RESPUESTAS A PREGUNTAS

1. Servidor web vs servidor de aplicaciones

Servidor web (ej: Apache, Nginx):
- Recibe solicitudes HTTP/HTTPS desde los navegadores.
- Está optimizado para servir contenido estático (HTML, CSS, JS, imágenes).
- Puede actuar como proxy inverso hacia un servidor de aplicaciones.
Ejemplo: Nginx entrega la página inicial estática de un restaurante.

Servidor de aplicaciones (ej: Gunicorn, uWSGI):
- Ejecuta el código de la aplicación (ejemplo: Django).
- Procesa la lógica de negocio, consultas a la base de datos y autenticación.
- Se comunica con el servidor web para responder solicitudes dinámicas.
Ejemplo: Gunicorn procesa el formulario de reservas en línea.

Resumen: El servidor web es como el portero, y el servidor de aplicaciones es el cocinero.

2. Protocolo DNS

DNS (Domain Name System) traduce nombres fáciles de recordar (reservasrestaurante.com) en direcciones IP que los servidores entienden.

Proceso:
1. El navegador consulta al resolver DNS.
2. El resolver busca en servidores raíz, TLD (.com) y luego el autorizado.
3. Devuelve la dirección IP al navegador.
4. El navegador ya puede conectarse al servidor real.

Ejemplo: Al pedir comida online, no escribes la IP, solo el nombre del sitio. DNS hace la traducción invisible.

3. Modelo vs Vista en Django

Modelo (Model):
- Representa los datos y la lógica de acceso a la base de datos.
- Define tablas, campos y relaciones.
Ejemplo: Clase 'Reserva' con nombre_cliente, fecha, hora.

Vista (View):
- Contiene la lógica de negocio que responde a solicitudes.
- Obtiene datos del modelo y decide qué plantilla usar.

Ejemplo: Vista 'hacer_reserva' guarda la información del formulario y muestra confirmación.

Resumen: El modelo es la base de datos inteligente y la vista es el mesero que coordina entre cliente y sistema.

4. HTTPS vs HTTP

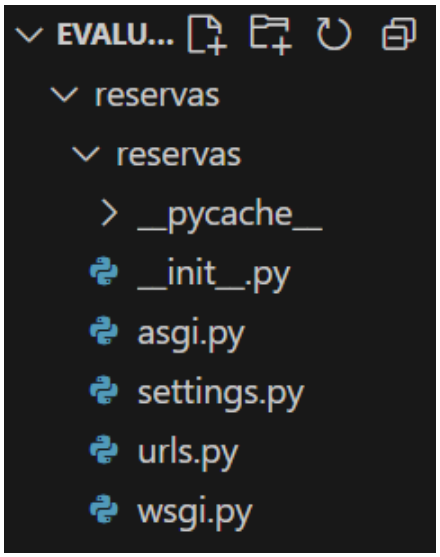
HTTP:
- Transmite datos en texto plano.
- Riesgo: Cualquiera puede leer datos sensibles.
Ejemplo: Enviar número de tarjeta en HTTP es inseguro.

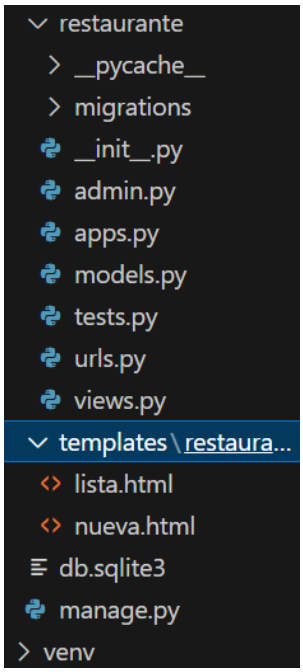
HTTPS:
- Usa certificados SSL/TLS para cifrar la comunicación.
- Ventajas: Confidencialidad, integridad y autenticidad.
Ejemplo: Al pagar comida online, HTTPS protege tu tarjeta contra atacantes.

Conclusión: Si un sistema maneja datos sensibles, HTTPS es obligatorio.

PANTALLAZOS

Proyecto y app creados correctamente, app registrada en INSTALLED_APPS. Rutas bien definidas y enlazadas.






```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'restaurante',
]
```

```python
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / "templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
```

Modelo Reserva completo con todos los campos correctos. Migraciones aplicadas y sin errores.

```python
from django.db import models

class Reserva(models.Model):
    nombre_cliente = models.CharField(max_length=100)
    fecha = models.DateField()
    hora = models.TimeField()
    personas = models.IntegerField()
    creada_en = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.nombre_cliente} - {self.fecha} {self.hora}"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell - reservas

```
(venv) PS C:\Users\woodp\Desktop\EVALUACION1BACK\reservas> python manage.py makemigrations
Migrations for 'restaurante':
  restaurante\migrations\0001_initial.py
    + Create model Reserva
(venv) PS C:\Users\woodp\Desktop\EVALUACION1BACK\reservas> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, restaurante, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0006_require_contenttypes_0002... OK
```

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying restaurante.0001_initial... OK
Applying sessions.0001_initial... OK
```

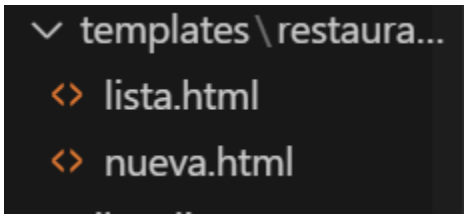
Implementa correctamente las 3 vistas (lista_reservas, nueva_reserva, cancelar_reserva) y sus templates, funcionando sin errores.

```python
from django.shortcuts import render, redirect, get_object_or_404
from .models import Reserva

def lista_reservas(request):
    reservas = Reserva.objects.all().order_by('-fecha')
    return render(request, 'restaurante/lista.html', {'reservas': reservas})

def nueva_reserva(request):
    if request.method == "POST":
        nombre = request.POST.get('nombre_cliente')
        fecha = request.POST.get('fecha')
        hora = request.POST.get('hora')
        personas = request.POST.get('personas')
        Reserva.objects.create(
            nombre_cliente=nombre,
            fecha=fecha,
            hora=hora,
            personas=personas
        )
        return redirect('lista_reservas')
    return render(request, 'restaurante/nueva.html')

def cancelar_reserva(request, id):
    reserva = get_object_or_404(Reserva, id=id)
    reserva.delete()
    return redirect('lista_reservas')
```



LISTAR Y AGREGAR RESERVAS

Lista de Reservas

Nueva Reserva

Cliente	Fecha	Hora	Personas	Acciones
No hay reservas registradas.				

Nueva Reserva

Nombre: MIGUEL CALDERON
Fecha: 16-10-2025
Hora: 10:33
Personas: 4
Guardar

Lista de Reservas

Nueva Reserva

Cliente	Fecha	Hora	Personas	Acciones
MIGUEL CALDERON	16 de octubre de 2025	10:33	4	Cancelar
MIGUEL CALDERON 2	4 de septiembre de 2025	10:37	4	Cancelar

CANCELAR RESERVAS

Lista de Reservas

Nueva Reserva

Cliente	Fecha	Hora	Personas	Acciones
MIGUEL CALDERON 2	4 de septiembre de 2025	10:37	4	Cancelar