UNIDAD 4.

Publicación de aplicaciones web.

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

Tema I. Clasificación de los medios de publicación de una aplicación web.

Evidencia de aprendizaje Tema I –RESUMEN

Temas	Saber	Saber hacer	Resultado de Aprendizaje
Clasificación de los medios de publicación de una aplicación web	Identificar los diferentes servicios para la publicación de aplicaciones web.	Desarrollar aplicaciones Web que permitan la gestión de información en Bases de Datos.	Los estudiantes identifican los diferentes servicios para la publicación de una aplicación web.
Evidencia de aprendizaje			
A partir de un caso práctico realizar la publicación de una aplicación web y lo documenta en un reporte técnico que incluya: Listado de herramientas, secuencia de configuración, secuencia de publicación, pruebas del funcionamiento y publicación de la aplicación web.			

Información sobre el profesor

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

Unidad: Publicación de Aplicaciones Web: Clasificación de los medios de publicación de una aplicación web

Saber — Dimensión conceptual ¿Qué es una aplicación web?

Una aplicación web es un sistema interactivo que se ejecuta en un navegador y permite al usuario realizar tareas específicas, como consultar información, enviar formularios o gestionar datos.

Clasificación de medios de publicación: Los medios para publicar una aplicación web se clasifican en:

Hosting compartido: económico, ideal para proyectos pequeños (ej. Hostinger, GoDaddy).

VPS (Servidor Privado Virtual): mayor control y rendimiento (ej. DigitalOcean, Linode).

Plataformas PaaS (Platform as a Service): simplifican el despliegue (ej. Heroku, PythonAnywhere).

Servidores locales o institucionales: usados en entornos académicos o empresariales.

Tecnologías involucradas

Frontend: HTML, CSS, JavaScript
Backend: Django (framework de Python)
Base de datos: SQLite, PostgreSQL, MySQL

Saber Hacer — Dimensión actuacional

Habilidades prácticas

Diseñar interfaces web usando HTML y CSS.

Agregar interactividad con JavaScript (menús, validaciones, efectos).
Construir el backend con Django:
Crear modelos para gestionar datos.
Configurar rutas y vistas.
Conectar con bases de datos.
Publicar la aplicación en plataformas como:
PythonAnywhere
Render
Heroku

Ejemplo de flujo de trabajo

Diseño → Programación → Pruebas → Configuración del servidor → Publicación → Mantenimiento

Saber Ser-Convivir — Dimensión socioafectiva
Competencias blandas que se desarrollan
Pensamiento crítico: al analizar problemas y proponer soluciones computacionales.

Creatividad: al diseñar interfaces y explorar enfoques distintos para resolver tareas.
Colaboración: al trabajar en equipo, compartir código y documentar procesos.
Responsabilidad digital: al publicar contenido ético, seguro y funcional.

Aplicación práctica

Los estudiantes desarrollarán una aplicación web que permita gestionar información de usuarios, tareas o productos, y la publicarán en una plataforma gratuita como PythonAnywhere. El proyecto debe incluir diseño visual, interactividad, conexión a base de datos y documentación técnica.

Rúbrica de Evaluación: Proyecto de Aplicación Web con Django

Criterio	**Excelente (10)**	**Bueno (8)**	**Suficiente (6)**	**Insuficiente (≤5)**
Diseño de interfaz (HTML/CSS)	Estética profesional, responsiva y coherente	Diseño funcional con algunos detalles	Diseño básico, poco atractivo	Diseño desorganizado o incompleto
Interactividad (JavaScript)	Funcionalidades dinámicas bien integradas	Algunas funciones interactivas	Interactividad limitada	Sin interactividad
Lógica backend (Django)	Uso correcto de modelos, vistas y rutas	Funciona con algunos errores menores	Funcionalidad parcial	Fallos graves o sin backend
Gestión de base de datos	CRUD completo, datos persistentes	CRUD parcial, errores menores	Datos guardados pero sin edición o eliminación	Sin conexión funcional a BD

Criterio	Excelente (10)	Bueno (8)	Suficiente (6)	Insuficiente (≤5)
Publicación en plataforma web	Aplicación publicada y accesible	Publicación con errores menores	Intento de publicación sin éxito	No se intentó publicar
Documentación técnica	Clara, completa y bien estructurada	Documentación aceptable	Documentación incompleta	Sin documentación
Creatividad y originalidad	Solución innovadora y bien pensada	Enfoque interesante	Enfoque común	Sin creatividad evidente
Trabajo colaborativo	Evidencia de trabajo en equipo y roles definidos	Colaboración parcial	Trabajo individual con apoyo mínimo	Sin colaboración

Guía Paso a Paso: Publicar una App Web con Django en PythonAnywhere
Requisitos previos:
Tener instalado Django y una app funcional localmente.
Crear una cuenta gratuita en PythonAnywhere.

Pasos
Preparar tu proyecto

Asegúrate de tener tu app en una carpeta con manage.py, requirements.txt, y settings.py configurado para producción.

Subir archivos

En PythonAnywhere, ve a la pestaña Files y sube tu proyecto (puedes usar Git o subir archivos manualmente).

Crear entorno virtual

En la consola, ejecuta:

bash
python3.10 -m venv myenv
source myenv/bin/activate
pip install -r requirements.txt
Configurar base de datos

Ejecuta migraciones:

bash
python manage.py migrate
Configurar la app web

En la pestaña Web, crea una nueva app.

Define el path del archivo wsgi.py.

Configura el entorno virtual y los archivos estáticos.

Probar y ajustar

Accede a tu URL pública.

Corrige errores si aparecen (revisa logs en la pestaña Web > Error log).

¿ Ejemplo de Proyecto: "Gestión de Tareas Personales"

Objetivo

Desarrollar una aplicación web que permita a los usuarios registrar, editar, eliminar y visualizar tareas personales, utilizando Django como backend y HTML/CSS/JavaScript para el frontend.

Estructura del Proyecto

Componente	Descripción
Frontend	HTML para estructura, CSS para estilo, JS para interactividad (validaciones, alertas)
Backend	Django: modelos, vistas, rutas, templates
Base de datos	SQLite (por defecto en Django)
Funcionalidad	CRUD completo: Crear, Leer, Actualizar, Eliminar tareas

Modelo de datos (Django)

```
# models.py
from django.db import models

class Tarea(models.Model):
    titulo = models.CharField(max_length=100)
    descripcion = models.TextField()
    fecha_limite = models.DateField()
    completada = models.BooleanField(default=False)

    def __str__(self):
        return self.titulo
```

Interfaz básica (HTML + CSS)

html

```
<!-- templates/tareas/lista.html -->
<h1>Mis Tareas</h1>
<ul>
  {% for tarea in tareas %}
    <li>{{ tarea.titulo }} - {{ tarea.fecha_limite }}</li>
  {% endfor %}
</ul>
```

css

```
/* static/css/estilos.css */
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
}
h1 {
  color: #333;
}
```

Funcionalidad con JavaScript

```
// static/js/validaciones.js
function validarFormulario() {
```

```
  const titulo = document.getElementById("titulo").value;
  if (titulo.trim() === "") {
    alert("El título no puede estar vacío");
    return false;
  }
  return true;
}
```

Presentación para Clase: Introducción al Proyecto

Diapositivas sugeridas

Título: “Proyecto Web: Gestión de Tareas con Django”

Objetivo del proyecto

Tecnologías utilizadas

Estructura de la aplicación

Modelo de datos

Interfaz de usuario

Proceso de publicación (PythonAnywhere)

Evaluación y rúbrica

Creatividad y colaboración

Entrega final y retroalimentación

Clasificación de los Medios de Publicación de una Aplicación Web con Python y Django

1 SABER – Dimensión Conceptual

Objetivo: Comprender los diferentes servicios para la publicación de aplicaciones web y su clasificación, así como su relación con la gestión de información en bases de datos utilizando Python y Django.

A) Conceptos Clave

• Publicación de aplicaciones web:
Definición: Proceso de poner una aplicación en línea para que pueda ser accedida por usuarios a través de Internet.
Ejemplo: Ejemplo: Subir un proyecto Django a un servidor en la nube.

• Servicios de Hosting Compartido:
Definición: Servidor donde múltiples sitios web comparten los mismos recursos.
Ejemplo: Ejemplo: Hostinger, Bluehost.

• Servicios VPS (Servidor Privado Virtual):
Definición: Servidor virtualizado con recursos dedicados, mayor control y flexibilidad.
Ejemplo: Ejemplo: Contabo, Linode.

• Servidores Dedicados:
Definición: Servidor físico completo dedicado a un solo cliente.
Ejemplo: Ejemplo: OVH, Hetzner.

• Servicios en la Nube:
Definición: Infraestructura escalable con pago por uso.
Ejemplo: Ejemplo: AWS, Google Cloud, Azure.

• Plataformas PaaS (Platform as a Service):
Definición: Plataformas que simplifican el despliegue y gestión de aplicaciones.
Ejemplo: Ejemplo: Heroku, PythonAnywhere.

2 SABER HACER – Dimensión Actuacional

Objetivo: Desarrollar y publicar aplicaciones web con Django que gestionen información en bases de datos.

A) Ejemplo de Configuración para Publicación

```
# En Django, ajustar settings.py para producción
DEBUG = False
ALLOWED_HOSTS = ['mi-dominio.com']

# Configuración de base de datos para producción
```

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'NAME': 'mi_basedatos',
      'USER': 'usuario',
      'PASSWORD': 'contraseña',
      'HOST': 'host-db',
      'PORT': '5432',
   }
}
```

B) Ejemplo de Despliegue en Heroku

```
# Instalar Heroku CLI y ejecutar:
heroku login
heroku create nombre-app
git push heroku main

# Migraciones de base de datos
heroku run python manage.py migrate
```

3 SER Y CONVIVIR – Dimensión Socioafectiva

Objetivo: Fomentar la colaboración, la creatividad y la resolución de problemas en la publicación de aplicaciones web.

- Analizar en equipo las opciones de publicación más viables para el proyecto.

- Valorar la diversidad de ideas al elegir la plataforma de despliegue.

- Probar en conjunto el funcionamiento de la aplicación en el entorno de producción.

- Documentar el proceso de publicación para futuros proyectos.

4 Glosario

- Hosting → Servicio que permite alojar un sitio o aplicación web.

- VPS → Servidor virtual privado con recursos dedicados.

- PaaS → Plataforma como servicio que gestiona el despliegue y escalabilidad.

- Producción → Entorno en el que una aplicación está activa para usuarios finales.

- Escalabilidad → Capacidad de aumentar recursos según la demanda.