**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:

# models.py

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:

# views.py

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:

# urls.py

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.)

Base de datos SQL Server

¿Qué es SQL Server?

SQL Server es un sistema de gestión de bases de datos relacionales (RDBMS) desarrollado por Microsoft, diseñado para almacenar, organizar y administrar datos de manera estructurada y segura. Su arquitectura permite manejar grandes volúmenes de información, facilitando el acceso concurrente por múltiples usuarios y aplicaciones. SQL Server utiliza el lenguaje de consulta estructurada (SQL) como base, y extiende sus capacidades mediante Transact-SQL (T-SQL), un dialecto propio que incorpora funciones avanzadas como procedimientos almacenados, triggers, manejo de errores y control de flujo.

Este sistema se integra fácilmente con otras tecnologías del ecosistema Microsoft, como .NET, Azure y Power BI, y también ofrece compatibilidad con estándares abiertos, lo que lo convierte en una plataforma versátil para desarrolladores, analistas y administradores de sistemas. SQL Server está disponible en diversas ediciones (Express, Standard, Enterprise, Developer), cada una adaptada a distintos niveles de complejidad, presupuesto y carga de trabajo, desde proyectos personales hasta infraestructuras empresariales de misión crítica.

Además de su robustez técnica, SQL Server incluye herramientas gráficas como SQL Server Management Studio (SSMS), que permiten administrar bases de datos, ejecutar consultas, monitorear el rendimiento y configurar la seguridad de forma intuitiva. Su motor de base de datos está optimizado para garantizar alta disponibilidad, integridad de los datos y escalabilidad horizontal y vertical.

¿Cómo funciona?

SQL Server opera bajo una arquitectura modular que combina almacenamiento eficiente, procesamiento de consultas, seguridad avanzada y conectividad con múltiples clientes. Sus componentes principales incluyen:

1. Almacenamiento de datos

Los datos se organizan en tablas compuestas por filas y columnas, donde cada fila representa un registro y cada columna un atributo. A diferencia de una hoja de cálculo, SQL Server permite definir tipos de datos, restricciones, índices y relaciones entre tablas, lo que garantiza una estructura lógica y coherente. Además, el sistema gestiona automáticamente el espacio en disco, la fragmentación y la recuperación ante fallos.

1. Lenguaje SQL (T-SQL)

Los usuarios interactúan con SQL Server mediante T-SQL, que permite ejecutar comandos para crear objetos (tablas, vistas, funciones), manipular datos (INSERT, UPDATE, DELETE), consultar información (SELECT con filtros, agrupaciones y joins), y controlar transacciones. T-SQL también incluye estructuras de programación como bucles, condicionales y manejo de errores, lo que lo convierte en una herramienta poderosa para automatizar procesos dentro de la base de datos.

1. Modelo cliente-servidor

SQL Server sigue el paradigma cliente-servidor, donde las aplicaciones cliente (como sitios web, sistemas ERP o herramientas de análisis) envían solicitudes al servidor de base de datos. El servidor procesa esas solicitudes, accede a los datos y devuelve los resultados. Esta comunicación puede realizarse localmente o a través de redes, incluyendo conexiones seguras por internet. El modelo permite escalabilidad, separación de responsabilidades y acceso concurrente por múltiples usuarios.

1. Integridad de los datos

* Para garantizar la fiabilidad de la información, SQL Server implementa mecanismos como:
* Integridad referencial: Uso de claves foráneas para relacionar tablas y evitar inconsistencias.
* Restricciones: Validación de datos mediante reglas como NOT NULL, CHECK, UNIQUE.
* Propiedades ACID: Aseguran que las transacciones sean Atómicas, Consistentes, Aisladas y Duraderas, lo que protege los datos ante errores, caídas del sistema o accesos simultáneos.

¿Para qué se utiliza?

SQL Server es una herramienta fundamental en múltiples contextos tecnológicos y empresariales. Sus principales usos incluyen:

* Gestión de información estructurada

Permite almacenar y organizar grandes volúmenes de datos, como registros de clientes, inventarios, transacciones financieras, historiales médicos o contenidos multimedia. Su capacidad para definir relaciones entre entidades facilita la creación de modelos de datos complejos y precisos.

* Soporte para aplicaciones empresariales

SQL Server es el motor detrás de muchas aplicaciones críticas, tanto web como de escritorio. Proporciona una base sólida para sistemas de facturación, comercio electrónico, gestión de recursos humanos, control de producción, entre otros. Su rendimiento y seguridad lo hacen ideal para entornos que requieren alta disponibilidad y protección de datos sensibles.

* Análisis de datos e inteligencia empresarial

Incluye herramientas como SQL Server Analysis Services (SSAS), Integration Services (SSIS) y Reporting Services (SSRS), que permiten realizar análisis multidimensionales, integrar datos desde múltiples fuentes y generar informes dinámicos. Además, se puede integrar con plataformas de machine learning, servicios cognitivos y visualización de datos como Power BI, lo que potencia la toma de decisiones basada en datos.

* Escalabilidad y rendimiento

SQL Server está diseñado para crecer junto con las necesidades del negocio. Puede manejar desde bases de datos pequeñas en equipos personales hasta sistemas distribuidos con miles de usuarios simultáneos. Soporta replicación, particionamiento, clustering y otras técnicas que permiten mantener el rendimiento incluso en escenarios de alta demanda.

Conexión entre Django y SQL Server

1. Configuración en settings.py

El bloque DATABASES define los parámetros necesarios para que Django se conecte a una instancia de SQL Server:

|  |  |
| --- | --- |
| Parámetro | Función |
| ENGINE | Define el motor de base de datos (mssql requiere el paquete mssql-django) |
| NAME | Nombre de la base de datos (PedidosDB) |
| USER / PASSWORD | Credenciales de acceso al servidor SQL |
| HOST | Nombre del servidor SQL (en tu caso, MARKC\SQLEXPRESS) |
| PORT | Puerto de conexión (vacío usa el predeterminado: 1433) |
| OPTIONS | Especifica el driver ODBC necesario para la conexión |

Antes de esto, es importante de tener instalado el driver ODBC Driver 17 for SQL Server y el paquete Python mssql-django:



1. Migración de Datos en Django

Django no crea las tablas automáticamente al definir modelos. Usa un sistema de migraciones que traduce tus modelos Python a instrucciones SQL. Aquí está el flujo completo:

1. Crear migraciones



Esto genera archivos en la carpeta migrations/ que describen los cambios en tus modelos.

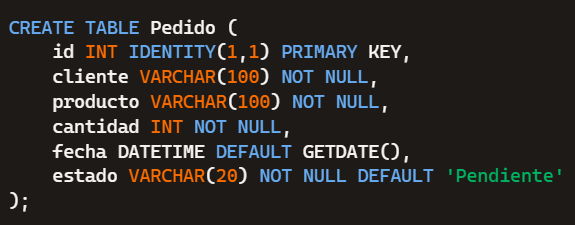
1. Aplicar migraciones



Django ejecuta las migraciones y crea las tablas en SQL Server, incluyendo Pedido con todos sus campos y restricciones.

Traducción del Modelo Pedido a SQL Server

El modelo se traduce a una tabla con esta estructura:



* id: se crea automáticamente como clave primaria
* estado: usa choices, lo que en Django se traduce a una validación en el formulario, pero no limita los valores en SQL Server (a menos que uses CHECK manualmente)

¿Cómo verificar que todo funciona?

1. Ejecuta python manage.py runserver
2. Accede a tus vistas (crear\_pedido, lista\_pedidos)
3. Inserta un pedido desde el formulario
4. Verifica en SQL Server Management Studio que el registro aparece en la tabla Pedido

idbi

# 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.

La integración de Django con SQL Server mediante el uso de mssql-django demuestra la flexibilidad del framework para adaptarse a entornos empresariales robustos. La configuración del motor, junto con el sistema de migraciones, permite traducir modelos Python a estructuras SQL de forma transparente y controlada. Esta capacidad de abstraer la lógica de persistencia sin sacrificar rendimiento ni seguridad convierte a Django en una herramienta poderosa para proyectos que requieren escalabilidad, integridad de datos y facilidad de mantenimiento.

# Recomendaciones

* Implementar pruebas automáticas para asegurar la funcionalidad del sistema.
* Revisar y optimizar la base de datos periódicamente para mejorar el rendimiento.
* Revisar y optimizar la base de datos periódicamente para mejorar el rendimiento, reducir redundancias y garantizar la integridad de los datos.
* Documentar claramente la arquitectura del backend y los modelos utilizados, facilitando futuras ampliaciones, mantenimiento colaborativo y auditorías técnicas.

# 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/

[**https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver17**](https://learn.microsoft.com/es-es/sql/sql-server/what-is-sql-server?view=sql-server-ver17)

https://intelequia.com/es/blog/post/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve

https://www.actian.com/es/what-is-sql-server/#:~:text=%C2%BFEs%20SQL%20Server%20una%20base,conocidos%20colectivamente%20como%20propiedades%20ACID.

# Cuestionario

1. ¿Qué lenguaje se usa con Django?

Python.

1. ¿Qué archivo define los modelos en Django?

models.py.

1. ¿Qué hace una vista en Django?

Procesa solicitudes y devuelve respuestas.

1. ¿Qué comando inicia un proyecto Django?

django-admin startproject.

1. ¿Qué devuelve HttpResponse?

Texto plano como respuesta HTTP.

1. ¿Quién desarrolló SQL Server?

Microsoft.

1. ¿Qué lenguaje usa SQL Server?

Transact-SQL (T-SQL).

1. ¿Qué herramienta administra SQL Server?

SQL Server Management Studio (SSMS).

1. ¿Qué estructura usa para guardar datos?

Tablas con filas y columnas.

1. ¿Qué propiedad asegura transacciones confiables?

ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).