**Imagen que contiene Logotipo

El contenido generado por IA puede ser incorrecto.UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA**

**CENTRO UNIVERSITARIO SANTA ELENA PETEN**

**FACULTAD DE INGENIERIA EN SISTEMAS**

ING. FRANCISCO JAVIER ZEPEDA REYES

CURSO: DESARROLLO WEB - 22025-1690-036-A

TRABAJO:

Expo No. 2

Grupo 2

Aldo Mamfredo Yax Gutiérrez - 1690-22-6510

FECHA: 30/08/2025

Tabla de contenido

[Introducción 3](#_Toc207241372)

[Objetivos 4](#_Toc207241373)

[Objetivo General 4](#_Toc207241374)

[Objetivos Específicos 4](#_Toc207241375)

[Backend 5](#_Toc207241376)

[Python con Django 5](#_Toc207241377)

[Instalación y configuración básica 6](#_Toc207241378)

[Modelos ORM 6](#_Toc207241379)

[Ejemplo de un modelo en Django: 6](#_Toc207241380)

[Vistas 7](#_Toc207241381)

[URLs 8](#_Toc207241382)

[Templates 8](#_Toc207241383)

[Manejo de solicitudes 9](#_Toc207241384)

[Conclusiones 10](#_Toc207241385)

[Recomendaciones 11](#_Toc207241386)

[Bibliografía 12](#_Toc207241387)

[Cuestionario 13](#_Toc207241388)

# Introducción

El presente documento aborda el desarrollo de un sistema web para la gestión de pedidos de comida, utilizando Python con Django. Se describen los conceptos clave del backend, manejo de base de datos, despliegue en la nube, control de versiones y se incluye un análisis práctico de la implementación. El objetivo es demostrar el entendimiento y aplicación de herramientas modernas de desarrollo web en un proyecto funcional, así como documentar el proceso siguiendo las normas APA.

# Objetivos

## Objetivo General

Explicar los fundamentos del desarrollo Backend con Python utilizando Django.

## Objetivos Específicos

Implementar modelos ORM para manejar las entidades y relaciones de la base de datos.

Configurar vistas, URLs y templates para manejar la interacción con los usuarios.

Gestionar rutas y solicitudes HTTP de manera eficiente y segura.

# Backend

## Python con Django

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo. (Amazon, 2024)

Django es un framework web de alto nivel basado en Python que fomenta el desarrollo rápido y un diseño limpio y pragmático. Desarrollado por desarrolladores experimentados, se encarga de gran parte de las complicaciones del desarrollo web, para que puedas concentrarte en escribir tu aplicación sin tener que reinventar la rueda. Es gratuito y de código abierto. (Django, Django, 2025)

Entre sus características principales se encuentran:

ORM integrado: permite interactuar con bases de datos relacionales mediante clases y objetos, evitando el uso directo de SQL.

Sistema de rutas y vistas: facilita la organización del código y el manejo de solicitudes HTTP.

Sistema de plantillas: permite generar HTML dinámico de forma ordenada y reutilizable.

Seguridad: incluye protección contra ataques comunes, como inyecciones SQL, Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF).

Para este proyecto se utilizó **Visual Studio Code** como editor de texto, junto con un entorno virtual (venv) para aislar las dependencias del proyecto y evitar conflictos con otras instalaciones de Python.

### Instalación y configuración básica

#### Creación del entorno virtual:

python -m venv venv

#### Activación del entorno:

venv\Scripts\activate

#### Instalación de Django:

pip install django

#### Creación de un proyecto Django:

django-admin startproject PedidosComida

#### Creación de una aplicación dentro del proyecto:

python manage.py startapp pedidos

## Modelos ORM

Un modelo es la fuente única y definitiva de información sobre sus datos. Contiene los campos y comportamientos esenciales de los datos que almacena. Generalmente, cada modelo se asigna a una sola tabla de la base de datos. (Django, Documentacion Modelo ORM, 2025)

### Ejemplo de un modelo en Django:

from django.db import models

class Estudiante(models.Model):

nombre = models.CharField(max\_length=100)

edad = models.IntegerField()

correo = models.EmailField(unique=True)

def \_\_str\_\_(self):

return self.nombre

CharField: define un campo de texto con longitud máxima.

IntegerField: define un campo numérico entero.

EmailField: almacena correos electrónicos y valida automáticamente el formato.

def\_\_str\_\_(self): método que define la representación de un objeto como cadena.

El ORM de Django permite realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) de forma sencilla y consistente, y soporta múltiples motores de base de datos como SQLite, PostgreSQL, MySQL y Oracle.

## Vistas

Las vistas en Django son funciones o clases que procesan las solicitudes HTTP y devuelven respuestas. Pueden devolver HTML, JSON o redireccionar a otras URLs según la lógica de la aplicación. (Django, Documentacion General, s.f.)

Ejemplo de una vista básica:

from django.http import HttpResponse

def saludo(request):

return HttpResponse('Hola desde Django Backend')

En aplicaciones más complejas, las vistas se combinan con modelos y templates para generar páginas dinámicas y manejar formularios, autenticación de usuarios y otras funcionalidades del backend.

## URLs

Las URLs son las rutas que conectan las solicitudes HTTP con las vistas correspondientes. En Django, se definen en archivos urls.py dentro del proyecto y las aplicaciones. (Django, Documentacion General, s.f.)

#### Ejemplo:

from django.urls import path

from . import views

urlpatterns = [

path('saludo/', views.saludo),

]

path: define la ruta URL.

views.saludo: indica la vista que se ejecutará al acceder a la ruta.

El uso de un sistema de URLs organizado facilita la escalabilidad y el mantenimiento del proyecto.

## Templates

Django utiliza un sistema de plantillas que permite generar HTML dinámico y separar la lógica del backend del diseño visual. Esto ayuda a mantener el código limpio y facilita la colaboración entre desarrolladores y diseñadores. (Django, Documentacion General, s.f.)

Ejemplo de template:

<h1>Bienvenido {{ nombre }}</h1>

{{ nombre }}: muestra el valor de la variable nombre enviada desde la vista.

Los templates pueden extenderse mediante {% extends %} y {% block %} para reutilizar estructuras comunes (por ejemplo, encabezados y pies de página).

## Manejo de solicitudes

El backend se encarga de recibir solicitudes HTTP (GET, POST, PUT, DELETE), procesarlas mediante vistas y modelos, y enviar una respuesta adecuada al frontend. Esto incluye:

Validación de datos de formularios.

Consultas a la base de datos mediante el ORM.

Redireccionamientos a otras páginas o envío de datos en formatos como JSON.

Implementación de medidas de seguridad (CSRF, autenticación, autorización).

Por ejemplo, al crear un nuevo pedido, la solicitud POST enviada desde el frontend se procesa en la vista correspondiente, se guarda en la base de datos mediante el modelo Pedido y luego se redirige al usuario a la lista de pedidos actualizada. (Django, Documentacion General, s.f.)

# Conclusiones

El desarrollo backend con Python y Django permite estructurar aplicaciones de forma ordenada y eficiente. Django ofrece una solución completa con ORM, vistas, rutas y plantillas, lo que facilita la creación de aplicaciones escalables. El uso de modelos ORM simplifica el acceso a la base de datos y el manejo de la lógica de negocio.

# Recomendaciones

Implementar pruebas automáticas para asegurar la funcionalidad del sistema.

Revisar y optimizar la base de datos periódicamente para mejorar el rendimiento.

# Bibliografía

Amazon. (2024). *Aws Python*. Obtenido de https://aws.amazon.com/es/what-is/python/

Django. (13 de 8 de 2025). *Django*. Obtenido de Django: https://www.djangoproject.com/

Django. (2025). *Documentacion Modelo ORM*. Obtenido de https://docs.djangoproject.com/en/5.2/topics/db/models/

Django. (s.f.). *Documentacion General*. Obtenido de https://docs.djangoproject.com/es/5.2/

# Cuestionario

1. ¿Qué diferencia hay entre Django y Flask?

2. ¿Qué es un ORM y para qué sirve?

3. ¿Cómo se definen las rutas en Django?

4. ¿Qué ventaja tienen las vistas en un backend?

5. ¿Cómo se integran los templates en un proyecto de Django?