uc3m | Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

Aplicaciones Web
Grado en Ciencia e Ingeniería de Datos

Parcial 2
OpenCourseWare 2023
Duración: 30 min.

Solo una opción es correcta en cada pregunta de opción múltiple. Puntuación por respuesta correcta: 1 point. Penalización por respuesta incorrecta: 1/2 points.

Marca: ☒ Anula: ▨ No uses: ◯ ×

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir alguna de estas normas será motivo de expulsión inmediata del examen.
- Marca la respuesta a cada pregunta de opción múltiple con "X" en la tabla de abajo.
- Si no marcas ninguna opción o marcas más de una, la pregunta se cuenta como no contestada (no suma ni resta puntos).
- Rellena **tus datos personales** antes de comenzar a realizar el examen.

Nombre:	Grupo:

Firma:

NIA: | | | | | | | | | |

	A	B	C
1			
2			
3			
4			
5			

■ ■ ■

1.- ¿Cómo puedes prevenir que usuarios no autenticados accedan a un controlador dado en Flask?

(a) Todos los controladores en Flask bloquean los intentos de acceso de usuarios no autenticados.

(b) El controlador necesita consultar la base de datos para saber si hay una sesión de usuario válida.

(c) Con la extensión *flask-login* la función de ese controlador puede decorarse con `@flask_login.login_req`

2.- Una aplicación Web ejecuta código JavaScript en el lado del cliente para validar los valores introducidos por los usuarios en un formulario. ¿Debe la aplicación validar esos valores de nuevo en el lado del servidor cuando recibe los datos del formulario?

(a) No, porque es redundante.

(b) Solo si la aplicación funciona sobre HTTP en lugar de HTTPS. Only if the application works on top of HTTP instead of HTTPS.

(c) Sí, porque los atacantes pueden saltarse la validación de datos en el lado del cliente, incluso si la aplicación funciona sobre HTTPS.

3.- El siguiente fragmento de código Python que construye una consulta a enviar a la base de datos, donde las variables `name` y `password` provienen de un formulario de autenticación:

```
query = f"""
  SELECT id, name, fullName, balance FROM Users
  WHERE name='{name}'
  AND password='{password}'
"""
```

(a) Puede ser explotado para saltarse la autenticación de usuarios.

(b) Puede ser explotado para realizar un ataque de *cross-site scripting* reflejado.

(c) Es seguro.

4.- Tras haber definido las clases de tu modelo en Flask con SQLAlchemy:

(a) Necesitas insertar el código fuente Python de tus clases de modelo en la base de datos con sentencias `INSERT INTO`.

(b) Debes crear todas las tablas y sus columnas en la base de datos con sentencias `CREATE TABLE`.

(c) Puedes llamar a la función `create_all`, que creará automáticamente todas las tablas y sus columnas en la base de datos, conforme a las declaraciones en tus clases de modelo.

5.- El siguiente fragmento de código JavaScript que usa la biblioteca JQuery, donde el elemento con identificador `add` es un botón en un formulario:

```
$("#add").click(buy);
```

(a) Registra un escuchador de eventos: la función `buy` será llamada cuando el usuario haga clic en el botón.

(b) Hace que el botón pertenezca a la clase CSS llamada `buy`.

(c) Pincha el botón sin intervención del usuario.

Pregunta 1 (2 puntos)

Explica clara y precisamente por qué el siguiente fragmento de código debe estar dentro de la construcción `$(function() {...})`.

```
$(function() {
    var reference;
    for (reference in accessories) {
        var accessory = accessories[reference];
        var option = $("<option>")
                     .attr("value", reference)
                     .text(accessory.label
                           + " (" + accessory.price.toFixed(2) + "€)");
        $("#accessories").append(option);
    }
    $("#add").click(addAccessory);
});
```

Pregunta 2 (1,5 puntos)

Dado el siguiente fragmento de una plantilla Jinja:

```
<div class="message">
  <div class="text">{{ post.text }}</div>
  <div class="metadata">
  <span class="author">
    <a href="{{ url_for('__________.__________', ____________=_____________) }}">
      {{ post.user.name }}</a>
  </span>
  <span class="date">{{ post.timestamp }}</span></div>
</div>
```

Y la siguiente función en `main.py`:

```
@bp.route("/user/<int:user_id>")
def user(user_id):
    # Muestra la página de perfil de un usuario
    (...)
```

Completa la llamada a `url_for` de arriba para que el enlace apunte a la página de perfil del autor del mensaje. Supón que los atributos de la clase `User` son `id`, `email`, `name` y `password`.

Pregunta 3 (1,5 puntos)

Una aplicación Web necesita trabajar con una clase `ClassRoom` que debe ser mapeada automáticamente a la base de datos por el sistema ORM del framework Web. La clase contiene los siguientes campos: un identificador numérico (clave primaria auto-generada), un nombre de aula (limitado a 64 caracteres), una capacidad (número de asientos en el aula) y un edificio (una relación de muchos a uno a una clase modelo llamada `Building`, cuya clave primaria se llama `id`). Ningún campo puede tomar un valor NULL.

Programa la clase `ClassRoom` para Flask y SQLAlchemy.

Pista: necesitarás *algunas* de las construcciones de los ejemplos listados a continuación.

```
db.Model
db.Column(..., unique=..., nullable=...)
db.Column(..., primary_key=True)
db.Column(..., db.ForeignKey("..."), ...) (with table name dot column name inside)
db.String(512)
db.relationship(..., backref=..., lazy=...)
db.Date
db.Integer
```