Práctica 2

Planificación de cursos

Aplicaciones Web
Año 2016/2017
Grado en Ing. Software
Fdi - UCM

Fecha de entrega: 3 de febrero de 2017.

Porcentaje en la calificación de prácticas: 50 % (1.50 puntos en la calificación final de la asignatura).

Objetivos: La práctica tiene como objetivo la evaluación de las siguientes competencias:

- Implementación de APIs de tipo REST (Tema 8).
- Implementación de aplicaciones web utilizando AJAX y el modelo SPA (Temas 7 y 8).

Un aplicación SPA (*Single Page Application*) es una aplicación web contenida en un **único** documento HTML. Los componentes de este documento se muestran u ocultan a medida que el usuario interactúa con la aplicación web.

Evaluación: Esta práctica contiene cuatro apartados, de los cuales el primero (*Funcionalidad básica*) es obligatorio. Entregando solamente este apartado, la calificación máxima obtenible es de APTO con 4/10. Si se desea obtener una calificación mayor, se deberán realizar las extensiones opcionales de la práctica. La calificación máxima variará en función del número de extensiones implementadas.

- Apartado 1: Calificación máxima de 4/10 (0.60 puntos en la nota final de la asignatura).
- Apartados 1 y 2: Calificación máxima de 5/10 (7.50 puntos en la nota final).
- Apartados 1, 2 y 3: Calificación máxima de 8/10 (1.20 puntos en la nota final).
- Apartados 1, 2, 3 y 4: Calificación máxima de 10/10 (1.50 puntos en la nota final).

Se recuerda que la ausencia de entregas o la entrega de una práctica total o parcialmente copiada conllevará la calificación de NO APTO no recuperable. Los casos de copias serán comunicados al Comité de Actuación ante Copias de la FdI.

En la corrección se valorará:

- El correcto funcionamiento de la aplicación web.
- El correcto diseño de la API RESTful y su implementación en *Express.js*. El servidor implementado ha de funcionar a través del protocolo **HTTPS**.

- El diseño de la página web. No es necesario que sea igual a las capturas de pantalla que se muestran en las figuras de este enunciado. Para facilitar el trabajo, podéis utilizar *Bootstrap* o tecnologías similares del lado del cliente, aunque el uso de estas tecnologías no es obligatorio.

- El correcto diseño de la base de datos relacional. Aconsejo leer el enunciado en su totalidad antes de abordar el desarrollo, aunque no se tenga intención de realizar las extensiones opcionales. Esto os ahorrará cambios en la BD a mitad de desarrollo.

En esta práctica implementaremos una aplicación web muy básica para gestionar cursos presenciales de diversa índole: programación, cocina, baile, música, etc. El servidor contendrá una base de datos con cursos disponibles. Los usuarios de la aplicación pueden buscar un determinado curso a partir de su nombre, y registrarse en uno o varios cursos.

De este modo, la base de datos manejará dos tipos diferentes de entidad: cursos y usuarios:

- Para cada curso se almacena un título, una descripción (que es un texto en formato HTML), la localidad en la que se impartirá el curso, la dirección de impartición, y el número de plazas disponibles. También se guardarán la fecha de inicio del curso, su fecha de finalización, y la lista de horarios de impartición del curso, que es un atributo multivalorado. Esto implica que los horarios de cada curso se guardan en una tabla aparte que contiene varias entradas, cada una con un día de la semana, una hora de inicio y una hora de finalización. Por último, un curso puede tener una imagen asociada, aunque dicha imagen es opcional.

- Para cada usuario se almacena una dirección de correo que lo identifica, una contraseña, su nombre, sus apellidos, su sexo (hombre o mujer) y su fecha de nacimiento. Un usuario puede estar inscrito en ninguno, uno o varios cursos.

La aplicación web debe ser desarrollada según el modelo de aplicación de página única (*Single Page Application*). Para ello, la funcionalidad del lado del servidor debe ser implementada mediante una API de tipo REST. Para la programación del lado del cliente debe utilizarse *jQuery* como librería de apoyo. Las peticiones del cliente al servidor se realizarán mediante AJAX, utilizando JSON como lenguaje de intercambio de datos. Dado que algunas de las funcionalidades del servidor requieren autenticación, el servidor ha de funcionar bajo el protocolo HTTPS.

La siguiente sección describe la parte obligatoria de la práctica, que consiste esencialmente en la implementación de una API REST para la gestión de los cursos, y la búsqueda de cursos por nombre en el cliente. Las partes opcionales de la práctica abordan la subida de imágenes, la creación de cuentas de usuario y la inscripción en cursos.

1 Funcionalidad básica.

En primer lugar, implementa un servidor utilizando *Node.js* y *Express.js* que proporcione un API REST con las siguientes funciones:

- **Inserción de cursos en la base de datos.** Es una petición de tipo POST que recibe los datos del curso a insertar, exceptuando la imagen. En caso de éxito, el servidor debe responder con el identificador del curso insertado.

- **Actualización de cursos en la base de datos.** En este caso se trata de una petición de tipo PUT, en la que la URL indica el identificador del curso que se desea modificar, y el cuerpo de la petición incluye la información actualizada del curso. El código de respuesta del servidor indicará si la operación se ha realizado con éxito o no.

- **Eliminación de cursos en la base de datos.** Se trata de una petición de tipo DELETE, cuya URL contiene el identificador del curso a eliminar. El servidor confirmará si la operación ha tenido éxito o no.

- **Obtención de la información de un curso.** Mediante una petición de tipo GET, y a partir del identificador de un curso, el servidor responderá con la información obtenida de la base de datos: título del curso, descripción, localidad, lugar, fecha de inicio, fecha de finalización, horarios y número de plazas disponibles.

- **Búsqueda por nombre de cursos.** También se trata de una petición de tipo GET. La URL contendrá en su *query string*:

  - Una cadena de texto (`str`).
  - Un número máximo de resultados a devolver (`num`).
  - La posición del primer resultado devuelto (`pos`).

  El servidor consultará en la BD aquellos cursos cuyo nombre contenga la cadena `str`, en orden creciente de fecha de inicio. Sólo se debe recuperar de la BD un determinado rango de cursos de entre todos aquellos que cumplan el criterio mencionado anteriormente. El parámetro `num` determina cuántos cursos recuperar de la BD, y el parámetro `pos` indica a partir de qué curso se comienza a contar. Por ejemplo, supongamos `str = 'cocina'`, `num = 5`, `pos = 3`, y que hay 20 entradas en la BD que contienen la subcadena `'cocina'`. Si ordenamos las 20 entradas por fecha, la BD debe devolver 5 resultados, comenzando a contar desde la fila 3. Es decir, debe devolver las filas 3, 4, 5, 6 y 7. Consulta la cláusula `LIMIT` en la documentación de MySQL para saber cómo obtener un subconjunto de filas dentro de una consulta. Para cada curso se devolverá su nombre, la localidad de impartición y las fechas de inicio y fin.

Una vez realizada la implementación del *backend* de esta parte de la práctica pasamos a abordar la implementación del cliente. En este caso tan solo se trata de implementar la funcionalidad de búsqueda por nombre (Figura 1). La interfaz contendrá un cuadro de texto y un botón de búsqueda. Cuando se pulsa en el botón de búsqueda se mostrará una tabla que contenga aquellos cursos cuyo nombre contenga la cadena dada. Se mostrarán un máximo de cinco cursos. Si existen más de cinco cursos que ajusten con la cadena, se segmentará el resultado mediante un control de paginación como el mostrado en la Figura 1 (abajo). Utiliza, para ello, los parámetros `num` y `pos` mencionados anteriormente.

De momento no te preocupes por el botón `[Identificarse]` de la esquina superior derecha ni por la columna `Vacantes` de la tabla, ni por los colores de alguna de las filas de la Figura 1. Todo esto forma parte de las extensiones opcionales de la práctica.

Al hacer clic en alguna de las filas de la tabla que contiene los resultados, debe mostrarse la información sobre el curso mostrado en esa fila (Figura 2). Se ha de mostrar la información devuelta por la petición GET descrita anteriormente (*Obtención de la información de un curso.*) De momento puedes ignorar la imagen que aparece al lado derecho, el número de vacantes y el botón de `[Inscribirse]`.

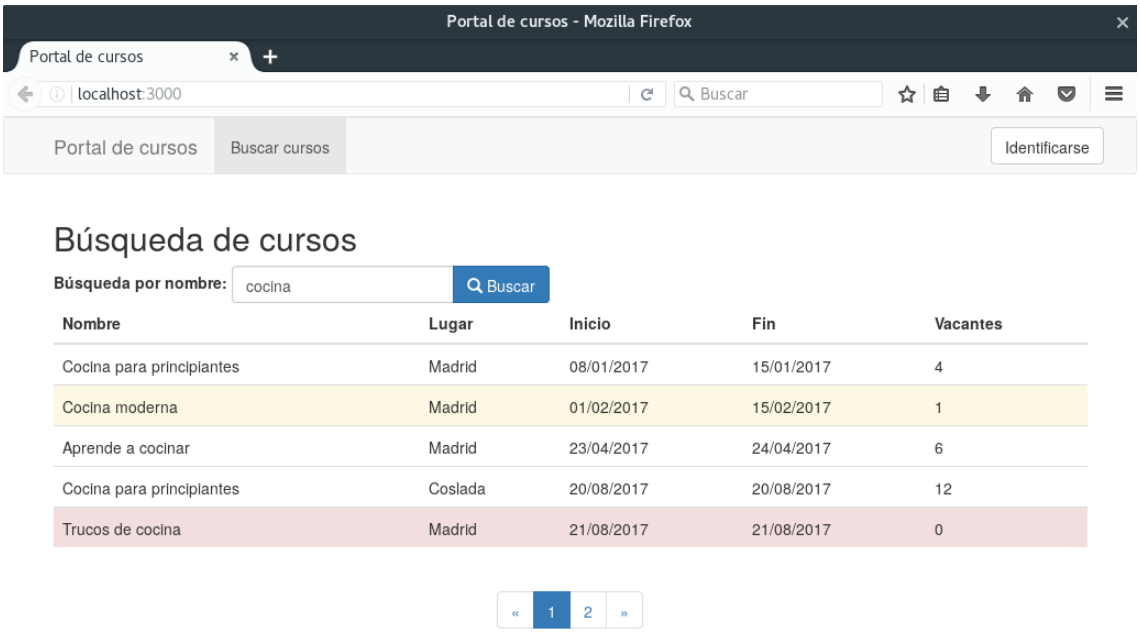

Figura 1: Búsqueda de cursos por nombre

En esta práctica no es necesario implementar funcionalidad en el lado del cliente correspondiente al resto de operaciones de modificación de cursos (inserción, actualización o borrado). Si quieres comprobar el funcionamiento de estas operaciones puedes hacerlo manualmente mediante uno de los clientes REST vistos en clase.

2 Gestión de imágenes

La primera extensión opcional de la práctica consiste en la implementación de la funcionalidad relativa a las imágenes de cada curso. En el lado del servidor se han de implementar dos nuevas operaciones:

- **Subida de una imagen a un curso ya existente**. Se trata una petición de tipo PUT, cuya URL contiene el identificador del curso en el que se quiere adjuntar una imagen. Si el curso ya tenía una imagen, será reemplazada por la nueva. El cuerpo de la petición debe contener la imagen. La respuesta del servidor indica si la operación tuvo éxito.
- **Obtención de la imagen de un curso**. En este caso es una petición de tipo GET. En la URL se indica el curso que contiene la imagen a recibir. La respuesta del servidor debe devolver, en caso de éxito, la imagen de dicho curso. En caso de error debe devolver un error 404 (*Not Found*).

Tras realizar esto, modifica la página del cliente para que en la vista de descripción de un curso (Figura

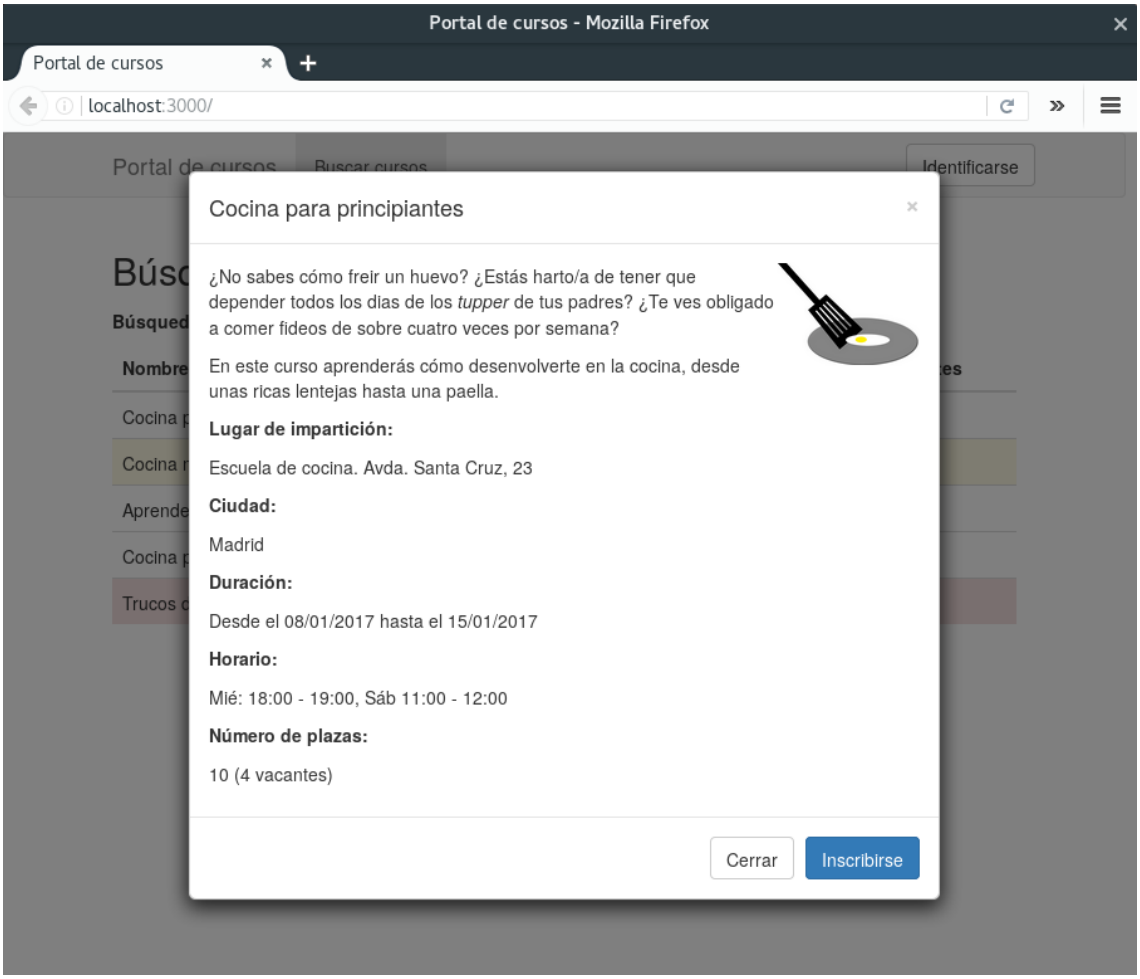

Figura 2: Información de un curso

2) se muestre la imagen correspondiente, si existe.

3 Gestión de usuarios

En esta extensión de la práctica se desarrollará la funcionalidad relativa a la gestión de usuarios y a la inscripción de usuarios en los cursos. En el lado del servidor debe extenderse la API con las operaciones que se enumeran a continuación. Salvo la primera, todas ellas han de ir autenticadas. Para ello utiliza el esquema de identificación básico visto en clase.

- **Añadir un usuario**. Mediante una petición POST se indicarán los datos del usuario a añadir: dirección de correo, contraseña, nombre, apellidos, sexo y fecha de nacimiento. Se devolverá un código u otro en función de si la operación tuvo éxito o no.
- **Comprobar la correcta identificación de un usuario.** Se trata de una sencilla petición GET autenticada, en la que el servidor comprobará si el nombre de usuario y contraseña del cliente son válidos. En caso afirmativo, se devolverá un código 200 (*OK*), mientras que en caso contrario se

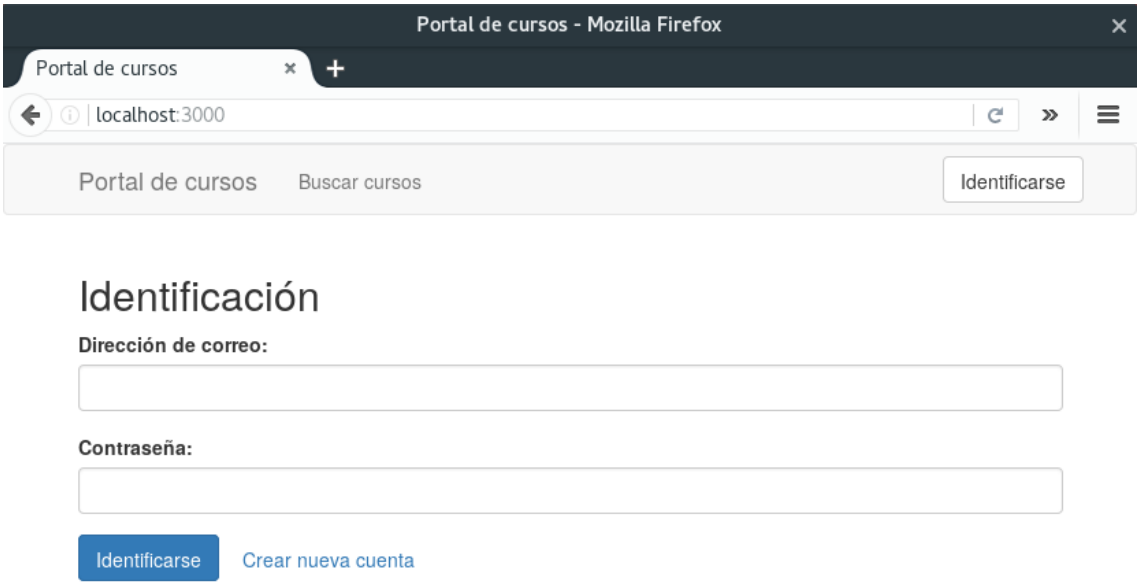

Figura 3: Identificación en el sistema

devolverá un código 401 (*Unauthorized*).

- **Inscripción en un curso**, mediante una petición POST. El cuerpo de la petición debe contener el identificador del curso en el que el usuario actualmente identificado se desea inscribir. El código de respuesta devuelto dependerá de si la operación se ha realizado correctamente o no.
- **Obtención de los cursos** en los que el usuario actualmente identificado está inscrito. Se trata de una petición de tipo GET. Para cada curso se devolverá su localidad de impartición y sus respectivas fechas de inicio y finalización.

Con respecto al cliente, se debe implementar lo siguiente:

- **Formulario de identificación de usuario**, como el mostrado en la Figura 3. Cuando el usuario pulse el botón `[Identificarse]` se comprobará que el usuario esté en la base de datos, mediante la correspondiente petición al servidor. Si la identificación tiene éxito, todas las peticiones que se realicen a partir de ese momento serán autenticadas.
- **Mostrar usuario actualmente conectado y cerrar sesión**. Se deberá mostrar en algún lugar de la página si existe un usuario actualmente conectado, junto con un botón para cerrar la sesión. Por ejemplo, en la Figura 4 se muestra en la esquina superior derecha.
- **Dar de alta un usuario**. Si en el panel de identificación de la Figura 3 se hace clic en el enlace `[Crear nueva cuenta]`, aparecerá el panel de la Figura 5, donde el usuario podrá introducir sus datos y añadirlos a la BD.

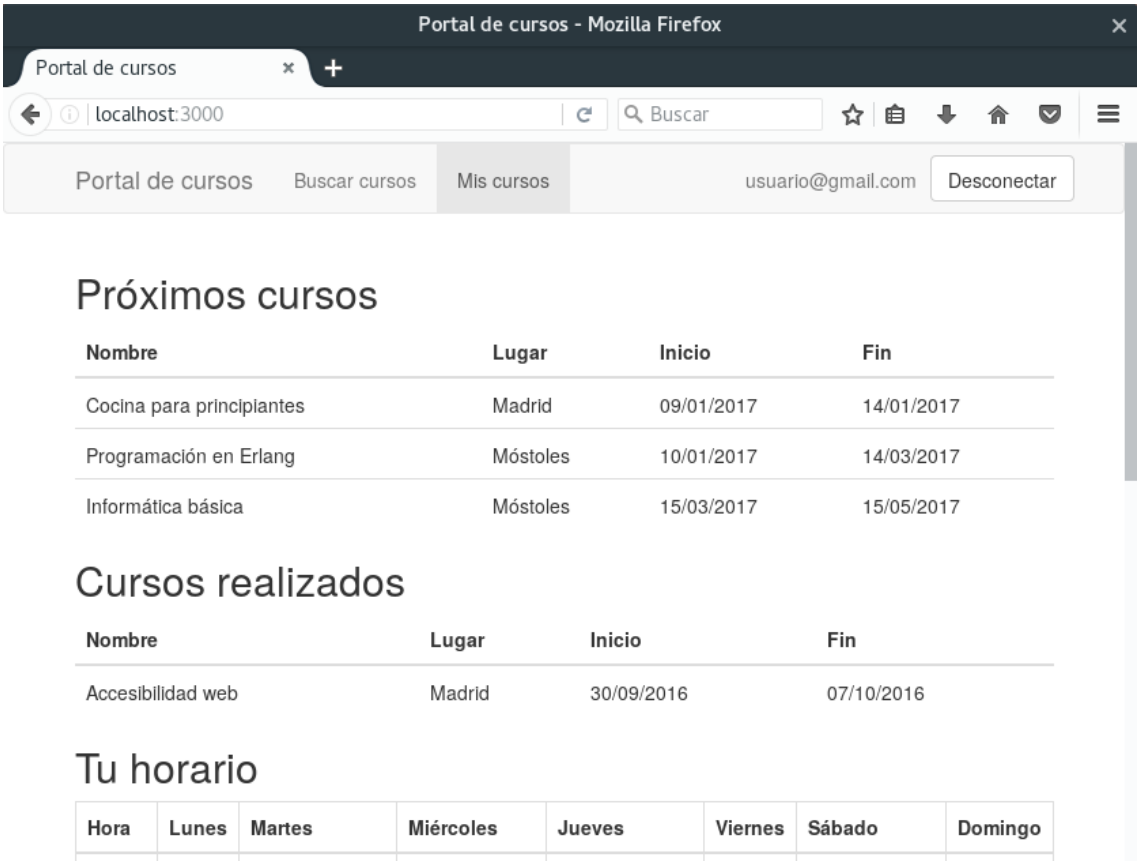

Figura 4: Vista de cursos de un usuario

- **Inscripción de un usuario en un curso**. Cuando se muestra la vista de un curso (Figura 2), si el usuario está autenticado, aparecerá un botón `[Inscribirse]` que permitirá al usuario apuntarse en el curso mostrado actualmente.
- **Mostrar cursos futuros y pasados**. Al hacer clic en el enlace `[Mis cursos]` (barra superior de la Figura 4) se mostrará un panel con los cursos en los que está inscrito un usuario. En la tabla `Próximos cursos` aparecerán los cursos cuya fecha de inicio es posterior a la fecha actual. En la tabla `Cursos realizados` aparecerán los restantes (incluyendo aquellos que aún no han finalizado). De momento ignora la sección `[Tu horario]` que aparece en la parte inferior de la Figura 4.
- **Vacantes**. Modifica la información mostrada en las Figuras 1 y 2 para que muestre el número de plazas libres (vacantes) en el curso. Además, en la tabla de búsqueda de cursos se mostrarán con fondo amarillo los cursos que solo tengan una vacante, y en rojo los que no tengan vacantes.

Algunas de las funcionalidades del cliente de esta parte de la práctica pueden requerir cambios en algunas de las funciones de la API del servidor realizadas en partes anteriores.

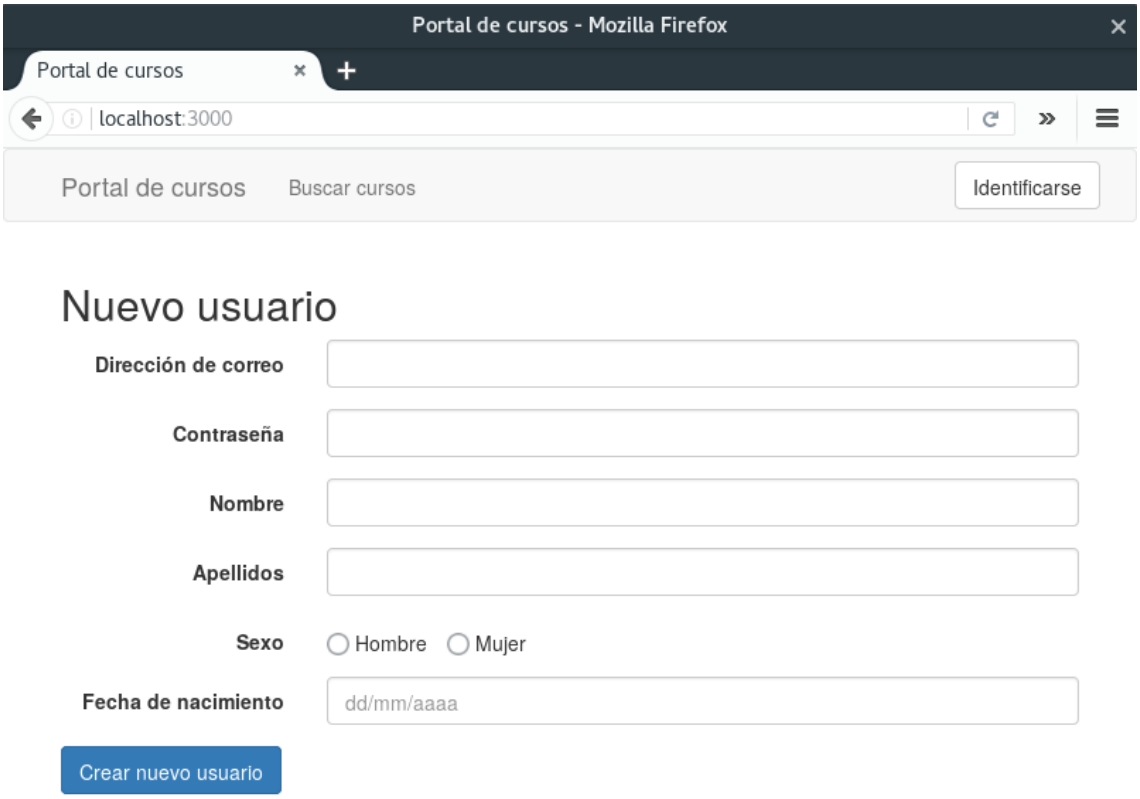

Figura 5: Creación de un nuevo usuario

4 Tabla de horarios

En la parte inferior de la vista de los cursos de un usuario aparece una sección titulada `Tu horario` (ver Figura 6). En esta sección se muestra una tabla con la planificación de una determinada semana, teniendo en cuenta los cursos en los que el usuario actual está inscrito. Esta tabla tiene siempre ocho columnas. El número de filas dependerá de los cursos inscritos que transcurran en la semana que se está visualizando. Para ello ten en cuenta las horas de inicio y finalización de cada curso de esta semana. Cada columna de la tabla ha de abarcar el rango desde la medianoche del inicio del día correspondiente (00:00h) hasta la madrugada del día siguiente (24:00h). Por ejemplo, supongamos que en una semana tenemos los siguientes cursos:

- Curso A: martes 12-14h y viernes 12-14h.
- Curso B: lunes 12:15-13h y jueves 12:15-13h.
- Curso C: viernes 18-19h.

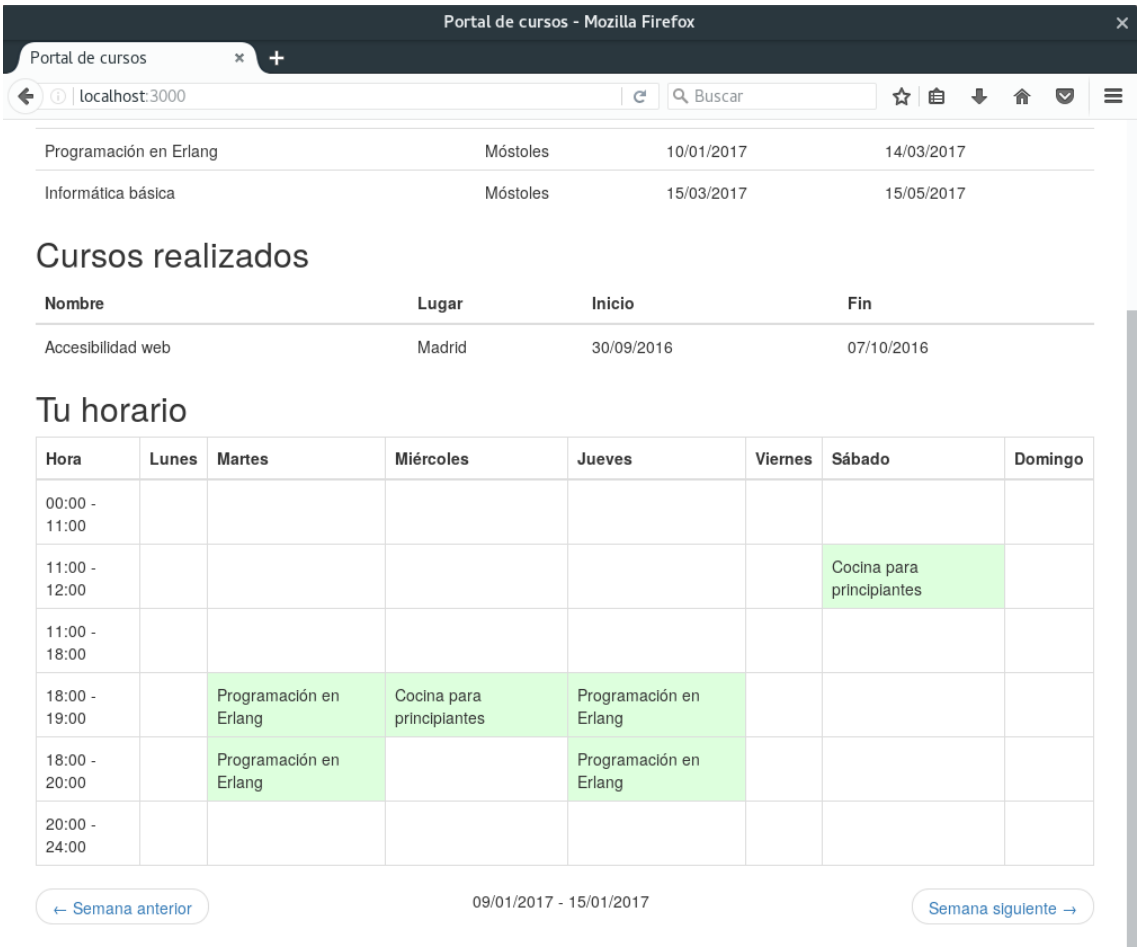

Figura 6: Tabla de horarios

Las filas a considerar serán las siguientes: 00-12h, 12-12:15h, 12:15-13h, 13-14h, 14-18h, 18-19h, 19-24h. En caso de solapamiento de cursos, se mostrarán todos ellos en la celda de la tabla correspondiente.
En la parte inferior de la tabla deben aparecer dos botones que permiten cambiar la semana visualizada.