

UNIVERSIDAD TÉCNICA DE COTOPAXI

*CARRERA INGENIERÍA EN SISTEMAS DE LA
INFORMACIÓN*

INTEGRANTES:

- MARÍA SALOMÉ AGILA GUEVARA

DOCENTE:

Ing. René Quisaguano

NIVEL:

SÉPTIMO SEMESTRE

TEMA:

Informe Comparativo

Octubre - Febrero

TEMA: Informe Comparativo del Proceso de Desarrollo: Diferencias entre Implementar Aplicaciones Web con C# y Python

INTRODUCCIÓN

En la actualidad, el desarrollo de aplicaciones web constituye una de las actividades más relevantes dentro del ámbito de la tecnología de la información. Entre los lenguajes más utilizados para este propósito destacan C# y Python, cada uno con sus propias fortalezas y debilidades. Este informe tiene como objetivo analizar y comparar los procesos de desarrollo con ambos lenguajes, poniendo énfasis en las diferencias más significativas que influyen en la selección del lenguaje según las necesidades y objetivos del proyecto

DESARROLLO

1. Características

Aspecto	C#: ASP.NET	Python: Django
Entorno de Desarrollo	Visual Studio es el IDE principal para C#, ofreciendo herramientas avanzadas como depuración y gestión de proyectos. Se utiliza con frameworks como ASP.NET, donde los archivos ASPX.CS gestionan la lógica de la aplicación y su interacción con el cliente.	Python es un lenguaje versátil y no depende de un ecosistema específico. Los frameworks más utilizados son Django y Flask. Django facilita el manejo de rutas mediante URLs y utiliza Views para gestionar la lógica de negocio.
Curva de Aprendizaje y Facilidad de Uso	C# tiene una sintaxis estricta y fuertemente tipada, lo que puede hacer su aprendizaje inicial más complejo.	Python es más accesible para principiantes gracias a su sintaxis sencilla y minimalista. Django simplifica la interacción con bases de datos mediante un ORM (Object Relational Mapper).

Rendimiento y Escalabilidad	C# ofrece un mejor rendimiento en tiempo de ejecución gracias a su compilación previa y optimización en el Common Language Runtime (CLR	Python es más lento porque es un lenguaje interpretado. Sin embargo, para aplicaciones web, la diferencia es mínima gracias a servidores eficientes. La simplicidad de Django permite implementaciones rápidas.
------------------------------------	---	---

2. Enrutamiento

En Django, el enrutamiento se realiza mediante el archivo `urls.py`. En este archivo, defines las rutas (URLs) de tu aplicación y las asocias a funciones llamadas vistas (views). Por ejemplo:

```

ones > Mascotas > urls.py
from django.urls import path
from . import views
urlpatterns = [
    # Propietarios
    path('', views.inicio),

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

```

En **C# con ASP.NET**, el enrutamiento se configura generalmente en `RouteConfig.cs` o en el archivo `Global.asax`. Aquí, asignas rutas a acciones de los. Por ejemplo, `routes.MapRoute` se utiliza para definir las rutas, lo que conecta una URL específica a un método de acción en un controlador de tu aplicación, se genera automáticamente.

```

Mascota.aspx  Reporte.aspx  RouteConfig.cs  styles1.css  Principal.aspx
MascotasPerdidass.RouteConfig
RegisterRoutes(RouteCollection r

using System.Web.Routing;
using Microsoft.AspNet.FriendlyUrls;

namespace MascotasPerdidass
{
    1 referencia
    public static class RouteConfig
    {
        1 referencia
        public static void RegisterRoutes(RouteCollection

```

3. Funcionalidad de vistas en Django y C# (aspx.cs)

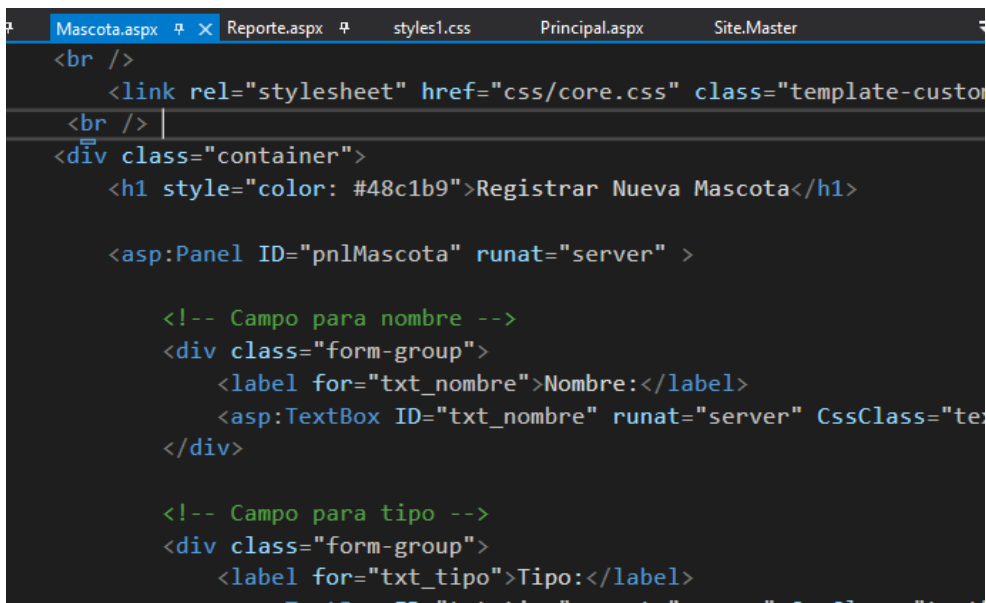
En **Django**, la funcionalidad se implementa dentro de las views. Estas son funciones o clases que reciben la solicitud HTTP y devuelven una respuesta al usuario, usualmente utilizando la función render.

```
from django.db.models import Max
from django.shortcuts import get_object_or_404
from django.contrib import messages

#Funcion para presentar en pantalla (renderizar) e
def inicio(request):
    return render(request, 'inicio.html')
```

En **C# con ASP.NET**, la funcionalidad se maneja a través de los archivos aspx.cs. Estos son archivos de código subyacentes asociados a las vistas .aspx. Mientras el archivo .aspx contiene la estructura HTML de la vista, el archivo aspx.cs contiene el código que le da la funcionalidad, como procesar eventos, validar formularios o consultar bases de datos.

.ASPX



```
Mascota.aspx  Reporte.aspx  styles1.css  Principal.aspx  Site.Master
<br />
<link rel="stylesheet" href="css/core.css" class="template-custom
<br />
<div class="container">
  <h1 style="color: #48c1b9">Registrar Nueva Mascota</h1>

  <asp:Panel ID="pnlMascota" runat="server" >

    <!-- Campo para nombre -->
    <div class="form-group">
      <label for="txt_nombre">Nombre:</label>
      <asp:TextBox ID="txt_nombre" runat="server" CssClass="te
    </div>

    <!-- Campo para tipo -->
    <div class="form-group">
      <label for="txt_tipo">Tipo:</label>
```

ASPX.CS

```

Mascota.aspx.cs Mascota.aspx Reporte.aspx styles1.css Principal.aspx Site.Master
MascotasPerdidass MascotasPerdidass.Mascota cadenaConexion
102 // Obtener los datos del formulario
103 string nombre = txt_nombre.Text;
104 string tipo = txt_tipo.Text;
105 string descripcion = txt_descripcion.Text;
106 string raza = txt_raza.Text;
107 int edad = int.Parse(txt_edad.Text);
108
109 int propietario_id = int.Parse(DropDownList1.SelectedValue);
110
111
112 bool exito = this.InsertarMascota(nombre, tipo, descripcion, raza, edad, propietario_id);
113
114 // Limpiar el formulario y mostrar el mensaje de éxito
115 if (exito)
116 {
117     LimpiarFormulario();
118     string script = "alert('Mascota insertada exitosamente');";
119     ClientScript.RegisterStartupScript(this.GetType(), "InsertSuccess", script, true);
120     CargarMascotas();
121 }

```

4. Plantillas en Django vs Site.Master en C# ASP.NET

En **Django**, plantillas es una característica clave que permite reutilizar el diseño de una aplicación en múltiples páginas. Se crea un archivo base (plantilla.html) que contiene el diseño principal de la aplicación y se pone en las demás páginas esto:

{% block content %}

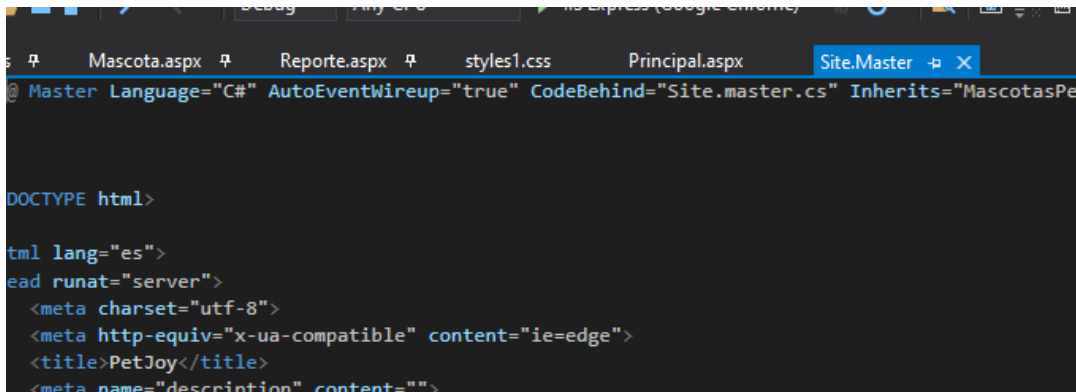
{% endblock %}

```

MASCO... Aplicaciones > Mascotas > templates > plantilla.html
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Sistema de Gestión de Mascotas</title>
8
9     <!-- Importando Bootstrap -->
10    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
11        integrity="sha384-QwTKZyypPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJY6hW+ALEwIH" crossorigin="anonymous">
12    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
13        integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous">
14

```

En **C#** con ASP.NET, el equivalente es el archivo Site.Master. Este archivo funciona como una plantilla principal para el sitio web, contiene la estructura general.



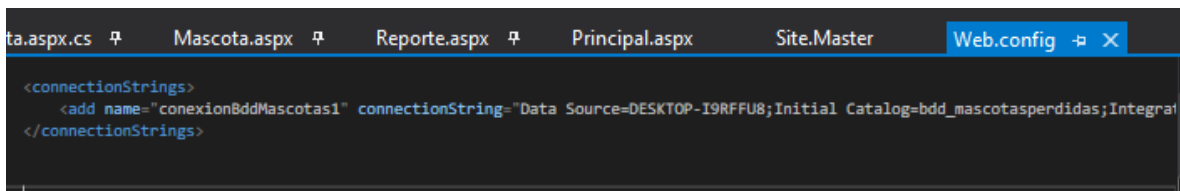
```
@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="MascotasPerdidas.Site.Master"

<!DOCTYPE html>

<html lang="es">
<head runat="server">
  <meta charset="utf-8">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>PetJoy</title>
  <meta name="description" content="">
```

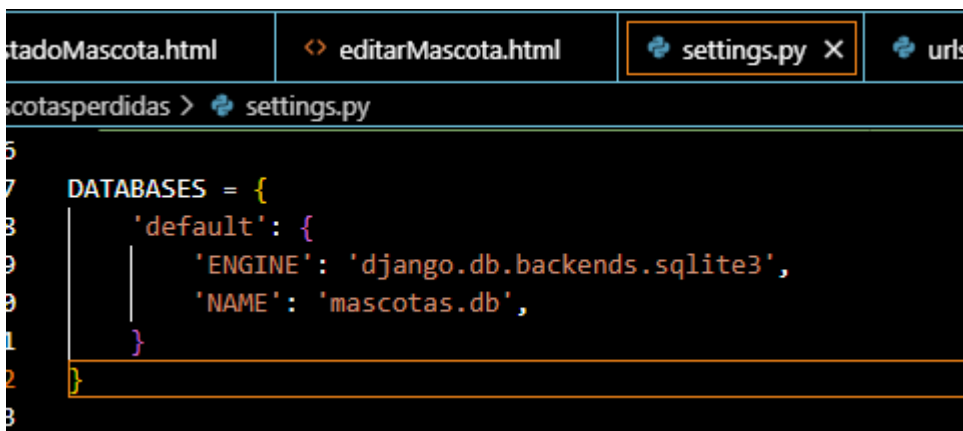
5. Configuración de la base de datos en C# ASP.NET vs Django

En **C# con ASP.NET**, la configuración de la base de datos se realiza a través del archivo `web.config`. Este archivo almacena la cadena de conexión y otros parámetros relacionados con la base de datos. En este archivo, defines el `connectionString`, que contiene la información necesaria para conectar la aplicación a la base de datos, como el servidor, el nombre de usuario, la contraseña y la base de datos



```
<connectionStrings>
  <add name="conexionBddMascotas1" connectionString="Data Source=DESKTOP-I9RFFU8;Initial Catalog=bdd_mascotasperdidas;Integrated Security=True;User ID=sa;" />
</connectionStrings>
```

En **Django**, la configuración de la base de datos se realiza en el archivo `settings.py`. Este archivo es la configuración principal de cualquier aplicación Django y contiene una sección específica para definir la base de datos que la aplicación utilizará.



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'mascotas.db',
    }
}
```

6. Gestión de Bases de Datos

C#: En proyectos realizados con ASP.NET, la gestión de bases de datos se realiza directamente mediante consultas SQL. También se hace uso de procedimientos almacenados

```
-- Tabla Propietario
CREATE TABLE Propietario (
    id_propietario INT IDENTITY(1,1) PRIMARY KEY,
    nombre NVARCHAR(100),
    apellido NVARCHAR(100),
    cedula NVARCHAR(10),
    telefono VARCHAR(15),
    direccion TEXT,
    email NVARCHAR(100)
);

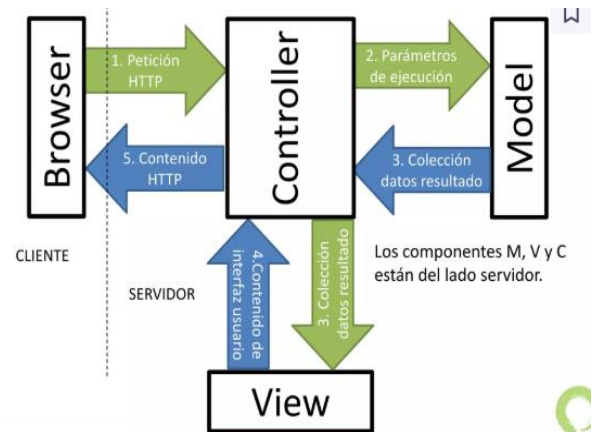
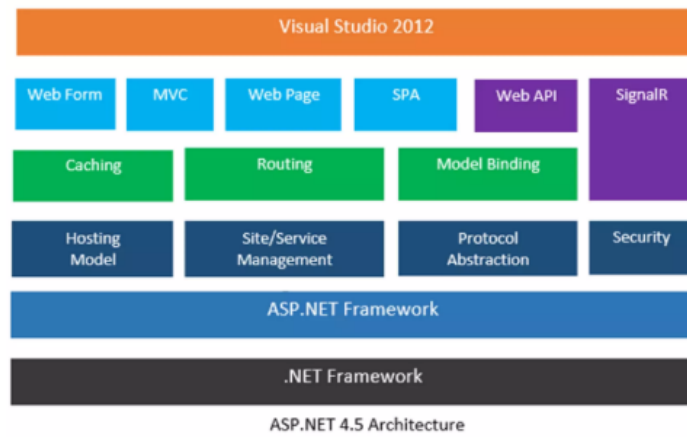
-- Procedimientos para Propietario
-- Insertar Propietario
CREATE PROCEDURE sp_insertar_propietario
    @Nombre NVARCHAR(100),
    @Apellido NVARCHAR(100),
    @Cedula NVARCHAR(10),
    @Telefono NVARCHAR(15),
    @Direccion TEXT,
    @Email NVARCHAR(100)
AS BEGIN
    -----
```

Python: Django utiliza un ORM que simplifica la conexión con bases de datos. Por ejemplo, se pueden definir modelos en el archivo models.py, y Django genera automáticamente las tablas correspondientes. Esto elimina la necesidad de escribir código SQL manualmente y acelera el proceso de desarrollo.

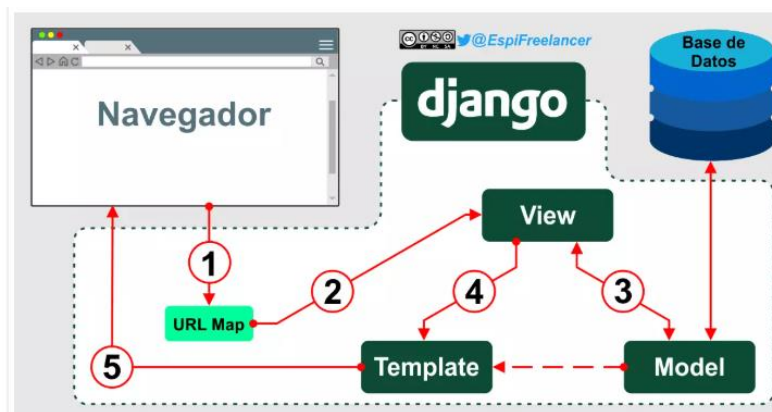
```
Aplicaciones > Mascotas > models.py
1  from django.db import models
2
3  # Modelo para Propietarios
4  class Propietario(models.Model):
5      id = models.AutoField(primary_key=True)
6      nombre = models.CharField(max_length=100)
7      apellido = models.CharField(max_length=100)
8      cedula = models.CharField(max_length=10)
9      telefono = models.CharField(max_length=15)
10     direccion = models.TextField()
11     email = models.EmailField()
12
```

7. Arquitectura

C# - ASP.NET

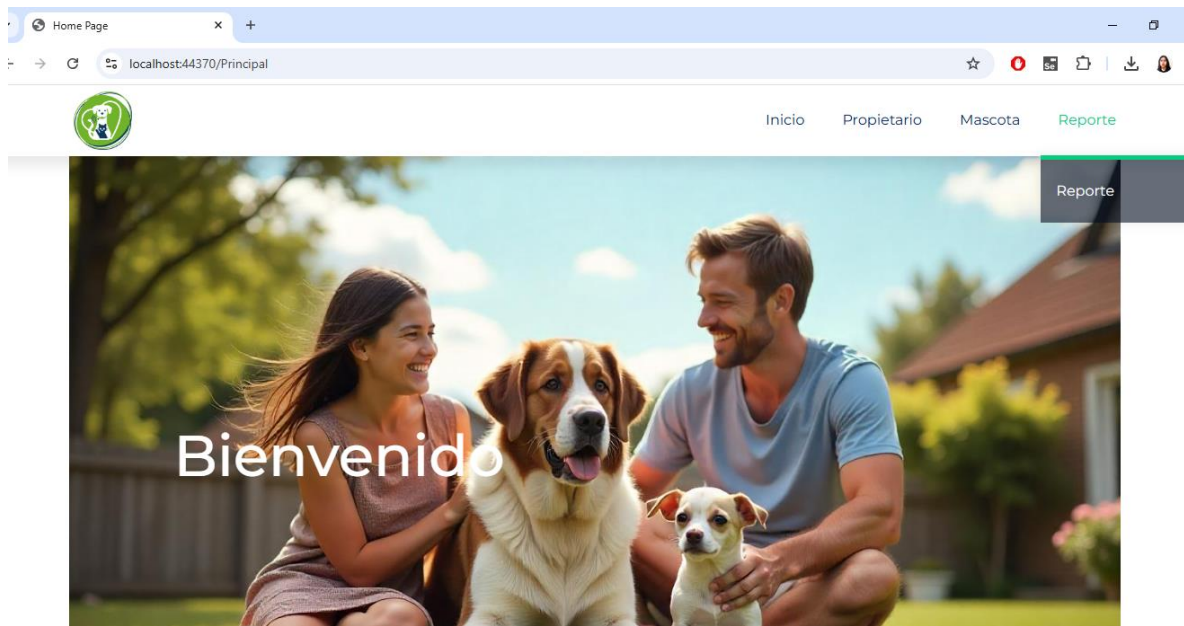


PYTHON – DJANGO

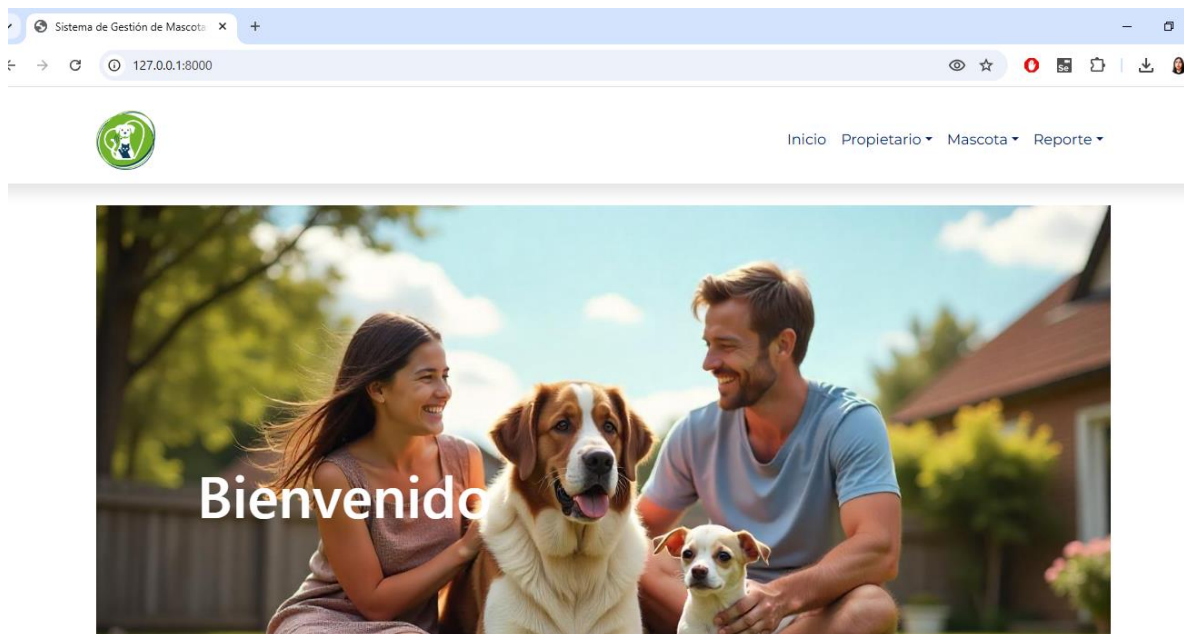


PÁGINAS FUNCIONALES

C# ASP.NET



PYTHON- DJANGO



CONCLUSIONES

- ✓ Se puede concluir que, la elección entre C# y Python para el desarrollo de aplicaciones web depende del contexto del proyecto y las prioridades del desarrollador.

- ✓ Python con el framework Django, destaca por su facilidad de uso, rapidez de desarrollo y herramientas integradas como el ORM, lo que lo hace una excelente opción para desarrollar en un menor tiempo.
- ✓ Desde mi punto de vista, desarrollar una aplicación en Django resultó más rápido debido a su enfoque en configuraciones predefinidas y herramientas integradas como el ORM. Por otro lado, en C#, aunque el desarrollo inicial es más lento por la configuración manual de varios componentes, la estructura final suele ser más robusta