



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENDIZAJE

- Denominación del Programa de Formación: Técnico en programación de Software
- Código del Programa de Formación: 233104-V2
- Nombre del Proyecto Formativo (si aplica): DESARROLLO DE APLICACIONES DE SOFTWARE PARA EL SECTOR EMPRESARIAL
- Fase del Proyecto (si aplica): Ejecución
- Actividad de Proyecto Formativo (si aplica): Desarrollar el software de acuerdo con el diseño y a la programación establecida
- Competencia: 220501096 DESARROLLAR LA SOLUCIÓN DE SOFTWARE DE ACUERDO CON EL DISEÑO Y LA PROGRAMACIÓN ESTABLECIDA
- Resultados de Aprendizaje: Rap 3. CODIFICAR EL SOFTWARE EMPLEANDO EL LENGUAJE DE PROGRAMACIÓN SEGÚN REQUERIMIENTO Y DISEÑO
- Duración de la Guía de Aprendizaje (horas): 72 horas (24 sincrónicas + 48 asincrónicas)

2. PRESENTACIÓN

Tu aplicación de productos ya tiene catálogo, vista de detalle y formularios para crear, editar y eliminar productos. Pero hay un problema: en este momento, CUALQUIER persona que entre a tu sitio puede borrar todo el catálogo, sin necesidad de iniciar sesión ni demostrar quién es. Es como si la puerta de una tienda no tuviera llave y cualquiera pudiera entrar a reorganizar o llevarse la mercancía.

Piensa en cómo funcionan las apps que usas todos los días: para entrar a tu cuenta de Instagram, Spotify o tu plataforma de videojuegos, primero debes iniciar sesión con un usuario y una contraseña. Solo entonces la app sabe quién eres y qué puedes hacer: ver tu perfil, publicar contenido, cambiar tu configuración, etc. Eso se llama autenticación.



En esta guía vas a aprender a usar el sistema de autenticación que viene incluido en Django (django.contrib.auth) para que tu aplicación de productos tenga usuarios reales: cada persona podrá registrarse, iniciar sesión, cerrar sesión, y solo los usuarios autenticados podrán crear, editar o eliminar productos.

En esta guía vas a:

Conocer el sistema de autenticación de Django y el modelo User.

Crear un superusuario y explorar el panel de administración.

Construir un formulario de registro para que nuevos usuarios creen su cuenta.

Implementar las vistas de inicio de sesión (login) y cierre de sesión (logout).

Proteger las vistas de crear, editar y eliminar productos para que solo usuarios autenticados puedan usarlas.

Al finalizar esta guía, tu aplicación dejará de ser una puerta sin llave: cualquier visitante podrá ver el catálogo, pero solo los usuarios registrados y con sesión iniciada podrán modificarlo. ¡Vamos a ponerle seguridad a tu proyecto!

3. FORMULACIÓN DE LAS ACTIVIDADES DE APRENDIZAJE

- **Descripción de la(s) Actividad(es)**

3.1 Actividades de reflexión inicial:

Descripción de la actividad:

Reflexiona: cuando inicias sesión en Instagram o en tu correo, ¿cómo sabe la app que eres tú? ¿Qué pasa si alguien intenta ver tu perfil privado sin haber iniciado sesión? ¿Por qué crees que algunas páginas de tu proyecto deben estar protegidas? Comparte tus respuestas con el grupo.

Ambiente requerido: Ambiente de trabajo individual o en parejas con acceso a computador con Python 3.x instalado, entorno virtual activo, proyecto Django configurado y conexión a internet para consultar documentación.



Estrategias o técnicas didácticas activas: Lluvia de ideas; discusión guiada; preguntas orientadoras; trabajo en parejas.

Materiales de formación Computador con Python 3, Django y las librerías requeridas instaladas; editor de código (VS Code o similar); navegador web; terminal o consola del sistema.

Material de apoyo: Documentación oficial de Django (<https://docs.djangoproject.com/>); Documentación de Python (<https://docs.python.org/3/>); guías y videos del instructor; repositorio del proyecto en GitHub.

Duración de la actividad: 1 hora.

3.2 Actividades de contextualización e identificación de conocimientos necesarios para el aprendizaje:

Descripción de la actividad:

Autenticación vs. autorización

Aunque suelen confundirse, son dos cosas distintas:

Autenticación: responde a la pregunta “¿quién eres?”. Es el proceso de identificar a un usuario, normalmente con un usuario y una contraseña.

Autorización: responde a la pregunta “¿qué puedes hacer?”. Una vez Django sabe quién eres, decide qué páginas o acciones puedes usar.

`django.contrib.auth`: la app de autenticación

Cuando creas un proyecto nuevo con Django, este ya incluye una aplicación llamada `django.contrib.auth` dentro de `INSTALLED_APPS` en `settings.py`. Esta app trae “de fábrica”:

Un modelo `User`, con campos como `username`, `password`, `email`, `first_name` y `last_name`.

Formularios listos para usar, como `UserCreationForm` (registro) y `AuthenticationForm` (inicio de sesión).

Funciones para iniciar sesión, cerrar sesión y verificar si un usuario está autenticado.

Un sistema de contraseñas seguro: Django nunca guarda la contraseña como texto plano, sino que la cifra (hashing) antes de guardarla.

El superusuario y el panel de administración



Django incluye un panel de administración (/admin/) al que solo pueden entrar usuarios con permisos especiales. Para crear el primer usuario con todos los permisos (superusuario), se usa un comando de manage.py:

```
python manage.py createsuperuser
```

Este comando te pedirá un nombre de usuario, un correo electrónico (opcional) y una contraseña. Con esos datos podrás iniciar sesión en /admin/ y administrar usuarios, productos y categorías desde la interfaz gráfica.

Pasos para crear tu superusuario

Abre la terminal en la carpeta de tu proyecto (donde está manage.py).

Ejecuta el comando `python manage.py createsuperuser`.

Ingresa un nombre de usuario, correo electrónico y contraseña (la contraseña no se mostrará en pantalla mientras la escribes, eso es normal).

Inicia el servidor con `python manage.py runserver` y visita la ruta /admin/.

Inicia sesión con el usuario y contraseña que acabas de crear.

El formulario UserCreationForm

Así como usaste `ModelForm` para crear formularios a partir de `Producto`, Django ofrece un formulario ya construido para registrar nuevos usuarios: `UserCreationForm`. Este formulario incluye los campos `username`, `password1` y `password2` (la contraseña se pide dos veces, para confirmarla), y valida automáticamente que la contraseña sea suficientemente segura.

Vista de registro (views.py):

```
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm

def registro(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
```



```
form = UserCreationForm()

return render(request, 'productos/registro.html', {'form': form})
```

Esta vista sigue exactamente el mismo patrón que crear_producto de la guía anterior: si la petición es GET, muestra el formulario vacío; si es POST y los datos son válidos, guarda el nuevo usuario en la base de datos y redirige a la página de inicio de sesión.

Template de registro (registro.html):

```
{% extends 'productos/base.html' %}

{% block titulo %}Crear cuenta{% endblock %}

{% block content %}
<h2>Crear una cuenta nueva</h2>
<form method="POST">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Registrarme</button>
</form>
{% endblock %}
```

Pasos para crear el registro

En views.py, importa UserCreationForm desde django.contrib.auth.forms.

Crea la función registro siguiendo el ejemplo anterior.

En urls.py, agrega la ruta 'registro/' apuntando a registro, con name='registro'.

Crea el template registro.html, similar a crear.html de la guía anterior.

Vistas de autenticación ya incluidas en Django

Django incluye vistas listas para iniciar y cerrar sesión, en django.contrib.auth.views: LoginView y LogoutView. Solo necesitas conectarlas en urls.py y crear el template del formulario de login (LogoutView no necesita template propio: simplemente cierra la sesión y redirige).

```
from django.contrib.auth import views as auth_views

urlpatterns = [
    ...
    path('login/', auth_views.LoginView.as_view(template_name='productos/login.html'),
```



```
name='login'),
    path('logout/', auth_views.LogoutView.as_view(next_page='lista_productos'),
name='logout'),
]
```

template_name='productos/login.html': indica qué template usar para mostrar el formulario de inicio de sesión.

next_page='lista_productos': indica a dónde redirigir al usuario después de cerrar sesión.

Template de login (login.html):

```
{% extends 'productos/base.html' %}

{% block titulo %}Iniciar sesión{% endblock %}

{% block content %}
<h2>Iniciar sesión</h2>
<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Entrar</button>
</form>
{% endblock %}
```

Después de un login exitoso, Django redirige por defecto a la URL configurada en LOGIN_REDIRECT_URL (en settings.py). Agrega esta línea a tu settings.py:

```
LOGIN_REDIRECT_URL = 'lista_productos'
```

Saber quién inició sesión: request.user

En cualquier template, puedes usar la variable user para saber si hay alguien autenticado y mostrar contenido distinto según el caso. Esto es muy útil en base.html, para mostrar enlaces diferentes en el menú:

```
<nav>
    <a href="{% url 'index' %}">Inicio</a> |
    <a href="{% url 'lista_productos' %}">Catálogo</a> |
    {% if user.is_authenticated %}
        <span>Hola, {{ user.username }}</span> |
        <a href="{% url 'logout' %}">Cerrar sesión</a>
    {% else %}
        <a href="{% url 'login' %}">Iniciar sesión</a> |
        <a href="{% url 'registro' %}">Crear cuenta</a>
```



```
{% endif %}  
</nav>
```

`user.is_authenticated`: es True si hay un usuario con sesión iniciada, y False si es un visitante anónimo.

`user.username`: el nombre de usuario de la persona que inició sesión.

Pasos para crear el login y el logout

En `urls.py`, importa las vistas de autenticación: `from django.contrib.auth import views as auth_views`.

Agrega las rutas 'login/' y 'logout/' usando `LoginView` y `LogoutView`, como en el ejemplo.

Crea el template `login.html`.

En `settings.py`, agrega `LOGIN_REDIRECT_URL = 'lista_productos'`.

Modifica `base.html` para mostrar “Hola, usuario” y “Cerrar sesión” si hay sesión iniciada, o “Iniciar sesión” y “Crear cuenta” si no la hay.

Ambiente requerido: Ambiente de trabajo individual o en parejas con acceso a computador con Python 3.x instalado, entorno virtual activo, proyecto Django configurado y conexión a internet para consultar documentación.

Estrategias o técnicas didácticas activas: Aprendizaje basado en proyectos (ABP); lectura analítica de documentación técnica; resolución de problemas por indagación; trabajo colaborativo entre pares; retroalimentación formativa.

Materiales de formación Computador con Python 3, Django y las librerías requeridas instaladas; editor de código (VS Code o similar); navegador web; terminal o consola del sistema.

Material de apoyo: Documentación oficial de Django (<https://docs.djangoproject.com/>); Documentación de Python (<https://docs.python.org/3/>); guías y videos del instructor; repositorio del proyecto en GitHub.

Duración de la actividad: 4 horas.

3.3 Actividades de apropiación:

Descripción de la actividad:



Crea un superusuario para tu proyecto y, desde el panel de administración (/admin/), verifica lo siguiente:

Que aparezca la sección “Users”, con el usuario que acabas de crear.

Que tus modelos (por ejemplo, Producto y Categoría) estén registrados y se puedan ver y editar desde ahí (recuerda admin.py de la guía de ForeignKey).

Implementa la vista registro y su ruta en urls.py.

Crea el template registro.html con el formulario de registro.

En base.html, agrega un enlace “Crear cuenta” usando {% url 'registro' %}.

Prueba el registro: crea al menos un usuario nuevo (diferente del superusuario) desde el navegador.

Implementa las rutas de login y logout en tu proyecto.

Crea el template login.html.

Actualiza base.html para mostrar el estado de la sesión (autenticado o no) y los enlaces correspondientes.

Prueba el flujo completo: regístrate, inicia sesión, verifica que el menú cambie, y luego cierra sesión.

Ambiente requerido: Ambiente de trabajo individual o en parejas con acceso a computador con Python 3.x instalado, entorno virtual activo, proyecto Django configurado y conexión a internet para consultar documentación.

Estrategias o técnicas didácticas activas: Aprendizaje basado en proyectos (ABP); lectura analítica de documentación técnica; resolución de problemas por indagación; trabajo colaborativo entre pares; retroalimentación formativa.

Materiales de formación Computador con Python 3, Django y las librerías requeridas instaladas; editor de código (VS Code o similar); navegador web; terminal o consola del sistema.

Material de apoyo: Documentación oficial de Django (<https://docs.djangoproject.com/>); Documentación de Python (<https://docs.python.org/3/>); guías y videos del instructor; repositorio del proyecto en GitHub.



Evidencias de aprendizaje: Código fuente funcional del proyecto Django con las funcionalidades implementadas. Capturas de pantalla que evidencien el correcto funcionamiento en el navegador.

Instrumentos de evaluación: Lista de chequeo de funcionalidades implementadas. Revisión de código por pares.

Duración de la actividad: 6 horas.

3.4 Actividades de Transferencia el Conocimiento:

Descripción de la actividad:

Agrega `LOGIN_URL = 'login'` en `settings.py`.

Importa `login_required` desde `django.contrib.auth.decorators` en `views.py`.

Agrega el decorador `@login_required` encima de las vistas `crear_producto`, `editar_producto` y `eliminar_producto`.

Cierra sesión y verifica que, al intentar acceder a `'crear/'`, `'editar/<id>/'` o `'eliminar/<id>/'`, Django te redirige automáticamente al formulario de login.

Inicia sesión nuevamente y verifica que ahora sí puedes crear, editar y eliminar productos sin problema.

PARTE TEÓRICA:

Responde las siguientes preguntas:

Explica con tus palabras la diferencia entre autenticación y autorización, usando un ejemplo distinto al de la guía.

¿Qué formulario de Django se usa para registrar nuevos usuarios y qué campos incluye por defecto?

¿Qué hace la variable `user.is_authenticated` y dónde se puede usar?

¿Qué ocurre cuando un usuario sin sesión iniciada intenta acceder a una vista protegida con `@login_required`?

¿Por qué es importante que las contraseñas se guarden cifradas (hash) y no como texto plano en la base de datos?



Ambiente requerido: Ambiente de trabajo individual o en parejas con acceso a computador con Python 3.x instalado, entorno virtual activo, proyecto Django configurado y conexión a internet para consultar documentación.

Estrategias o técnicas didácticas activas: Aprendizaje basado en proyectos (ABP); lectura analítica de documentación técnica; resolución de problemas por indagación; trabajo colaborativo entre pares; retroalimentación formativa.

Materiales de formación Computador con Python 3, Django y las librerías requeridas instaladas; editor de código (VS Code o similar); navegador web; terminal o consola del sistema.

Material de apoyo: Documentación oficial de Django (<https://docs.djangoproject.com/>); Documentación de Python (<https://docs.python.org/3/>); guías y videos del instructor; repositorio del proyecto en GitHub.

Evidencias de aprendizaje: Actividad práctica con capturas de pantalla. Respuestas escritas a las preguntas teóricas (entregadas en documento o plataforma del instructor).

Instrumentos de evaluación: Rúbrica de evaluación de desempeño. Cuestionario de preguntas abiertas.

Duración de la actividad: 3 horas.

4. PLANTEAMIENTO DE EVIDENCIAS DE APRENDIZAJE PARA LA EVALUACIÓN EN EL PROCESO FORMATIVO.

Fase del proyecto formativo	Actividad del proyecto formativo	Actividad de Aprendizaje	Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación

Evidencia de Conocimiento: Respuestas a las preguntas teóricas de la actividad evaluativa (sección 3.4.2).

Evidencia de Desempeño: Implementación funcional de todas las actividades prácticas indicadas en las secciones 3.3 y 3.4.1, demostrada mediante capturas de pantalla y revisión del código fuente.



Evidencia de Producto: Proyecto Django actualizado con las funcionalidades de la guía correctamente implementadas, almacenado en repositorio Git.

5. GLOSARIO DE TÉRMINOS

Django: Framework web de alto nivel para Python que fomenta el desarrollo rápido y seguro.

Proyecto: Contenedor principal de una aplicación web, con configuraciones globales.

Aplicación (App): Módulo reutilizable que se encarga de una funcionalidad específica dentro de un proyecto.

MVT (Model-View-Template): Patrón de diseño de Django que organiza el código en tres capas lógicas: datos (Model), lógica (View) y presentación (Template).

DRY (Don't Repeat Yourself): Principio de desarrollo que busca reducir la duplicación de código.

ORM y CRUD: El ORM traduce código Python a consultas SQL; CRUD son las operaciones básicas de Crear, Leer, Actualizar y Eliminar datos.

Django Forms y ModelForm: Sistemas de Django para crear formularios; ModelForm genera los campos automáticamente a partir de un modelo.

Autenticación: Proceso mediante el cual una aplicación identifica quién es un usuario, normalmente con usuario y contraseña.

Autorización: Proceso mediante el cual una aplicación decide qué puede hacer un usuario ya identificado.

django.contrib.auth: Aplicación incluida en Django que provee el sistema de usuarios, contraseñas, login y permisos.

User: Modelo de Django que representa a un usuario, con campos como username, password y email.

Superusuario: Usuario con todos los permisos, creado con el comando `createsuperuser`, que puede acceder al panel de administración.

UserCreationForm: Formulario incluido en Django para registrar nuevos usuarios, con campos username, password1 y password2.



LoginView / LogoutView: Vistas incluidas en Django para iniciar y cerrar sesión, sin necesidad de escribirlas desde cero.

request.user: Objeto que representa al usuario que hace la petición; puede ser un usuario autenticado o un usuario anónimo.

user.is_authenticated: Propiedad que indica si el usuario actual inició sesión (True) o es un visitante anónimo (False).

@login_required: Decorador que obliga a que una vista solo pueda usarse si el usuario tiene una sesión iniciada.

LOGIN_URL / LOGIN_REDIRECT_URL: Configuraciones en settings.py que indican a dónde enviar al usuario para iniciar sesión y a dónde redirigirlo después de un login exitoso.

Hash (cifrado de contraseñas): Proceso que transforma una contraseña en una cadena de texto irreconocible, para que no se almacene en texto plano.

6. REFERENTES BIBLIOGRÁFICOS

Django Software Foundation. (2024). Django documentation. <https://docs.djangoproject.com/>

Mozilla Developer Network. (2024). Django Web Framework (Python). <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>

Holovaty, A. & Kaplan-Moss, J. (2009). The Definitive Guide to Django. Apress.

Vincent, W. S. (2022). Django for Beginners. Leanpub.

Python Software Foundation. (2024). The Python Tutorial. <https://docs.python.org/3/tutorial/>



7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Jheyson Fabian Villavisan	Instructor	CDM	25/06/2026

8. CONTROL DE CAMBIOS (diligenciar únicamente si realiza ajustes a la guía)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					