UNIDAD 3.

Desarrollo de aplicaciones web (Back end).

(El estudiante demostrará el proceso para el desarrollo de una aplicación web.)

Tema III. Manejo de sesiones.

Evidencia de aprendizaje Tema III –RESUMEN

Temas	Saber	Saber hacer	Resultado de Aprendizaje
Manejo de sesiones	Identificar la importancia y el funcionamiento de sesiones en ambientes Web.	Desarrollar Aplicaciones Web que implementen el uso de sesiones.	Los estudiantes identifican los métodos de intercambio de información hacia el servidor y la base de datos.
Evidencia de aprendizaje			
A partir de un caso práctico realizar una aplicación web que contenga lo siguiente: Páginas web, hojas de estilos, gestión de contenido de base de datos desde formularios, implementación de seguridad por medio de sesiones.			

Información sobre el profesor

Profesor	Correo	Días de Clase
Bernardo Prado Díaz	Tenor_prado@yahoo.com.mx	Lunes y Martes

SABER: Fundamentos teóricos del manejo de sesiones

¿Qué es una sesión web?

Una **sesión** es un mecanismo que permite almacenar información del usuario entre distintas peticiones HTTP. Como HTTP es un protocolo sin estado, las sesiones permiten "recordar" quién es el usuario, qué ha hecho y qué datos ha enviado, incluso si navega entre distintas páginas.

¿Por qué son importantes las sesiones?

- **Persistencia de datos**: permiten mantener datos como nombre de usuario, carrito de compras, preferencias, etc.

- **Seguridad**: ayudan a controlar el acceso a recursos protegidos.
- **Experiencia personalizada**: permiten mostrar contenido adaptado al usuario.

¿Cómo funcionan las sesiones en Django?

Django gestiona sesiones automáticamente usando cookies. Cuando un usuario accede al sitio, Django crea una **cookie de sesión** que se asocia a un registro en la base de datos. Puedes guardar y recuperar datos usando el objeto request.session.

Características clave del sistema de sesiones en Django

- Se accede mediante request.session, que es un diccionario.
- Los datos se almacenan en el servidor (por defecto en la base de datos).
- La cookie enviada al cliente solo contiene el ID de sesión.
- Puedes configurar el tiempo de expiración, el almacenamiento y la seguridad.

SABER HACER: Aplicación práctica del manejo de sesiones en Django

Guardar datos en la sesión

python

```
def iniciar_sesion(request):
    request.session['usuario'] = 'Carlos'  # Guarda el nombre en la sesión
    request.session['rol'] = 'cliente'     # Guarda el rol del usuario
    return render(request, 'bienvenida.html')  # Redirige a otra vista
```

Cada dato se guarda como un par clave-valor, igual que en un diccionario.

Recuperar datos de la sesión

python

```
def mostrar_usuario(request):
    nombre = request.session.get('usuario', 'Invitado')  # Recupera el nombre o muestra 'Invitado'
    rol = request.session.get('rol', 'sin rol')          # Recupera el rol
    return render(request, 'perfil.html', {'nombre': nombre, 'rol': rol})
```

Usar .get() evita errores si la clave no existe.

Eliminar datos de la sesión

python

```
def cerrar_sesion(request):
    request.session.flush()  # Elimina toda la sesión
    return redirect('iniciar_sesion')  # Redirige al login
```

flush() borra todos los datos y genera una nueva sesión.

Ejemplo completo: control de acceso por sesión

python

```
def vista_protegida(request):
    if 'usuario' in request.session:  # Verifica si el usuario está en sesión
        return render(request, 'privado.html')  # Muestra contenido protegido
    else:
        return redirect('iniciar_sesion')  # Redirige si no hay sesión activa
```

Configuración adicional (opcional)

En settings.py puedes ajustar el comportamiento de las sesiones:

python

SESSION_COOKIE_AGE = 3600 # Tiempo de vida de la sesión en segundos (1 hora)

SESSION_EXPIRE_AT_BROWSER_CLOSE = True # Expira al cerrar el navegador

SESSION_SAVE_EVERY_REQUEST = True # Guarda la sesión en cada petición

Resultado de aprendizaje

- **Identificar** el propósito y funcionamiento de las sesiones en aplicaciones web.
- **Comprender** cómo Django gestiona las sesiones de forma segura y eficiente.
- **Desarrollar** aplicaciones web que usen sesiones para personalizar la experiencia del usuario, proteger vistas y mantener datos persistentes.
- **Aplicar** buenas prácticas en el manejo de sesiones, como validación, expiración y limpieza de datos.

Agregado para practica:
SABER: Fundamentos teóricos de la autenticación con sesiones

¿Qué es la autenticación?

La autenticación verifica que un usuario es quien dice ser. En Django, esto se logra mediante el sistema de usuarios (django.contrib.auth) que permite:

- Iniciar sesión
- Cerrar sesión
- Restringir acceso a vistas
- Asignar permisos y roles

¿Cómo se relaciona con las sesiones?

Una vez que el usuario se autentica, Django guarda su información en la sesión. Esto permite que el servidor lo "recuerde" en cada petición, sin necesidad de volver a autenticarse.

- La sesión se gestiona con request.session
- El usuario autenticado se accede con request.user
- Django usa cookies para mantener el ID de sesión

Componentes clave del sistema de autenticación

Componente	Función
User model	Representa al usuario
AuthenticationForm	Formulario de login
login()	Inicia sesión
logout()	Cierra sesión
@login_required	Protege vistas
request.user.is_authenticated	Verifica si el usuario está logueado

SABER HACER: Implementación práctica en Django

Paso 1: Crear formulario de login

python

```
# forms.py
from django.contrib.auth.forms import AuthenticationForm
```

Django ya incluye este formulario, no necesitas definir campos manualmente.

Paso 2: Vista para iniciar sesión

python

```
# views.py
from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect
from django.contrib.auth.forms import AuthenticationForm

def iniciar_sesion(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)  # Crea el formulario con los datos enviados
        if form.is_valid():  # Valida credenciales
            usuario = form.get_user()  # Obtiene el usuario autenticado
            login(request, usuario)  # Inicia sesión y guarda en la sesión
            return redirect('dashboard')  # Redirige a vista protegida
    else:
        form = AuthenticationForm()  # Muestra formulario vacío
    return render(request, 'login.html', {'form': form})
```

Paso 3: Vista protegida con sesión activa

python

```
from django.contrib.auth.decorators import login_required

@login_required(login_url='iniciar_sesion')
```

```
def dashboard(request):
    return render(request, 'dashboard.html', {'usuario': request.user})
```

Paso 4: Cerrar sesión

python

```
from django.contrib.auth import logout

def cerrar_sesion(request):
    logout(request)  # Elimina la sesión del usuario
    return redirect('iniciar_sesion')
```

Paso 5: Plantilla HTML para login

html

```
<!-- login.html -->
<h2>Iniciar sesión</h2>
<form method="POST">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Entrar</button>
</form>
```

Paso 6: Mostrar usuario en sesión

html

```
<!-- dashboard.html -->
<h2>Bienvenido, {{ usuario.username }}</h2>
<a href="{% url 'cerrar_sesion' %}">Cerrar sesión</a>
```

Resultado de aprendizaje

- Identificar cómo funciona la autenticación y su relación con las sesiones.
- Comprender el flujo completo de login/logout en Django.
- Desarrollar aplicaciones web que gestionen usuarios autenticados.
- Aplicar seguridad y control de acceso usando decoradores y validaciones.

Manejo de Sesiones en Aplicaciones Web con Python y Django

1 SABER – Dimensión Conceptual

Objetivo: Comprender la importancia y el funcionamiento de las sesiones en entornos web y cómo implementarlas en aplicaciones desarrolladas con Python y Django.

A) Conceptos Clave

• Sesión:
Definición: Mecanismo que permite almacenar información específica de un usuario en el servidor durante su interacción con una aplicación web.
Ejemplo: Ejemplo: Mantener a un usuario autenticado mientras navega entre páginas.

• ID de sesión:
Definición: Identificador único asignado a cada sesión de usuario.
Ejemplo: Ejemplo: token o cookie generada por Django para identificar la sesión.

• Cookie:
Definición: Archivo pequeño almacenado en el navegador del cliente que contiene datos de sesión o preferencias.
Ejemplo: Ejemplo: 'sessionid' en Django.

• Persistencia de sesión:
Definición: Capacidad de mantener datos a lo largo de múltiples peticiones HTTP.
Ejemplo: Ejemplo: Carrito de compras que recuerda los productos agregados.

• Seguridad en sesiones:
Definición: Medidas para proteger los datos de sesión contra ataques como secuestro de sesión o CSRF.
Ejemplo: Ejemplo: Uso de HTTPS y regeneración de ID de sesión.

2 SABER HACER – Dimensión Actuacional

Objetivo: Implementar sesiones en aplicaciones web desarrolladas con Django.

A) Ejemplo de Uso de Sesiones en Django

```
# archivo views.py
from django.shortcuts import render, redirect

def iniciar_sesion(request):
    if request.method == 'POST':
        usuario = request.POST.get('usuario')
        request.session['usuario'] = usuario
        return redirect('bienvenida')
    return render(request, 'login.html')

def bienvenida(request):
    usuario = request.session.get('usuario', 'Invitado')
    return render(request, 'bienvenida.html', {'usuario': usuario})

def cerrar_sesion(request):
    try:
        del request.session['usuario']
    except KeyError:
        pass
    return redirect('login')
```

```
<!-- archivo templates/login.html -->
<form method="POST">
    {% csrf_token %}
    <input type="text" name="usuario" placeholder="Usuario">
    <button type="submit">Iniciar Sesión</button>
</form>
```

```
<!-- archivo templates/bienvenida.html -->
<h1>Bienvenido, {{ usuario }}</h1>
<a href="{% url 'cerrar_sesion' %}">Cerrar sesión</a>
```

3 SER Y CONVIVIR – Dimensión Socioafectiva

Objetivo: Fomentar el trabajo colaborativo y la responsabilidad en el manejo de información sensible en aplicaciones web.

- Respetar la privacidad de los datos del usuario.

- Asignar roles claros para el desarrollo y seguridad de sesiones.

- Revisar el código en equipo para evitar vulnerabilidades.

- Aplicar buenas prácticas de programación segura.

4 Glosario

- Sesión → Mecanismo para almacenar información de usuario en el servidor.

- Cookie → Archivo pequeño que guarda información de usuario en el navegador.

- ID de sesión → Identificador único de cada sesión.

- Persistencia → Mantenimiento de datos durante varias peticiones.

- CSRF → Ataque que fuerza a un usuario a ejecutar acciones no deseadas.

- HTTPS → Protocolo seguro para transmitir datos cifrados.