Sin embargo para el caso de los profesores no es suficiente, ya que debe guardarse automáticamente el profesor que la crea o modifica. Para este tipo de modificaciones más profundas es necesario personalizar el formulario como se indica en el apartado siguiente, así como codificar las consultas que sean necesarias en C:\wamp\www\symeval\lib\model\doctrine

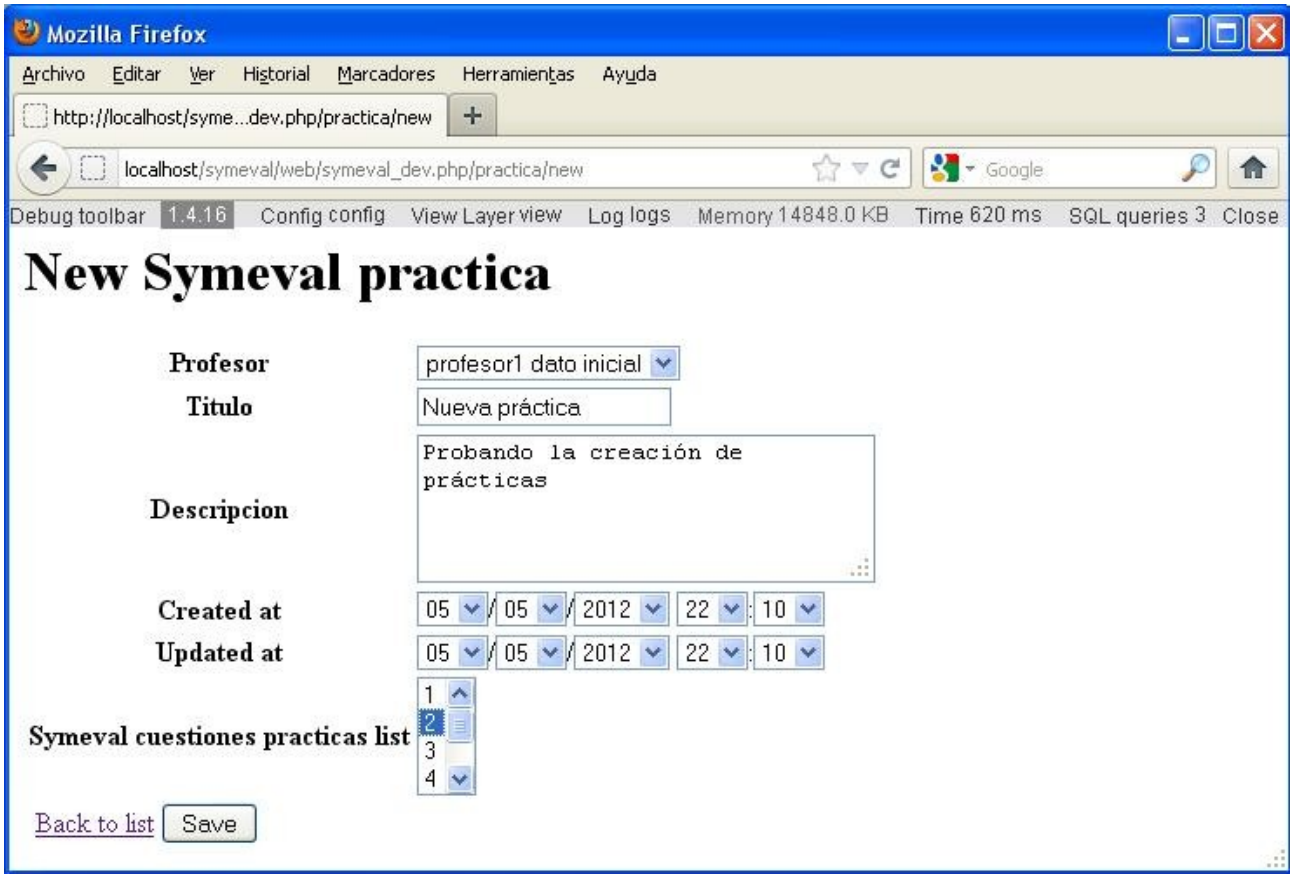

Ilustración 35: Formulario de edición sin usar plantillas de personalización visual

6.c.ii Personalización de formularios

De la misma manera que en las páginas para visualizar datos, en los formularios de edición y creación de nuevos contenidos es necesario realizar personalizaciones para mostrar solo los campos que sean necesarios. Esta necesidad se hace patente especialmente en las clases que usan herencia. Asimismo se puede observar en la ilustración 36 como Symfony crea etiquetas de texto para el formulario tomando como base los nombres de los campos de la BD. Esto debe ser modificado para no revelar la estructura interna de la BD así como para ofrecer al usuario nombres más intuitivos y/o traducidos. También será necesario modificar el tamaño por defecto de los campos del formulario para adecuarlos a los datos de la aplicación y mejorar la experiencia del usuario, introducir valores por defecto, modificar las consultas o realizar postvalidaciones.

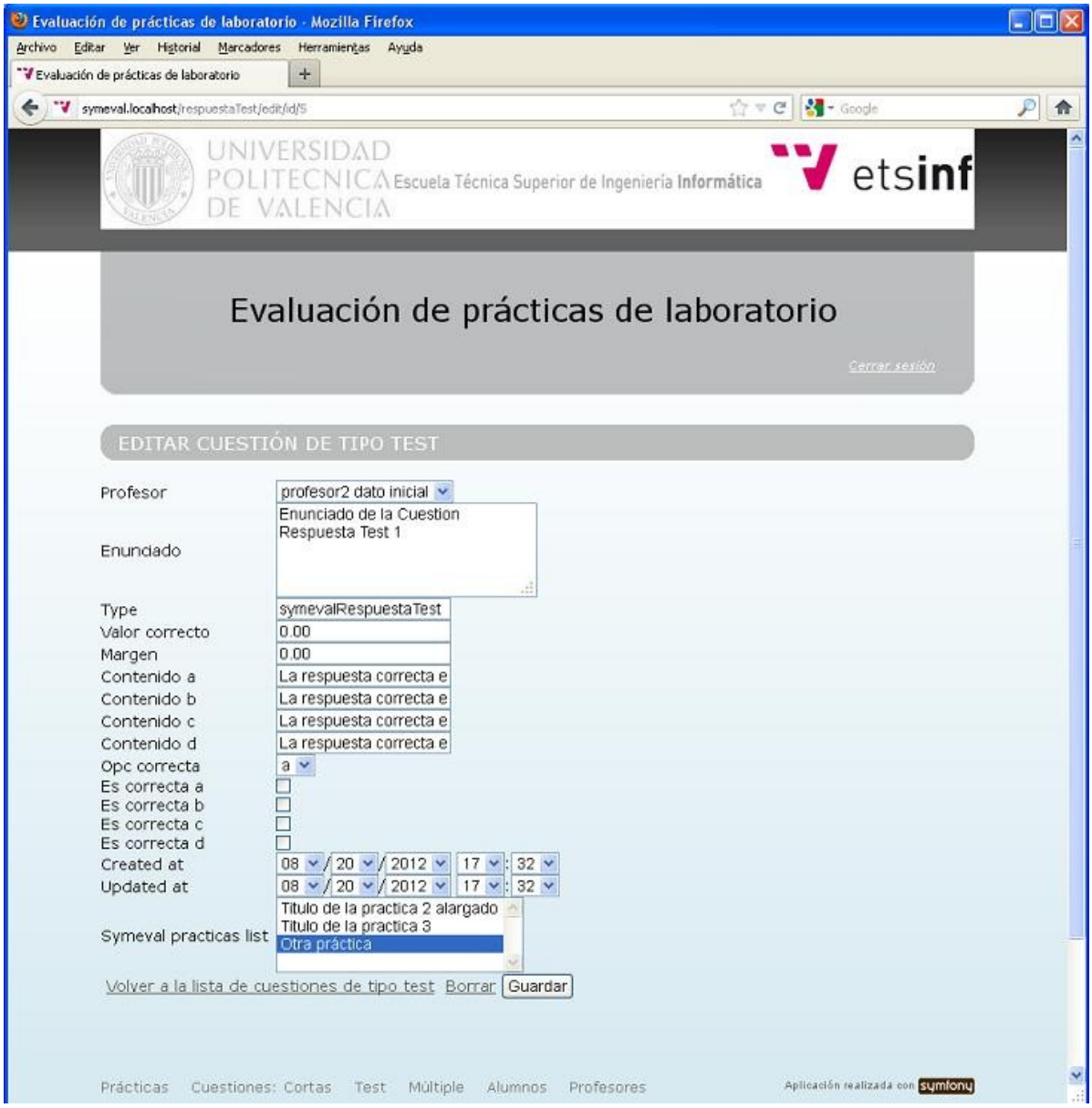

Ilustración 36: Formulario de edición sin personalización funcional

Los cambios mencionados se implementarán en las plantillas de formularios, que se encuentran en C:\wamp\www\symeval\lib\form\doctrine

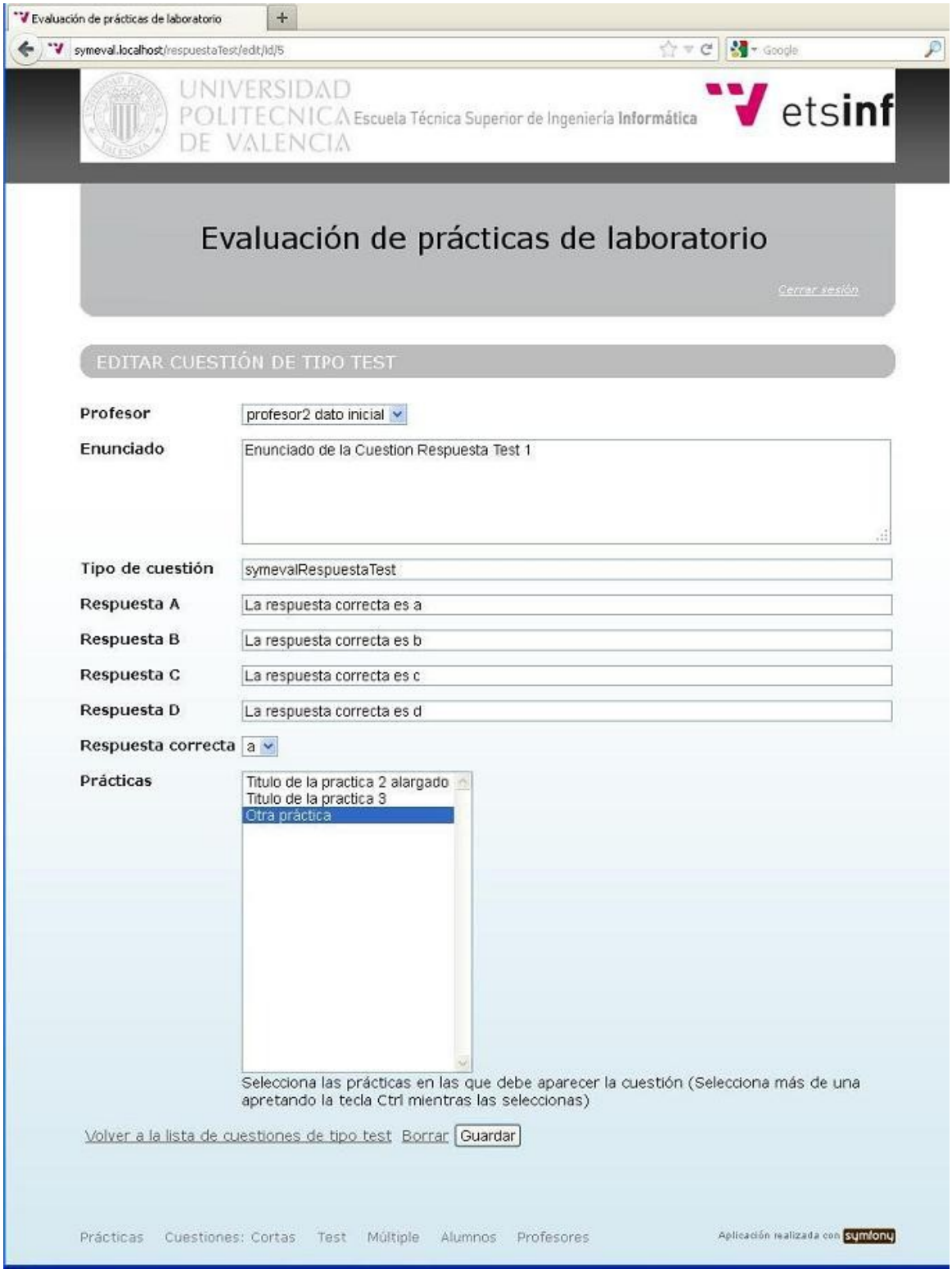

Ilustración 37: Formulario de edición personalizado

6.d **CREACIÓN DE NUEVAS FUNCIONALIDADES**

Hasta ahora se ha desarrollado la aplicación por linea de comando y modificando archivos ya existentes, pero para entrar a las distintas secciones hay que empezar escribiendo la url de un módulo. La página de inicio sigue siendo la página por defecto de Symfony. En este capítulo se explicará como cambiarla por una página creada codificando desde cero.

6.d.i Creación de la página de inicio de sesión

La página de inicio de la aplicación debe ser, de acuerdo con el análisis de requisitos y los casos de uso, la página donde los usuarios realicen el inicio de sesión. Todas las páginas de que consta la aplicación están contenidas en módulos que corresponden a alguna clase del modelo UML, concretamente cada página es una *acción* de un *módulo.*

Para crear páginas completamente nuevas en Symfony es necesario crear nuevas acciones y sus plantillas asociadas, ya sea en un módulo nuevo o en alguno de los módulos existentes. No tiene sentido crear una clase login en un modelo UML, de ahí que no se haya creado anteriormente por linea de comando un módulo login para la aplicación.

Para crear la página de inicio de sesión, en lugar de crear un módulo nuevo se usará el ya existente módulo de usuarios, creando la correspondiente acción *executeLogin* en C:\wamp\www\symeval\apps\symeval\modules\alumno\actions

Nótese que una *acción* es una función php, de manera que podría resumirse de forma escueta y simplificada que se puede crear una nueva página no contemplada en el modelo UML creando una función php y una plantilla asociada. Además las variables creadas en la *acción* son accesibles desde la plantilla, junto a otras variables como sf_user y sf_request. Será habitual necesitar también crear alguna consulta que se invoque desde la *acción* y cuyos datos se muestren desde la plantilla. Esto es lo que se conoce como arquitectura MVC, Modelo (~consulta) – Vista (~plantilla) – Controlador (*~acción*)

En este caso, como la página no se limita a mostrar datos sino que es un formulario de inicio de sesión es necesario también un formulario que recoja los datos de inicio de sesión (nif y contraseña). En apartados anteriores se ha mencionado el emplazamiento de los formularios en la estructura de directorios de Symfony, de esta manera se creará el nuevo formulario *LoginForm.class.php* en C:\wamp\www\symeval\lib\form

Resta pues crear la plantilla *loginSuccess.php* en su correspondiente ruta del módulo alumno C:\wamp\www\symeval\apps\symeval\modules\alumno\templates

El resto de páginas y funcionalidades requeridas en la aplicación se crearán de la misma manera.

6.d.ii Configuración de la página de inicio de la aplicación

Para que la página de inicio de sesión creada en el apartado anterior quede definida como página de inicio de la aplicación o *homepage* se configurará como tal la *acción* en el archivo routing.yml localizado en C:\wamp\www\symeval\apps\symeval\config

Después de reconfigurar el archivo routing.yml es necesario limpiar la caché de Symfony. Esto se lleva a cabo ejecutando desde la linea de comandos *php symfony cc*, tal y como se indicó en un capítulo anterior. De lo contrario se producirán problemas de acceso.

Debe recordarse que la limpieza de caché debe realizarse de hecho siempre que se añada alguna clase en el directorio *lib,* ya que como se ha mencionado, Symfony carga automáticamente las clases php y gracias a ello no es necesario incluir sentencias *require* al implementar código.

6.d.iii Establecimiento de credenciales de acceso

Para restringir el acceso a las páginas a los usuarios que tengan las credenciales de acceso necesarias se realizarán las configuraciones pertinentes en los ficheros security.yml, que deben ser creados para los módulos que los requieran en la ruta C:\wamp\www\symeval\apps\symeval\modules\<*modulo*>\config

Las mencionadas credenciales se otorgan a los usuarios autentificados durante el proceso de *login* o inicio de sesión, y se eliminarán durante el proceso de *logout* o cierre de sesión.

6.e EL CONTROLADOR

Se ha comentado en el apartado 6.d.i que Symfony hace uso de la arquitectura MVC siglas de Modelo - Vista - Controlador. El objetivo que se persigue con esta arquitectura es separar la codificación de las páginas de la presentación y visualización de las mismas, así como del almacenamiento de la información o capa de persistencia.

En Symfony la vista es equivalente a las plantillas comentadas en apartados anteriores. El modelo es casi equivalente a la base de datos de la aplicación. El "casi" se debe a que íntimamente relacionadas con la base de datos están las consultas, a medio camino entre modelo y controlador.

El controlador en Symfony es equivalente a las acciones. Desde ellas se inicia el flujo de datos al acceder a una página de la aplicación. En las ilustraciones nº 38 y 39 se muestra de forma gráfica esta equivalencia.

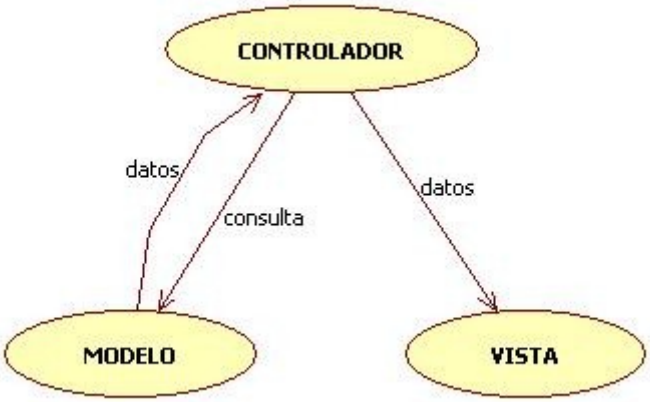

Ilustración 38: Esquema de flujo de datos en la arquitectura MVC

Ilustración 39: Esquema de flujo de datos en la arquitectura MVC de Symfony

Para ejemplificar lo dicho, se usará como ejemplo la página de listado de notas.

Cuando un alumno selecciona la opción "Mis notas" accede a la página listanotas. El flujo de ejecución se inicia en la acción (controlador) *executeListanotas* añadida en el archivo de acciones del módulo Alumno, localizado en:
C:\wamp\www\symeval\apps\symeval\modules\alumno\actions\actions.class.php

Veáse como la acción es una función php añadida:

```
 public function executeListanotas(sfWebRequest $request)
 {
        $this->symeval_notas = symevalAlumnoTable::notas($this->getUser()->getAttribute("id"));

        if(count($this->symeval_notas) < 1){

                $this->getUser()->setFlash("notice", "No tienes aún ninguna nota. Resuelve alguna
práctica.");
                $this->redirect('@alumno_practicas');
        }
 }
```

La acción (controlador) hace uso de la consulta symevalAlumnoTable::notas($this->getUser()->getAttribute("id")) para acceder a la base de datos (modelo) y obtener los datos que debe mostrar la página. Para poder hacer uso de la consulta previamente hay que crearla añadiendo el código a C:\wamp\www\symeval\lib\model\doctrine\symevalAlumnoTable.class.php

```
        public static function notas($id){

    return Doctrine_Query::create()

      ->from('symevalNota n')

        ->where('n.alumno_id = ?', array($id))

        ->execute();

  }
```

El paso final, la visualización de los datos es tarea de la plantilla (vista) listanotasSuccess.php creada en C:\wamp\www\symeval\apps\symeval\modules\alumno\templates\

```
<?php use_stylesheet('practicas.css') ?>

<h1>Lista de notas</h1>

<div id="practicas">
<table class="practicas">
```

```
 <thead>
  <tr>
    <th>Práctica</th>
          <th>Nota</th>
          <th>Fecha</th>
          <th>Id alumno</th>
  </tr>
 </thead>


 <tbody>
  <?php foreach ($symeval_notas as $symeval_nota): ?>
  <tr>
    <td class="descripcion"><?php echo symevalPracticaTable::practicaPorId($symeval_nota-
>getPractica_id())->getTitulo() ?></td>
          <td class="nota"><?php echo $symeval_nota->getNota() ?></td>
    <td class="fecha_creacion"><?php echo $symeval_nota->getCreatedAt() ?></td>
          <td class="id"><?php echo $symeval_nota->getAlumno_id() ?></td>
  </tr>
  <?php endforeach; ?>
 </tbody>
</table>
</div>
```

La plantilla hace uso de los datos pasados por el controlador en forma de variables, para mostrarlos haciendo uso de la hoja de estilos practicas.css. La hoja de estilos es invocada a través del *helper* use_stylesheet('practicas.css'). Un *helper* es una función que empaqueta código usado con mucha frecuencia en plantillas. Con el *helper use_stylesheet('practicas.css')* evitamos tener que incluir la etiqueta <link>. En la aplicación se hace uso de este *helper* en todas las plantillas para invocar las hojas de estilo correspondientes. Otro *helper* muy usado es *url_for()* para mostrar las direcciones URL en los enlaces.

Para el desarrollo de la aplicación se han modificado todas las plantillas y formularios del esquema generado por Symfony y se han creado las consultas, hojas de estilo, acciones y nuevas plantillas necesarias.

En Symfony las acciones de un módulo se almacenan por convención en C:\wamp\www\symeval\apps\symeval\modules\<modulo>\actions\actions.class.php

Las consultas se deben crean todas en la ruta C:\wamp\www\symeval\lib\model\doctrine

Las plantillas en C:\wamp\www\symeval\apps\symeval\modules\<modulo>\templates\

Las hojas de estilo en C:\wamp\www\symeval\web\css

Cuando se hace uso de formularios el flujo de datos cambia, ya que a todas las acciones que realiza el controlador se suma la creación del formulario y su paso a la plantilla. El formulario a su vez puede acceder al modelo. El flujo de datos de la arquitectura MVC en Symfony al hacer uso de formularios se puede apreciar en la ilustración nº 40.

Los formularios del esquema se encuentran en C:\wamp\www\symeval\lib\form\doctrine, los creados se deben situar en C:\wamp\www\symeval\lib\form\

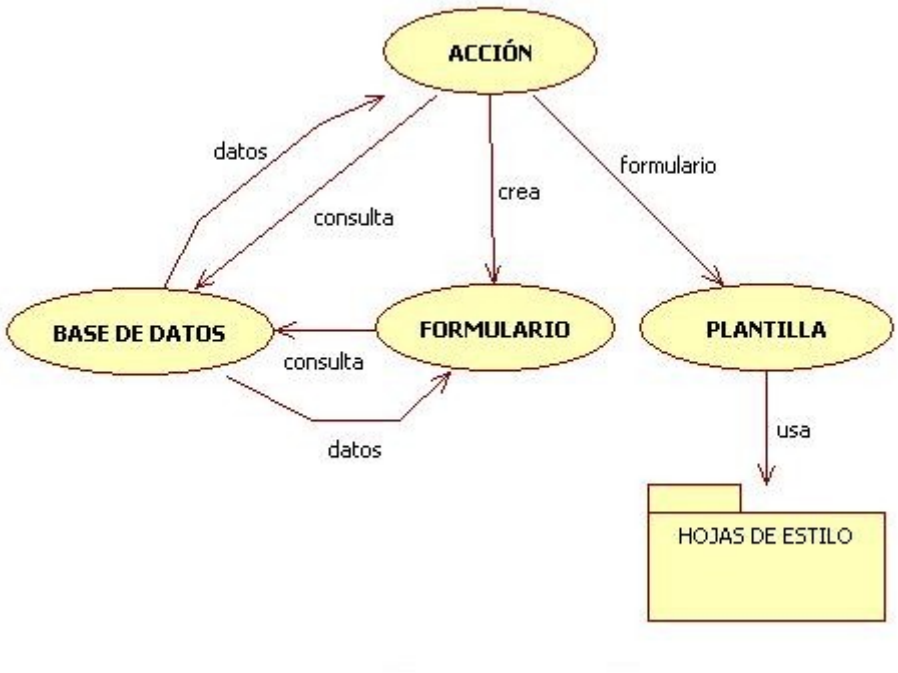

Ilustración 40: Esquema de flujo de datos con formulario en la arquitectura MVC de Symfony

Para mostrar el proceso del controlador cuando se crea un formulario se usará como ejemplo el proceso de corrección de una práctica:

En primer lugar en la acción (controlador) *executeResolverPractica* se crea el formulario y se accede al modelo para obtener datos de la práctica.

```
$this->form = new ResolverPracticaForm();

if ($practica = symevalPracticaTable::practicaPorId($this->idprac)) {

        $this->titulo_practica = $practica->getTitulo();
        $this->descripcion_practica = $practica->getDescripcion();
        $this->cuestiones_practica = $practica->getSymevalCuestionesPracticas();
}
```

Como se ha mencionado, el formulario ResolverPracticaForm.class.php se crea en la ruta C:\wamp\www\symeval\lib\form\

Desde el formulario se accede a su vez al modelo para obtener las cuestiones. El controlador por su parte pasa a la plantilla resolverPracticaSuccess.php, creada en la ruta C:\wamp\www\symeval\apps\symeval\modules\alumno\templates\ tanto los datos de la práctica como el propio formulario, y en la plantilla se muestran haciendo uso de la hoja de estilos resolverpractica.css, creada en la ruta anteriormente mencionada.

El alumno rellena las respuestas y al pulsar el botón "corregir" Symfony realiza las validaciones básicas inherentes a los formularios, las postvalidaciones y retorna al controlador donde finaliza el flujo de datos realizando las operaciones pertinentes, en este caso redirigiendo a la página de prácticas. Las postvalidaciones son operaciones que se realizan después de las validaciones estándar de Symfony. Para que un formulario realice postvalidaciones es necesario implementar en el formulario la invocación a la postvalidación y la función que la contenga.

```
$this->validatorSchema->setPostValidator(
  new sfValidatorCallback(array('callback' => array($this,'corregir'))));

  public function corregir($validator, $values)
  { .. }
```

Es en la postvalidacion "corregir" donde se efectua la corrección de la práctica y cálculo de la nota, cumpliendo de esta manera con la convención de desarrollo de software de mantener en la medida de lo posible el código que efectua una operación completa en un mismo archivo. El cálculo se efectua de manera dinámica independientemente del número de cuestiones. Se comprueban las respuestas del alumno a las cuestiones en el buble *foreach($cuestionespr as $cuestionpr)* y se almacena el número de aciertos conseguidos y el número total de cuestiones. Al finalizar el bucle, el cálculo se efectua en la instrucción *$nota = round($numaciertos * (10.0/$cuentacuest),2);*

Un alumno puede realizar una práctica únicamente una vez. Una vez hecha ya no tendrá disponible el enlace de acceso en el listado de prácticas. No obstante al finalizar el cálculo de la nota se comprueba si el alumno dispone ya de nota para la práctica que está resolviendo, en caso de que no tenga nota, el sistema almacena la nota obtenida y establece un mensaje que notificará al alumno de su nota. Si el alumno ya tenía nota, se conserva la nota anterior y el mensaje notificará al alumno de ello sin mostrarle la nueva nota. Se evita de este modo que se averigue qué respuestas son correctas por prueba y error.

```
// chk no tenga ya nota
if (!$nota_prac = symevalNotaTable::notaPorIds($values['idprac'],$idusr)) {
// guardar nota
sfContext::getInstance()->getUser()->setFlash("notice", $variables);

$nota_prac = new symevalNota();
$nota_prac->setPracticaId($values['idprac']);
$nota_prac->setAlumnoId($idusr);
```

```
$nota_prac->setNota($nota);
$nota_prac->save();
}
else {
//si ya tiene nota ya ha resuelto la práctica, está haciendo trampas
sfContext::getInstance()->getUser()->setFlash("error", "Ya has resuelto esta práctica anteriormente. No se guardará la nota nueva.");
```

6.f REQUISITOS Y PRUEBAS

Dado que el uso planteado para la aplicación es en el ámbito de una red local sin esperarse una carga de trabajo demasiado importante, los requisitos para el servidor de la aplicación son los requisitos del framework Symfony, definidos en el capítulo 4.

Los requisitos para los clientes de la aplicación son aún más relajados, cualquier ordenador actual con un cliente web o navegador de internet que cumpla los estándares debería ser capaz de usar la aplicación, independientemente del sistema operativo.

El desarrollo y pruebas se han realizado en un ordenador portátil con procesador AMD Sempron a 1,8 Ghz con 2GB de RAM, usando el navegador Firefox en su versión 12.0 sobre Windows Xp SP 3 y una resolución de escritorio de 1400x1050 pixels.

No obstante la aplicación se ha diseñado para que los datos se visualicen correctamente, aunque no con tanta comodidad, con una resolución mínima de 800x600, así que la aplicación podría ser usada incluso desde tabletas o teléfonos móviles con pantalla de gran tamaño.

7. CONSIDERACIONES ADICIONALES

7.a INCOMPATIBILIDADES

Los proyectos creados con Symfony 1.4.16 como se explica en el capítulo 2, no son accesibles en una instalación de Wamp 2.2d (la última versión existente en el momento de la realización de este proyecto) en Windows XP Service Pack 3 con todas las actualizaciones de seguridad instaladas y el conjunto mínimo de servicios necesarios para el funcionamiento del sistema iniciados. A pesar de no producirse ningún error en la instalación de Wamp ni en ninguno de los pasos de la creación del proyecto Symfony, al acceder a la URL del proyecto recién creado se produce un error en el proceso *httpd.exe* (el servidor web Apache) que a su vez invoca al proceso drwatson32.exe (el depurador del sistema).

Con estas mismas condiciones la página de inicio de Wamp funciona correctamente. Siendo en ambos casos archivos php y funcionando solo en uno de ellos parece que existe algún tipo de incompatibilidad entre Symfony y el servidor web Apache de Wamp 2.2d, al menos en el momento de realizar este proyecto.

Utilizando la versión de Wamp 2.2a no se produce esta incompatibilidad y se pueden acceder los proyectos Symfony sin que el servidor web Apache genere error alguno.

7.b VERSIÓN SANDBOX DE SYMFONY

El entorno de pruebas o *sandbox* es un proyecto vacío de Symfony que incluye todas las librerías y configuraciones necesarias para poder probar Symfony, muy útil para dar los primeros pasos con el framework desplegando en instantes un proyecto ya creado sin necesidad de usar la linea de comandos. Sin embargo en la versión sandbox no se pueden crear nuevos proyectos como se explica en el capitulo 4, de manera que para este fin se ha de usar la versión de Symfony normal.

7.c CLAVES DE LA BD

Los campos configurados en el archivo *schema.yml* con el parámero *primary: true* serán claves primarias de la base de datos. Si no se indica explicitamente los campos que deben ser claves primarias, Symfony crea como claves primarias campos *id* autoincrementales.

Antes de usar la metodología ORM era deseable usar como claves primarias datos únicos en el mundo real, por ejemplo DNIs, NIFs, matrículas y demás documentos de identificación. Sin embargo, los campos que son clave primaria por defecto no aparecen mostrados en los formularios de creación de datos nuevos ni en los formularios de modificación creados a través de la linea de comandos de Symfony, de manera que es aconsejable siempre que sea posible, dejar a Symfony crear las claves primarias para aprovechar la potencia de generar formularios con validaciones con un solo comando sin tener que implementar codificación adicional.

7.d CODIFICACIÓN DE LOS ARCHIVOS

Los archivos .yml usados para la carga de datos iniciales de la aplicación están codificados en ANSI, no permitiendo el uso de acentos. De introducir acentos en uno de estos archivos no se devuelve error alguno al realizar la carga pero a pesar de ello no se realiza la carga de los datos contenidos en ese archivo, comprometiendo la integridad de los datos de inicio.

De la misma manera, las plantillas de la aplicación de estar codificadas en *ANSI* y usar acentos o "ñ" en ella, mostrarán carácteres extraños en su lugar. Para visualizarlos correctamente se codificarán estos archivos en *UTF-8* en el caso de las plantillas locales. En el caso de los archivos de configuración de formularios la codificación usada será *UTF-8 without BOM* ya que de lo contrario se muestra una franja gris en la parte superior de las páginas de desarrollo.
Una manera sencilla de manejar las codificaciones de los archivos es con el editor de texto gratuito Notepad++.

7.e ENCRIPTACIÓN DE CONTRASEÑAS

Almacenar las contraseñas en la base de datos como texto legible es un riesgo enorme para la seguridad y puede ir contra la ley de protección de datos. Las contraseñas deben almacenarse encriptadas, por ejemplo guardando en vez de la contraseña el resultado de cifrarla usando el algoritmo *sha256*. De esta manera para validar las contraseñas al entrar en la aplicación, se validará si el resultado de aplicar *sha256* a la contraseña introducida por el usuario es el mismo valor almacenado en la base de datos.

Esto plantea un problema adicional. Al editar datos de usuarios sin modificar la contraseña manualmente, esta se recifraría guardándose como contraseña el resultado de aplicar *sha256* al resultado de aplicar *sha256* a la contraseña original. Para evitar esta problemática, al editar se cifran solo las contraseñas introducidas menores de 40 carácteres, pudiéndose crear contraseñas de hasta 255 carácteres al crear usuarios nuevos.

8. MEJORAS PROPUESTAS

Para trabajar en un servidor local en el ámbito académico, se pueden colocar los archivos de Symfony en la raíz del servidor web. Sin embargo, se trata de una mala práctica para servidores de producción en ámbito empresarial, ya que los usuarios podrían ver el funcionamiento interno de la aplicación. Se propone situarlos fuera del directorio raíz del servidor web, con permisos de acceso restrictivos si el sistema operativo usado lo permite.

De la misma manera que en el caso anterior, para servidores de producción al crear la base de datos es altamente recomendable crear una contraseña. Esto se lleva a cabo añadiendo el parámetro "-p" al crear la base de datos.

Aunque supondría un uso mayor de recursos, para quien no se sienta cómodo manejando ficheros de texto o para un modelo de tamaño mayor, puede ser una mejora interesante usar una herramienta gráfica para diseñar la BD de la aplicación como Fabforce DBDesigner 4. Otra posibilidad es usar Mysql Workbench para crear el modelo Entidad Relación y a partir de ese modelo generar el archivo *schema.yml* de forma automática con Mysql workbench schema exporter.

Dotar a la aplicación de nuevas funcionalidades:

- Permitir la realización de una práctica solo en un intervalo de tiempo a especificar.
- Permitir que un alumno repita una práctica.
- Mantener como profesor responsable de la práctica al profesor creador y realizar un registro de los profesores que han realizado modificaciones y las fechas en que se realizaron.
- Permitir que las respuestas de las cuestiones de tipo múltiple se puedan configurar para que se considere la cuestión respondida correctamente respondiendo correctamente solo algunas de las respuestas.
- Permitir configurar las cuestiones para que resten un porcentaje de su valor en caso de fallo.
- Permitir que las cuestiones cortas tengan respuestas textuales, en cuyo caso no se aplicaría el margen de error.
- Creación de prácticas con cuestiones al azar.
- Creación de prácticas con cuestiones al azar no usadas con anterioridad.
- Listar las prácticas de las que el profesor logueado es responsable.
- Introducir en el sistema el concepto de asignatura, de manera que una práctica deba estar relacionada con una única asignatura, y los alumnos solo puedan ver y resolver prácticas de las asignaturas en las que se encuentren matrículados.

9. DEFINICIONES

Apache	Servidor web de uso libre cuyas siglas significan A PACHEd server (Un servidor parcheado).
BD	Base de Datos.
Doctrine	Nombre del sistema ORM preinstalado en la versión de Symfony 1.4.16.
Gimp	The GNU Image Manipulation Program, programa de edición gráfica para manipular imágenes, de uso libre.
MySQL	Sistema de gestión de BD relacionales.
Notepad++	Procesador de texto de uso libre que permite colorear el texto según el estándard de multitud de lenguajes de programación, así como guardar los archivos con diferentes codificaciones, entre otras funciones.
OpenOffice	Suite ofimática de uso libre compatible con la suite Office.
ORM	Object Relational Mapper, Mapeador Relacional de Objetos. Modelo para interactuar con elementos de una BD como si fueran objetos.
PHP	PHP Hypertext Preprocessor, lenguaje de programación del lado del servidor. El resultado que genera su ejecución se sirve al navegador web del cliente.
SHA256	Secure Hash Algorithm, Algoritmo de hash seguro de 256 bits usado para cifrar datos
SQL	Structured Query Language, Lenguaje de consulta estructurado usado para interactuar con una BD.
StarUML	Programa de uso libre que permite crear modelos UML , casos de uso o diagramas de estado entre muchas otras funciones.
UML	Unified Modeling Language, Lenguaje de modelado unificado. Estándard usado para crear modelos de objetos.
WAMP	Apache, MySQL y PHP para Windows.

10. REFERENCIAS

- Félix Buendía García.
  Una guía para la realización y supervisión de proyectos final de carrera (PFC) en el ámbito de la web.
  Editorial Universidad Politécnica de Valencia.

- Página oficial del proyecto Symfony en español.
  - http://www.symfony.es/

- Página oficial del proyecto WAMP.
  - http://www.wampserver.com/en/

- Información sobre el entorno de pruebas de Symfony.
  - http://www.librosweb.es/symfony_1_2/capitulo3/instalando_el_entorno_de_pruebas.html

- Configuración de un servidor web para el entorno de pruebas de Symfony.
  - http://www.poetryofprogramming.com/symfony/setting-the-web-server-to-gain-access-to-symfony_datawebsf-directory/

- **Documentación de Symfony.**
  - http://www.symfony-project.org/

- Página oficial del proyecto Doctrine.
  - http://www.doctrine-project.org/

- **The symfony Reference Book.**
  http://www.symfony-project.org/reference/1_4/en/

- W3schools.
  - http://www.w3schools.com

- Star UML.
  - http://www.staruml.sourceforge.net

- Página oficial de Gimp.
  - http://www.gimp.org/

- Página oficial de OpenOffice.
  - http://www.openoffice.org/es/

- Página oficial de Notepad++.
  http://notepad-plus-plus.org

- Página oficial de Fabforce DBDesigner 4.
  - http://www.fabforce.net/dbdesigner4

- Mysql Workbench.
  - http://www.mysql.com/products/workbench/

- Mysql workbench schema exporter.
  - https://github.com/johmue/mysql-workbench-schema-exporter

11. ANEXOS

ANEXO I – Manual de usuario

Al acceder a la aplicación en http://symeval.localhost se muestra la página de inicio de sesión. Se proporciona en el presente documento un dato de cada tipo de usuario para poder realizar el primer acceso. Una vez iniciada sesión como profesor se puede consultar los datos del resto de alumnos desde la propia aplicación, y de la misma manera accediendo como administrador se pueden consultar los datos de los profesores:

Alumno:	71717171C	Profesor:	12345678Z	Administrador:	71717171X
Contraseña:	prueba	Contraseña:	prueba	Contraseña:	admin

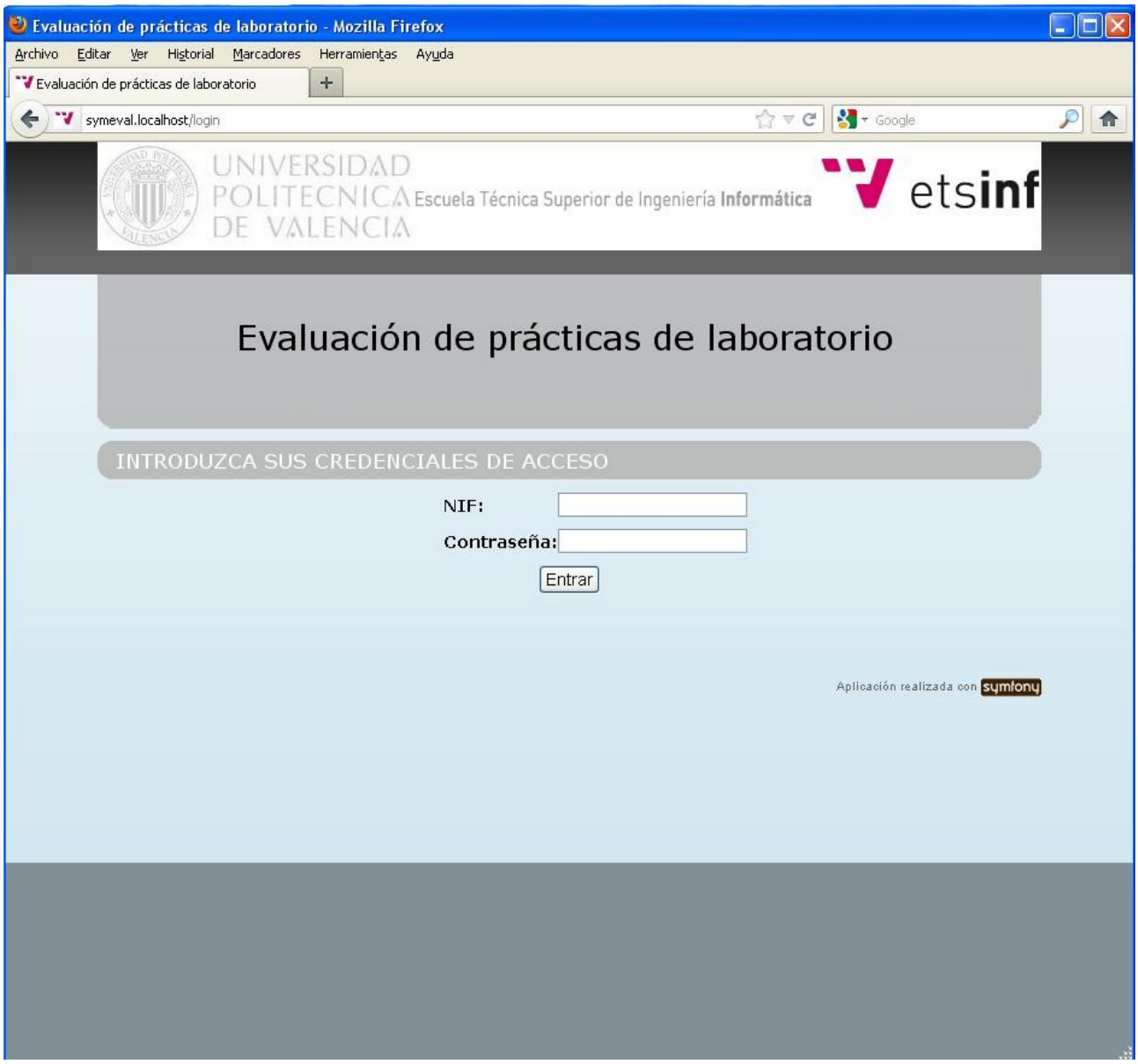

Ilustración 41: Página de inicio de sesión

Si el usuario que inicia sesión es un alumno, es redirigido a la página de prácticas, definida como página de inicio de los alumnos, donde dispone de la posibilidad de ver las prácticas existentes, consultar sus detalles sin ver sus cuestiones hasta que no decida resolverla, resolver las prácticas que no haya resuelto todavía, y de listar las notas que ha obtenido en las prácticas que ya ha resuelto, así como de cerrar la sesión.

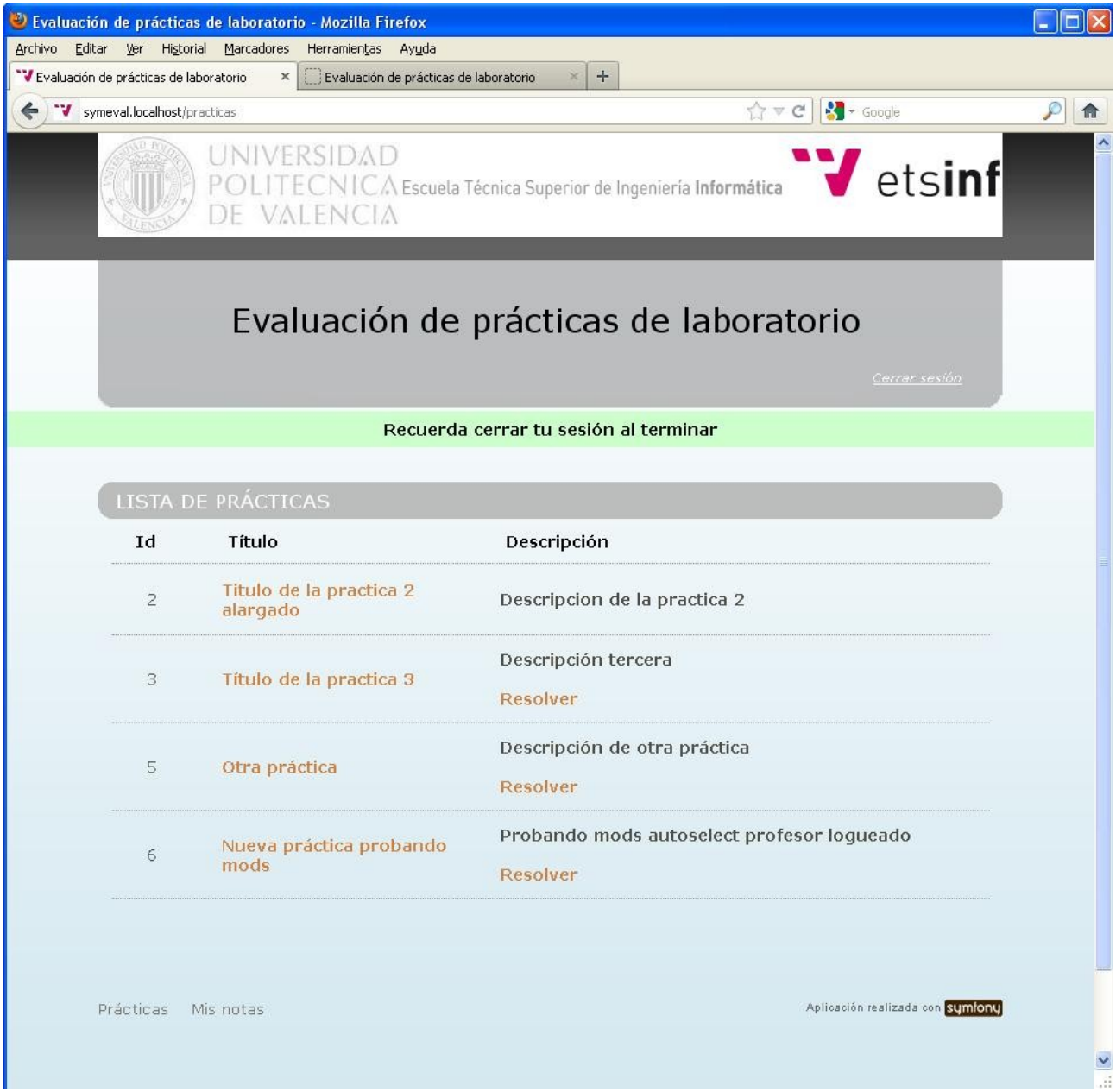

Ilustración 42: Página de inicio de los alumnos

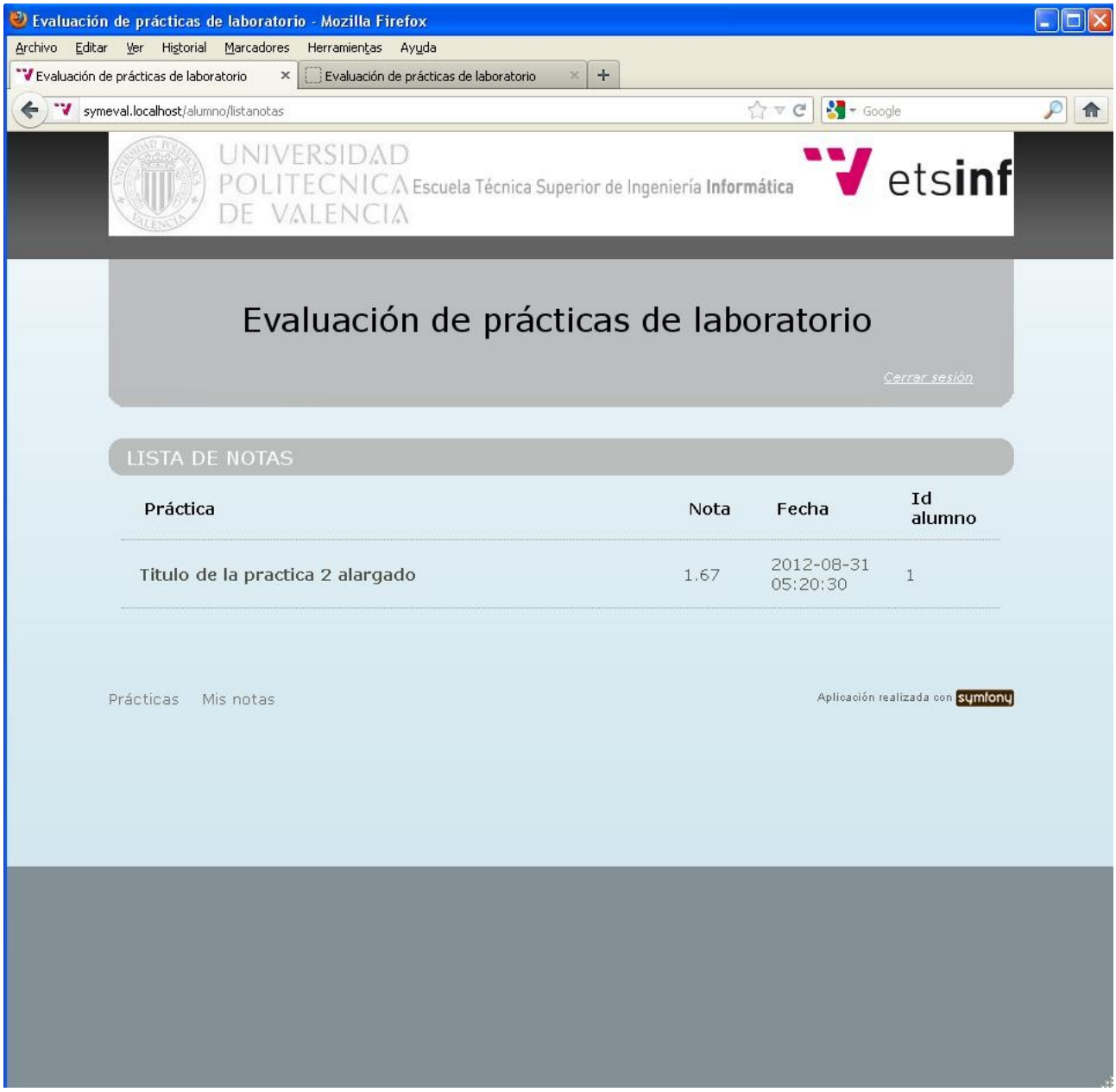

Ilustración 43: Listado de notas de alumno

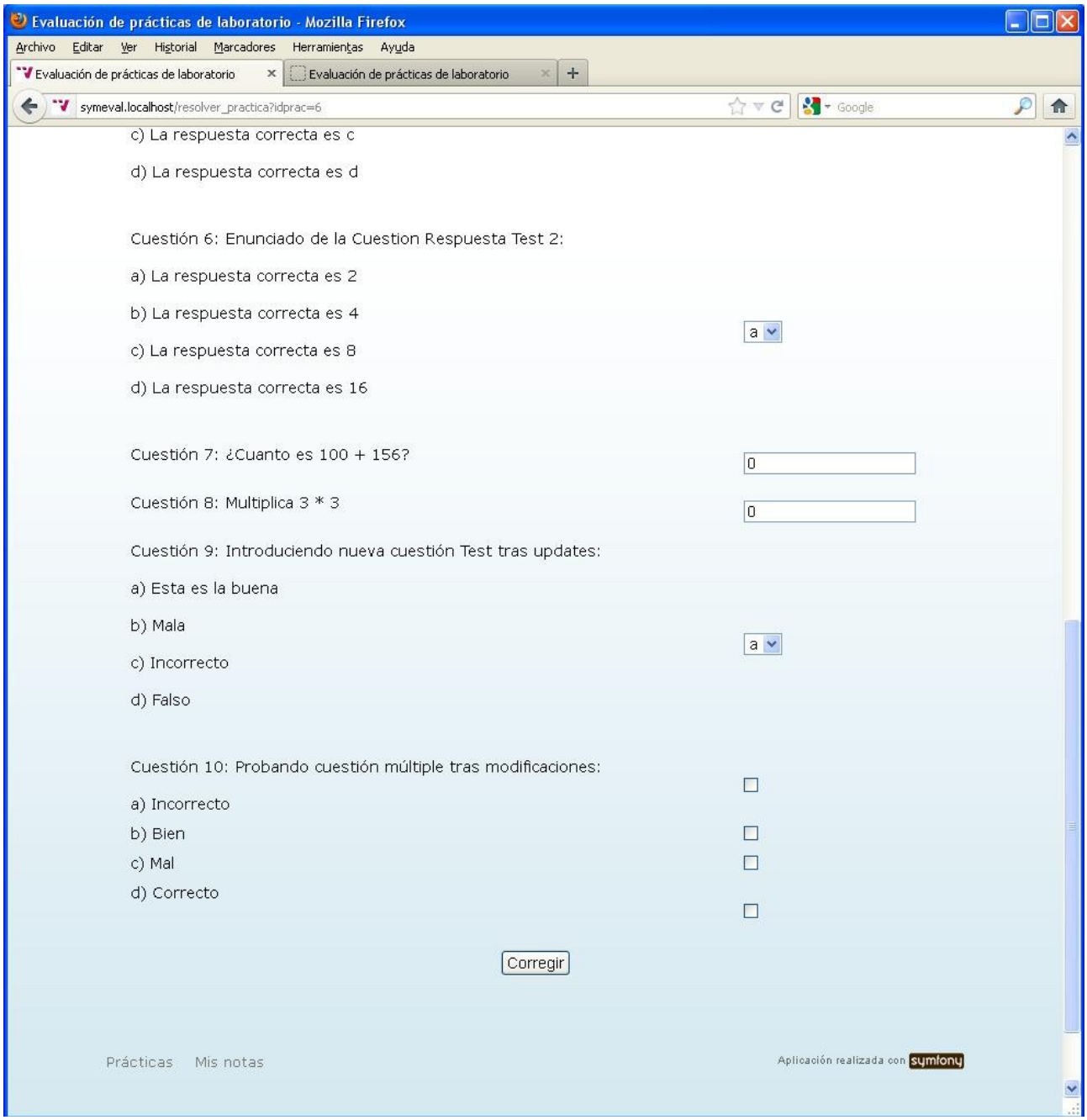

Ilustración 44: Página para resolver práctica

Si el usuario que inicia sesión es un profesor es redirigido a la página de gestión de prácticas, definida como página de inicio de los profesores, donde puede crear nuevas prácticas, ver los detalles de las prácticas existentes incluyendo sus cuestiones, las respuestas correctas de estas y las notas que han sacado los alumnos en esa práctica, así como navegar a las páginas de gestión de cuestiones y gestión de alumnos, donde puede consultar también las notas obtenidas por ese alumno. Puede también cerrar la sesión.

Ilustración 45: Página de inicio de los profesores

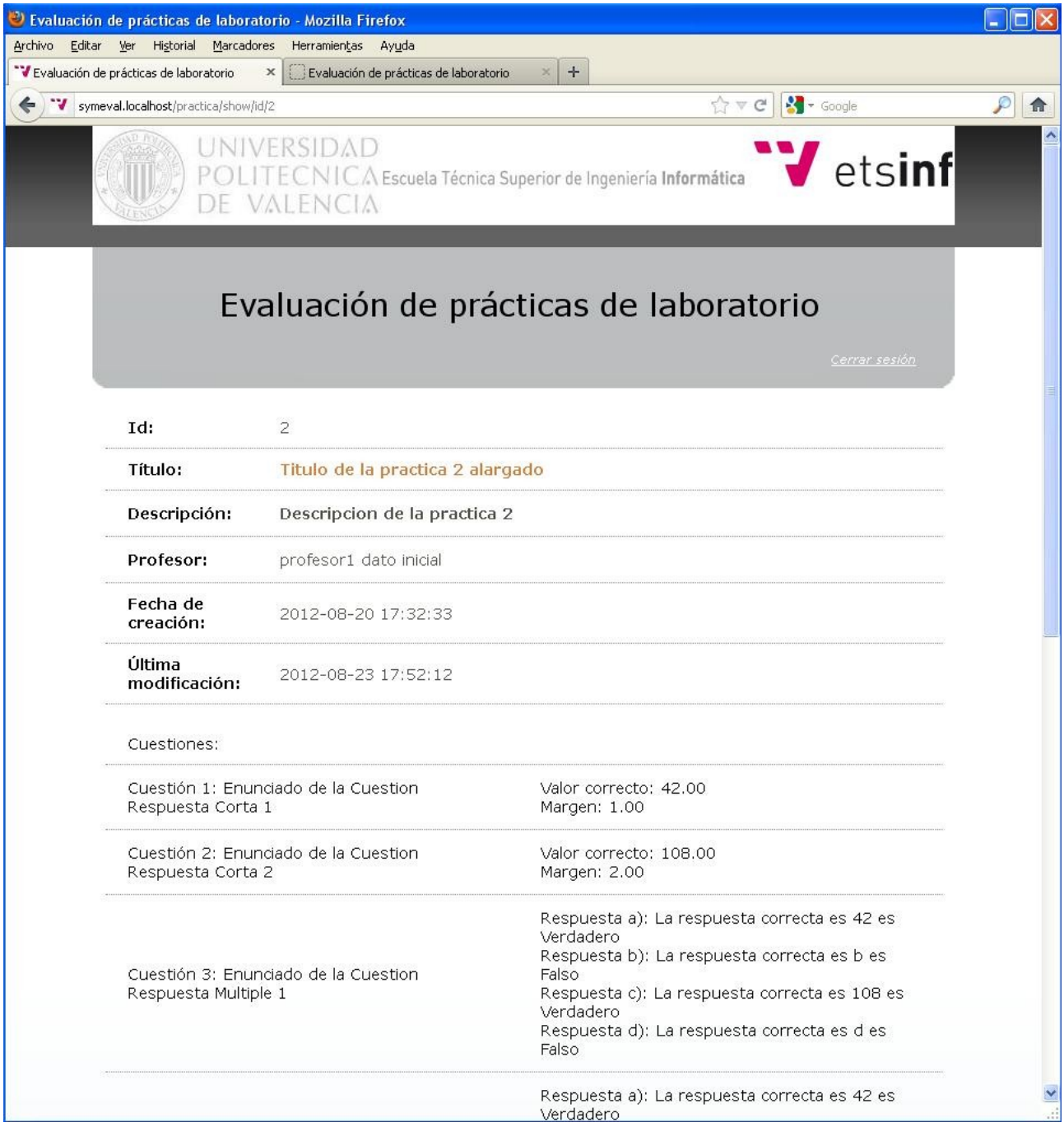

Ilustración 46: Página de práctica (Vista de profesores)

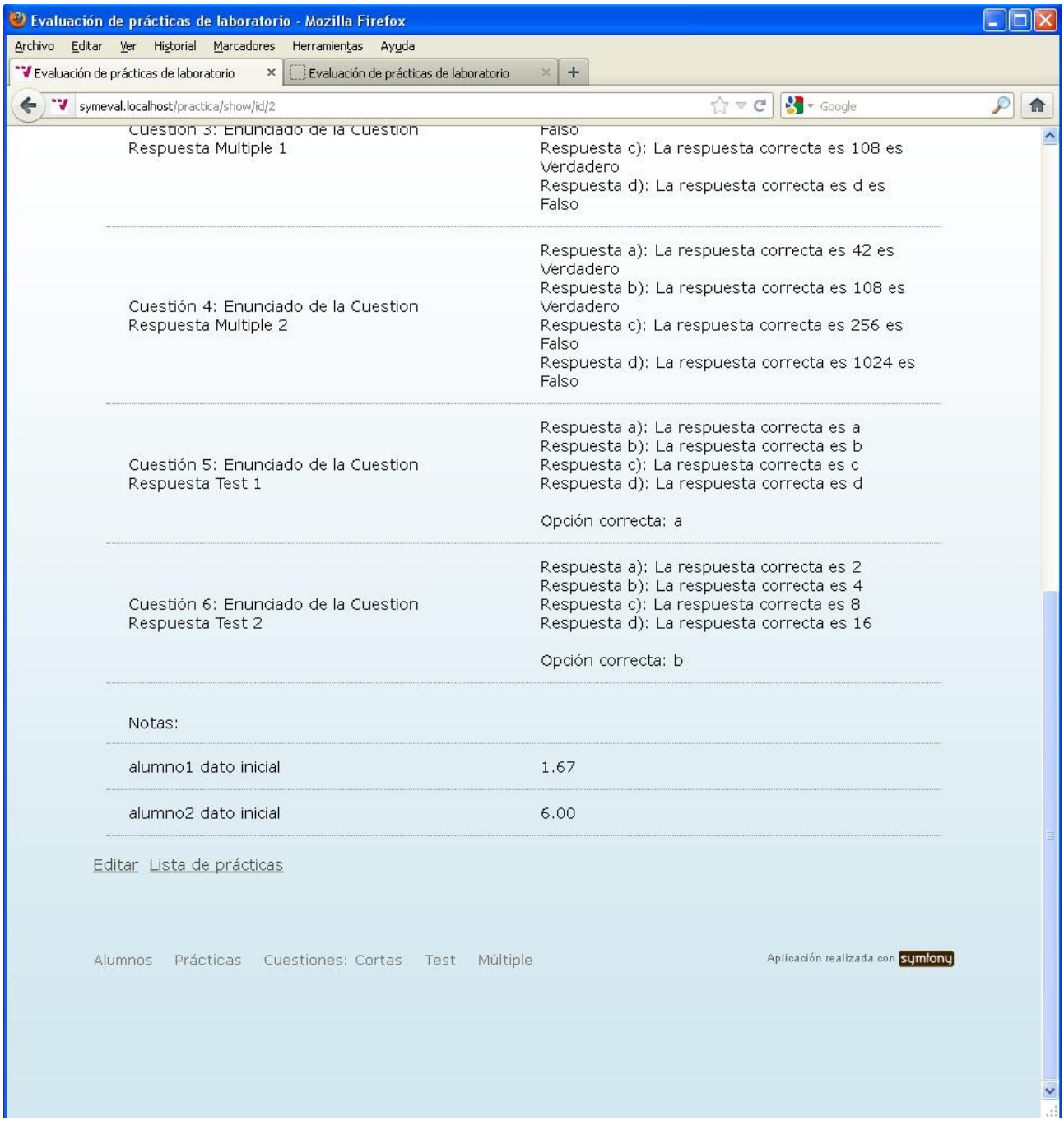

Ilustración 47: Notas de práctica (Vista de profesores)

Ilustración 48: Gestión de alumnos

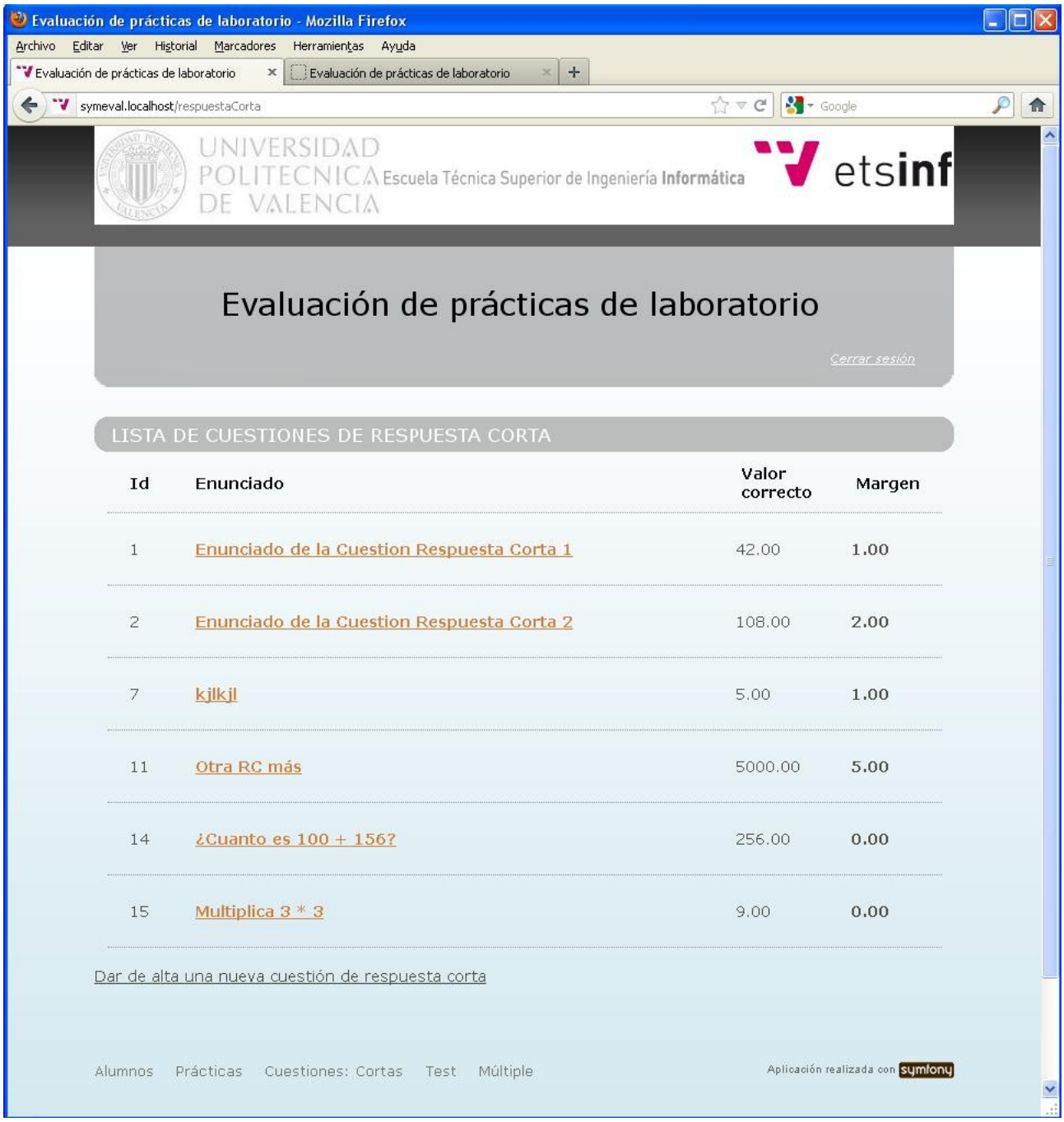

Ilustración 49: Gestión de cuestiones de respuesta corta

Si el usuario que inicia sesión es el administrador, es redirigido a la página de gestión de profesores, donde puede dar de alta nuevos profesores y modificar los datos de los existentes, así como realizar las mismas acciones que los demás profesores.

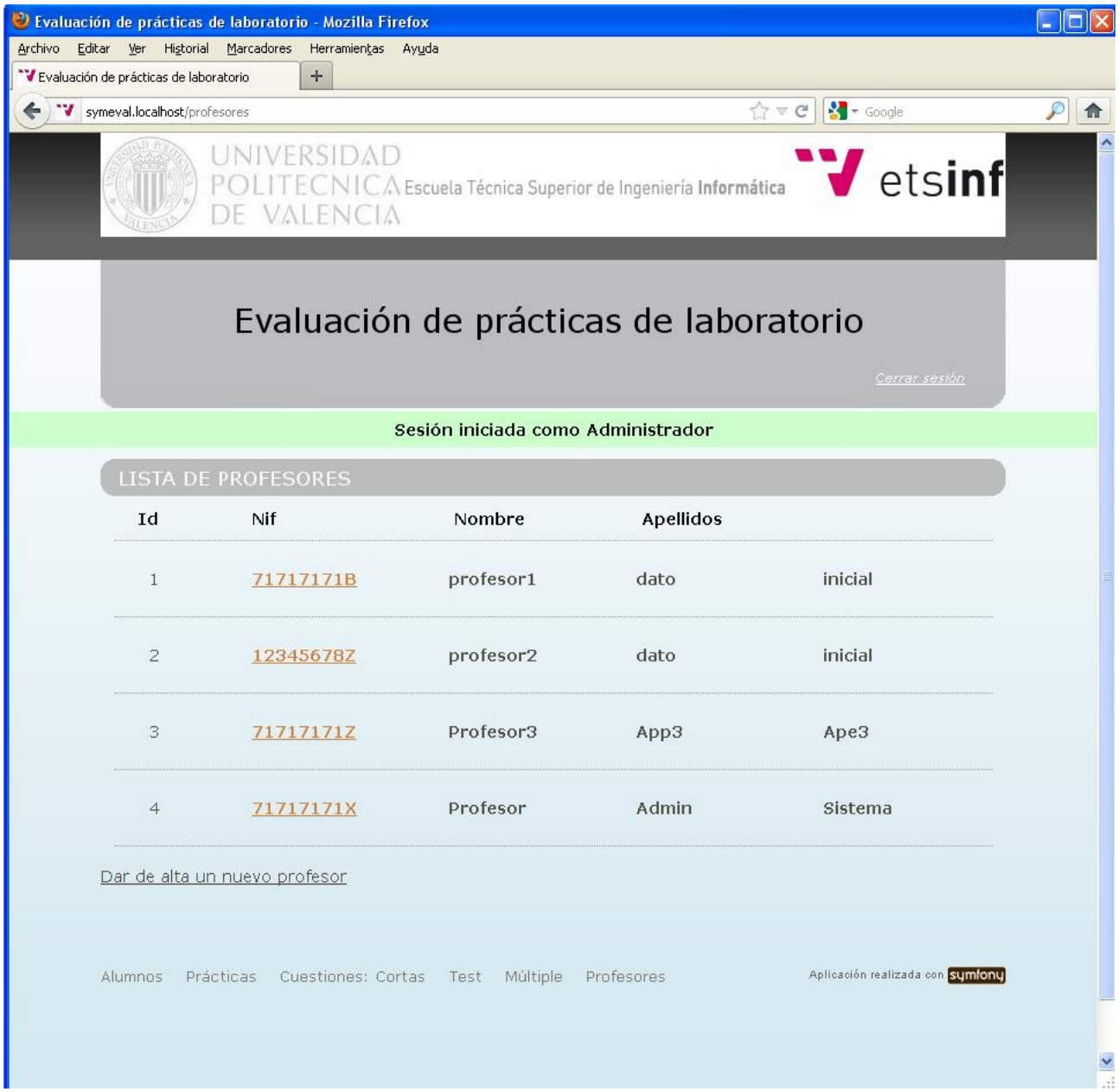

Ilustración 50: Gestión de profesores

Si el usuario que inicia sesión no existe en el sistema o la contraseña es incorrecta, es redirigido a la página de inicio de sesión con un mensaje de error que le ofrece ayuda para que reintente iniciar la sesión.

ANEXO II - Sentencias sql para crear la Base de Datos

Al usar Doctrine no es necesario escribir manualmente las sentencias SQL para crear la Base de Datos de la aplicación. Se crean en *C:\wamp\www\symeval\data\sql\schema.sql* usando el ORM Doctrine como se explica en el capítulo 5. Nótese que crearlas de esta manera con Symfony es mucho más eficaz que hacerlo a mano o con la interfaz de wamp. A modo ilustrativo se muestra parte del contenido del archivo *schema.sql*:

```
CREATE TABLE symeval_alumno (nif VARCHAR(9), contrasenya VARCHAR(255) NOT NULL, nombre VARCHAR(255) NOT NULL, apellido1 VARCHAR(255) NOT NULL, apellido2 VARCHAR(255) NOT NULL, PRIMARY KEY(nif)) ENGINE = INNODB;
CREATE TABLE symeval_cuestion (id BIGINT AUTO_INCREMENT, tipo_cuestion VARCHAR(1) NOT NULL, enunciado TEXT NOT NULL, nif_profesor VARCHAR(9) NOT NULL, type VARCHAR(255), valor_correcto DECIMAL(18, 2) NOT NULL, margen DECIMAL(18, 2) DEFAULT 0 NOT NULL, contenido_a VARCHAR(255) NOT NULL, contenido_b VARCHAR(255) NOT NULL, contenido_c VARCHAR(255) NOT NULL, contenido_d VARCHAR(255) NOT NULL, opc_correcta VARCHAR(1) NOT NULL, es_correcta_a TINYINT(1) NOT NULL, es_correcta_b TINYINT(1) NOT NULL, es_correcta_c TINYINT(1) NOT NULL, es_correcta_d TINYINT(1) NOT NULL, created_at DATETIME NOT NULL, updated_at DATETIME NOT NULL, INDEX symeval_cuestion_type_idx (type), INDEX nif_profesor_idx (nif_profesor), PRIMARY KEY(id)) ENGINE = INNODB;
CREATE TABLE symeval_cuestiones_practica (id_practica BIGINT, id_cuestion BIGINT, PRIMARY KEY(id_practica, id_cuestion)) ENGINE = INNODB;
CREATE TABLE symeval_nota (id_practica BIGINT, nif_alumno VARCHAR(9), nota DECIMAL(18, 2) NOT NULL, created_at DATETIME NOT NULL, updated_at DATETIME NOT NULL, PRIMARY KEY(id_practica, nif_alumno)) ENGINE = INNODB;
CREATE TABLE symeval_practica (id BIGINT AUTO_INCREMENT, titulo VARCHAR(255) NOT NULL, descripcion TEXT, nif_profesor VARCHAR(9) NOT NULL, created_at DATETIME NOT NULL, updated_at DATETIME NOT NULL, INDEX nif_profesor_idx (nif_profesor), PRIMARY KEY(id)) ENGINE = INNODB;
CREATE TABLE symeval_profesor (nif VARCHAR(9), contrasenya VARCHAR(255) NOT NULL, nombre VARCHAR(255) NOT NULL, apellido1 VARCHAR(255) NOT NULL, apellido2 VARCHAR(255) NOT NULL, PRIMARY KEY(nif)) ENGINE = INNODB;
....
```

ANEXO III - Archivo schema.yml de definición de la BD

En el capítulo 5 se explica como crear la BD de la aplicación a partir del archivo de definición del esquema. A continuación se muestra parte del archivo:

```
---
symevalAlumno:
  columns:
    nif:         { type: string(9), notnull: true, unique: true }
    contrasenya: { type: string(255), notnull: true }
    nombre:      { type: string(255), notnull: true }
    apellido1:   { type: string(255), notnull: true }
    apellido2:   { type: string(255), notnull: true }

symevalProfesor:
  columns:
    nif:         { type: string(9), notnull: true, unique: true }
    contrasenya: { type: string(255), notnull: true }
    nombre:      { type: string(255), notnull: true }
    apellido1:   { type: string(255), notnull: true }
    apellido2:   { type: string(255), notnull: true }

symevalPractica:
  actAs: { Timestampable: ~ }
  columns:
    profesor_id: { type: integer, notnull: true }
    titulo:      { type: string(255), notnull: true }
    descripcion: { type: string(4000) }
  relations:
    symevalProfesor: { onDelete: CASCADE, local: profesor_id, foreign: id, foreignAlias:
symevalProfesorPracticas }
...
```

ANEXO IV - Archivos *.yml* para introducir los datos iniciales en la aplicación

Gracias al uso de Doctrine crear los datos de inicio de la aplicación no requiere desarrollar código sql ni introducirlos manualmente a través de la interfaz de wamp. Se crean con archivos de texto plano situados en el directorio *data/fixtures/* del proyecto y la tarea de Doctrine *data-load* como se indica en el capítulo 5. Se muestran dos de ellos como ejemplo:

```
# data/fixtures/alumnos.yml
symevalAlumno:
  Alumno1:
    nif: 71717171C
    contrasenya: prueba
    nombre: alumno1
    apellido1: dato
    apellido2: inicial

  Alumno2:
    nif: 12345678Z
    contrasenya: prueba
    nombre: alumno2
    apellido1: dato
    apellido2: inicial

# data/fixtures/profesores.yml
symevalProfesor:
  Profesor1:
    nif: 71717171C
    contrasenya: prueba
    nombre: profesor1
    apellido1: dato
    apellido2: inicial

  Profesor2:
    nif: 12345678Z
    contrasenya: prueba
    nombre: profesor2
    apellido1: dato
    apellido2: inicial
```