# Análisis de la Aplicación Django

## 1. Introducción

Esta documentación detalla los requisitos funcionales, el análisis, el diseño y la estructura del proyecto basado en Django para una aplicación que incluye funcionalidades como gestión de usuarios, predicción mediante Machine Learning y generación de reportes.

## 2. Requisitos Funcionales

### 2.1. Gestión de Usuarios

- Registro de usuarios.

- Inicio de sesión con autenticación por correo electrónico y contraseña.

- Perfil de usuario editable.

### 2.2. Predicción mediante Machine Learning

- Recepción de datos de entrada desde el usuario.

- Ejecución del modelo de predicción en el servidor.

- Devolución de resultados al usuario con un resumen visual.

### 2.3. Reportes

- Generación de reportes basados en datos históricos.

- Descarga de reportes en formato PDF o Excel.

### 2.4. Gestión General

- Sistema de navegación fácil de usar.

- Interfaz adaptada para móviles y escritorios.

## 3. Estructura del Proyecto

### 3.1. Estructura General

```  
mi\_proyecto/  
├── usuarios/ # Gestión de usuarios  
│ ├── migrations/  
│ ├── templates/  
│ │ ├── registro.html  
│ │ ├── login.html  
│ │ ├── perfil.html  
│ ├── static/  
│ ├── models.py # Modelos de usuarios  
│ ├── views.py # Vistas para registro, login, perfil  
│ ├── urls.py # Rutas de la aplicación  
├── predictor/ # Predicción de ruleta  
│ ├── migrations/  
│ ├── templates/  
│ │ ├── prediccion.html  
│ ├── static/  
│ ├── models.py # Modelos para manejar datos  
│ ├── views.py # Lógica para realizar predicciones  
│ ├── urls.py # Rutas para predicción  
├── reportes/ # Gestión de reportes  
│ ├── migrations/  
│ ├── templates/  
│ │ ├── reportes.html  
│ ├── static/  
│ ├── models.py # Modelos relacionados con reportes  
│ ├── views.py # Lógica para generación de reportes  
│ ├── urls.py # Rutas para reportes  
├── manage.py # Archivo principal para comandos administrativos  
├── config/ # Configuración global del proyecto  
│ ├── \_\_init\_\_.py  
│ ├── asgi.py  
│ ├── settings.py # Configuración del proyecto  
│ ├── urls.py # Rutas generales del proyecto  
│ ├── wsgi.py  
```

## 4. Diagrama de Clases

```  
Clase Usuario  
---------------  
+ id: int  
+ email: str  
+ password: str  
+ fecha\_registro: datetime  
+ es\_admin: bool  
  
Clase Prediccion  
---------------  
+ id: int  
+ datos\_entrada: str  
+ resultado: str  
+ usuario: Usuario  
+ fecha\_prediccion: datetime  
  
Clase Reporte  
---------------  
+ id: int  
+ nombre: str  
+ usuario: Usuario  
+ contenido: str  
+ fecha\_generacion: datetime  
```

## 5. Detalles de la Implementación

### 5.1. Usuarios

\*\*`models.py`\*\*:

```python  
from django.contrib.auth.models import AbstractUser  
from django.db import models  
  
class Usuario(AbstractUser):  
 fecha\_registro = models.DateTimeField(auto\_now\_add=True)  
 es\_admin = models.BooleanField(default=False)  
```

\*\*`views.py`\*\*:

```python  
from django.shortcuts import render, redirect  
from django.contrib.auth import authenticate, login  
from .models import Usuario  
  
# Vista para registro  
def registro(request):  
 if request.method == 'POST':  
 # Lógica para registrar usuario  
 pass  
 return render(request, 'registro.html')  
  
# Vista para login  
def login\_usuario(request):  
 if request.method == 'POST':  
 # Lógica para autenticación  
 pass  
 return render(request, 'login.html')  
```

### 5.2. Predicción

\*\*`views.py`\*\*:

```python  
from django.shortcuts import render  
import tensorflow as tf  
import numpy as np  
  
# Cargar el modelo  
modelo = tf.keras.models.load\_model('ruta\_a\_modelo.keras')  
  
# Vista para predicción  
def prediccion(request):  
 if request.method == 'POST':  
 datos = request.POST['datos']  
 datos\_array = np.array([int(x) for x in datos.split(',')]).reshape(1, -1)  
 resultado = modelo.predict(datos\_array)  
 return render(request, 'prediccion.html', {'resultado': resultado})  
 return render(request, 'prediccion.html')  
```

## 6. Consideraciones Finales

- \*\*Seguridad\*\*: Implementar protección CSRF en formularios.

- \*\*Base de Datos\*\*: Usar PostgreSQL para escalabilidad.

- \*\*Pruebas\*\*: Incluir pruebas unitarias en cada aplicación.

- \*\*Documentación\*\*: Mantener comentarios claros en el código.