Gestión de la Información en la Web
Curso 2024–2025
Práctica 12 – Aplicaciones web con Django

Fecha de entrega: domingo 12 de enero de 2025, 23:55h

Entrega de la práctica
La entrega de la práctica se realizará a través del Campus Virtual de la asignatura mediante un fichero **django_XX.zip** donde **XX** es el número de grupo.

Versiones
Python 3.11 o superior y **Django 5.x.y**.

Calificación
Se tendrá en cuenta que la aplicación web sea correcta y se haga un uso adecuado de las capacidades de Django. También se tendrá en cuenta la calidad del código, su legibilidad y la evaluación de Pylint.

Declaración de autoría e integridad
El fichero ZIP entregado contendrá un archivo **AUTORES.TXT** donde se indique la asignatura, la práctica, el grupo y los autores. Este archivo también contendrá la siguiente declaración de integridad:

> *Declaramos que esta solución es fruto exclusivamente de nuestro trabajo personal. No hemos sido ayudados por ninguna otra persona o sistema automático ni hemos obtenido la solución de fuentes externas, y tampoco hemos compartido nuestra solución con otras personas de manera directa o indirecta. Declaramos además que no hemos realizado de manera deshonesta ninguna otra actividad que pueda mejorar nuestros resultados ni perjudicar los resultados de los demás.*

No se corregirá ninguna entrega que no venga acompañada de dicho fichero.

Aplicaciones web con Django

En esta práctica vamos a realizar una aplicación web completa usando el *framework* Django. Se tratará de una aplicación web muy simple donde los usuarios podrán escribir preguntas y también respuestas a las preguntas a un único nivel, es decir, se podrá contestar preguntas pero no se podrá contestar a una contestación. **Podéis ver un ejemplo de cómo debe funcionar la aplicación web en el vídeo disponible en el Campus Virtual.**

Para desarrollar esta apliación web se usarán las siguientes facilidades de Django:

- **Gestión de usuarios**: los usuarios sin autenticar únicamente podrán visualizar el listado con los títulos de las preguntas, su autor, fecha y número de contestaciones. Los usuarios autenticados podrán también acceder al detalle de la pregunta (texto y sus contestaciones), además de enviar preguntas y respuestas. La aplicación permitirá autenticar usuarios y cerrar su sesión, pero no es necesario que permita la creación/modificación de usuarios (esto se hará desde la interfaz de administración).
- **ORM** para definir las clases que queremos almacenar en la base de datos **SQLite**. También se usará el ORM para realizar búsquedas en la BD.
- **Validación de entradas** del usuario mediante el uso de **formularios**, teniendo cuidado de escapar el posible código HTML que se almacene en la BD. También se usará *tokens* **anti-CSRF** para evitar falsificación de peticiones.
- **Plantillas** para generar con facilidad las páginas HTML devueltas, teniendo cuidado de mitigar posibles ataques XSS.
- La **interfaz de administración** nos servirá para editar[1] **usuarios, preguntas y respuestas**.

El proyecto Django se llamará **giw** y el *app* en el que desarrollaremos toda la aplicación web se llamará **preguntas**.

Rutas

La aplicación debe constar de las 7 rutas que se detallan a continuación, todas configuradas con el prefijo `/preguntas/` para todo el *app*. Todas las páginas generadas tendrán en común una cabecera con el título de la aplicación web («La web para preguntas y respuestas») y debajo una *barra de menú* con el siguiente comportamiento:

- Para usuarios **autenticados**, se mostrará un mensaje con el texto «Bienvenido `nombre_usuario`», seguido de un botón para cerrar la sesión y otro para ir a la página principal y ver todas las preguntas.
- Para usuarios **no autenticados**, contendrá el mensaje «No autenticado» y un botón para acceder a la página de inicio de sesión.

Las rutas concretas de la aplicación web serán:

1. GET /preguntas/: **página principal** de la aplicación web. Esta ruta es accesible para **todos los usuarios**, estén autenticados o no, y muestra un **listado enumerado** con todas las preguntas (título, autor, fecha y número de contestaciones). Si el usuario está autenticado, al final del listado de preguntas se mostrará un formulario para añadir una nueva pregunta escribiendo su título y su texto. Las preguntas deben aparecer **ordenadas de más nuevas a más antiguas**, estando arriba las más nuevas.
2. POST /preguntas/: ruta accesible únicamente a **usuarios autenticados**. Recibe los datos de un formulario con campos «**titulo**» y «**texto**» y añade una nueva pregunta del usuario actual a la BD. Tras insertar la pregunta, **redirige** al usuario a la página principal.
3. GET /preguntas/N: ruta accesible únicamente a **usuarios autenticados**. Muestra el título, autor, fecha y texto de la pregunta de **clave primaria N**. Si esa pregunta tiene respuestas, se muestra un listado enumerado con todas ellas, mostrando su texto, autor y fecha. Las respuestas deben aparecer **ordenadas de más nuevas a más antiguas**, estando arriba las más nuevas. Al final de la página se muestra un formulario para añadir una nueva respuesta a la pregunta actual. Si no existe ninguna pregunta con clave primaria N, debe genera un **error 404**.

[1] Crear, borrar, actualizar y consultar.

4. POST /preguntas/N/respuesta: ruta accesible únicamente a **usuarios autenticados**. Recibe un formulario con el campo «**texto**» y añade una nueva respuesta del usuario actual a la pregunta de **clave primaria N**. Una vez insertada la respuesta, **redirige** al usuario a la página que muestra los detalles de la pregunta **N**. Si no existe ninguna pregunta con clave primaria N, debe generar un **error 404**.
5. GET /preguntas/login: ruta accesible para todos los usuarios que muestra el formulario de inicio de sesión.
6. POST /preguntas/login: ruta accesible para todos los usuarios que maneja un intento de inicio de sesión. Recibe un formulario con los campos «**username**» y «**password**» y autentica al usuario. Si la autenticación es correcta, registra al usuario en la sesión de navegación y lo **redirige** a la página principal. Si la autenticación es incorrecta, muestra un página con el mensaje «Usuario o contraseña incorrectos».
7. GET /preguntas/logout: ruta accesible únicamente a **usuarios autenticados** que maneja el cierre de sesión del usuario actual. Elimina al usuario de la sesión de navegación y lo **redirige** a la página principal.

Modelos

Todos los modelos necesarios para el almacenamiento de usuarios ya nos los proporciona Django de manera automática, así que únicamente debemos preocuparnos de las clases que representan los objetos concretos de la apliación web. Específicamente tendréis que almacenar lo siguiente:

- **Preguntas**: contienen un **identificador autogenerado**, un **título** (máximo 250 letras), un **texto** (máximo 5000 letras), una **fecha** de creación (que se establece de manera automática) y un **autor** (usuario del sistema). Ningún campo puede ser nulo, y cuando se elimine un usuario se deben borrar todas sus preguntas.
- **Respuestas**: contienen un **identificador autogenerado**, un **texto** (máximo 5000 letras), una **fecha** de creación (que se establece de manera automática), un **autor** (usuario del sistema) y una **pregunta** a la que contesta. Ningún campo puede ser nulo, y cuando se elimine un usuario o una pregunta se deben borrar todas sus respuestas relacionadas.

Aspectos importantes

- Usad **decoradores** para restringir el acceso a usuarios autenticados y a los métodos HTTP válidos para cada vista (GET, POST o ambos).
- Una vista que reciba peticiones con distintos métodos HTTP puede realizar la distinción de casos y llamar a otras vistas especializadas para cada método. Tanto la vista inicial como las especializadas pueden tener sus propios **decoradores**.
- Lo más interesante es que los modelos escapen los campos que puedan contener código HTML **de forma automática**, para evitar XSS persistente.
- Los modelos son clases Python normales y se les puede añadir métodos. Por ejemplo, os puede resultar muy útil un método num_respuestas() en la clase Pregunta que devuelva el número de respuestas que tiene dicha pregunta.
- El módulo django.shortcuts contiene una función get_object_or_404() que realiza una búsqueda de un objeto de una clase con unas ciertas condiciones de búsqueda. Si lo encuentra devuelve ese objeto, y si no lanza una excepción Http404 que hace que una vista devuelva un error 404 (`https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/#get-object-or-404`).
- El mismo módulo django.shortcuts también contiene una función redirect() que genera una respuesta HTTP para redirigir al usuario a una ruta dada (`https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/#redirect`).
- Comprobad que no escribís ninguna URL «a mano» sino que las generáis mediante código a partir del nombre de sus rutas (**tanto en las plantillas como en las vistas**).
- No olvidéis registrar las preguntas y las respuestas para que se visualicen en la **interfaz de administración**.
- Utilizad los formularios tanto para **validar** entradas (añadir nueva pregunta, añadir nueva respuesta, iniciar sesión) como para **generar el código del formulario HTML** correspondiente.