Práctica 5: Desarrollo de una aplicación Web con Flask (III)

Ejercicio 1
Publicar mensajes

Ahora que los usuarios pueden iniciar sesión en la aplicación, vamos a permitirles publicar mensajes.

Primero, comprueba si ya tienes un formulario para publicar mensajes en la vista principal. Si no lo tienes, créalo. Tanto si ya tienes el formulario como si lo has creado, presta atención a lo siguiente:

- Haz que la acción del formulario apunte a un nuevo controlador que crearás más adelante en este ejercicio. Por ejemplo, puedes suponer que se llamará **`main.new_post`**, y por tanto puedes definirlo con **`action="{{ url_for('main.new_post') }}"`**.
- Envía los datos con una petición **`POST`**, ya que publicar un nuevo mensaje es una operación potencialmente insegura.
- Establece el nombre del área de texto donde los usuarios escribirán sus mensajes, para que luego puedas recuperar ese texto desde el controlador. Por ejemplo: **`<textarea name="text">`**.

A continuación, programa el nuevo controlador que recibirá los datos de este formulario en el *blueprint* **`main`**. Su código debe:

1. Obtener el texto del mensaje recibido desde el formulario (tienes ejemplos de cómo hacerlo en el *blueprint* **`auth`** de la práctica anterior).
2. Crear un nuevo objeto mensaje para representarlo, e inicializar sus atributos **`text`**, **`user`** y **`timestamp`**. Recuerda que puedes obtener el objeto del usuario que está autenticado actualmente desde la variable **`flask_login.current_user`** (y, por cierto, asegúrate de que sólo los usuarios autenticados pueden acceder a este controlador). Para ejemplos de cómo crear objetos mensaje, incluyendo cómo establecer su marca de tiempo, echa un vistazo a las demás funciones del mismo *blueprint* que tienes de las prácticas anteriores.

3. Guardar el objeto mensaje en la base de datos:

```
db.session.add(message)
db.session.commit()
```

4. Redirigir al usuario a la vista que muestra el mensaje recién creado. El controlador asociado a esa vista necesitará saber qué mensaje debe mostrar. Puedes hacerlo pasándolo a través de la URL: **`url_for("main.post", message_id=message.id)`**. Harás que ese controlador lea ese identificador de mensaje en el siguiente ejercicio. Hasta entonces, esa vista seguirá mostrando el mismo mensaje de ejemplo que en prácticas anteriores, en lugar del mensaje que el usuario acaba de crear.

Probablemente necesites importar símbolos como **`request`**, **`url_for`**, **`redirect`** o **`db`**. Consulta el *blueprint* **`auth`** para ver ejemplos.

Ejercicio 2
Mostrar mensajes

Ahora, reemplazaremos el controlador que muestra el contenido de un mensaje para que muestre un mensaje real de la base de datos en lugar del mensaje fijo que mostraba hasta ahora. Reemplaza tu versión anterior de la función por esta:

```
@bp.route("/post/<int:message_id>")
@flask_login.login_required
def post(message_id):
    message = model.Message.query.filter_by(id=message_id).first()
    if not message:
        abort(404, "Post id {} doesn't exist.".format(message_id))
    return render_template("main/post.html", post=message)
```

Presta atención a los siguientes detalles:

- Hemos cambiado su ruta a **`/post/<int:message_id>`**. Por eso, ahora se espera que las rutas contengan un entero con un identificador de mensaje (el del mensaje que se quiere mostrar). Por ejemplo, la ruta **`/post/42`** se usará para mostrar el mensaje con identificador 42. Ese valor será recibido por la función a través de su parámetro **`message_id`** debido a cómo ha sido declarada esta función:

  ```
  def post(message_id):
      (...)
  ```

- Se buscará el mensaje con el identificador recibido en la base de datos. Para ello, creamos una consulta sobre la columna **`id`** de la tabla de mensajes, y obtenemos sólo el primer resultado (ya que el identificador es una clave primaria, sólo una fila puede contener el identificador que estamos buscando).

- Si no recibimos ningún resultado, enviamos una respuesta HTTP de error 404 (no encontrado) y el controlador se detiene (recuerda importar **`abort`** desde el módulo **`flask`**).
- Si recibimos un resultado, se representa la plantilla para mostrar el mensaje.

A modo de curiosidad (no necesitas cambiar este código si no quieres), nota que podríamos haber producido incluso un código más compacto con la función de conveniencia **`first_or_404`** que proporciona **`flask_sqlalchemy`**. Esta función abortará automáticamente con el error 404 si no se recibe ningún resultado de la base de datos:

```
@bp.route("/post/<int:message_id>")
@flask_login.login_required
def post(message_id):
    message = model.Message.query.filter_by(id=message_id).first_or_404()
    return render_template("main/post.html", post=message)
```

Prueba a publicar nuevos mensajes y a acceder a sus vistas.

Ejercicio 3
Mostrar los mensajes más recientes

A continuación, modificaremos la página principal para que muestre los 10 mensajes más recientes de la base de datos. Ve a la función **`index`** del *blueprint* **`main`** y elimina todo el código anterior que creaba un usuario de ejemplo y mensajes de ejemplo. En su lugar, obtendrás y pasarás a la plantilla **`index.html`** los 10 mensajes más recientes con la siguiente consulta de SQLAlchemy:

```
messages = model.Message.query.order_by(model.Message.timestamp.desc()).limit(10).all()
```

Esta consulta obtiene los 10 primeros mensajes (debido a la llamada **`limit(10)`**) de la tabla de mensajes en orden descendente de marca de tiempo (debido a la llamada **`order_by(model.Message.timestamp.desc())`**). Los resultados se leen como una lista de objetos de mensaje (debido a la llamada **`all()`** al final).

Comprueba que tu código funciona. Vete a la página principal y comprueba que se muestran los mensajes más recientes. Si no tienes más de 10 mensajes en tu base de datos, publica algunos mensajes nuevos.

Ejercicio 4
Enlaces hacia la vista de mensajes

Modifica la plantilla **`post_template.html`** para que cada mensaje que muestra la aplicación enlace (de la forma que decidas) a la vista de mensaje asociada al mismo. Usa la función **`url_for`** de la misma forma que la estás usando en la última línea del controlador que crea mensajes nuevos.

Ejercicio 5
Perfil público de usuario

Modifica el controlador de perfil de usuario para que reciba un identificador de usuario (de la misma forma que lo hace el controlador de mensajes, por ejemplo con una ruta como **`/user/42`** para el usuario con identificador 42), y muestre los datos públicos de ese usuario desde la base de datos (en este caso, sólo sus nombres de usuario), así como todos sus mensajes ordenados de más a menos reciente.

La consulta de SQLAlchemy para seleccionar los mensajes que necesitas es, suponiendo que la variable con el usuario a mostrar se llama **`user`**:

```
messages = (
    model.Message.query
        .filter_by(user=user)
        .order_by(model.Message.timestamp.desc())
        .all()
)
```

Como ves, esta consulta filtra los mensajes por usuario, de forma que sólo se reciben aquellos del usuario a mostrar, y los ordena por marca de tiempo de más a menos reciente.

Por último, modifica la plantilla **`post_template.html`** para que el nombre del usuario autor del mensaje enlace a su perfil público. Añade también un enlace en la página principal que apunte al perfil público del usuario actualmente autenticado.