REDEPSE

Sistema de Registro y Gestión de Escuelitas Deportivas de la Provincia

Desarrollo profesional con Django Framework

¿Por Qué Django?

Desarrollo Rápido

Framework completo que acelera el desarrollo con herramientas integradas y arquitectura robusta.

Seguridad Integrada

Protección contra SQL injection, XSS, CSRF y gestión segura de sesiones por defecto.

Escalabilidad

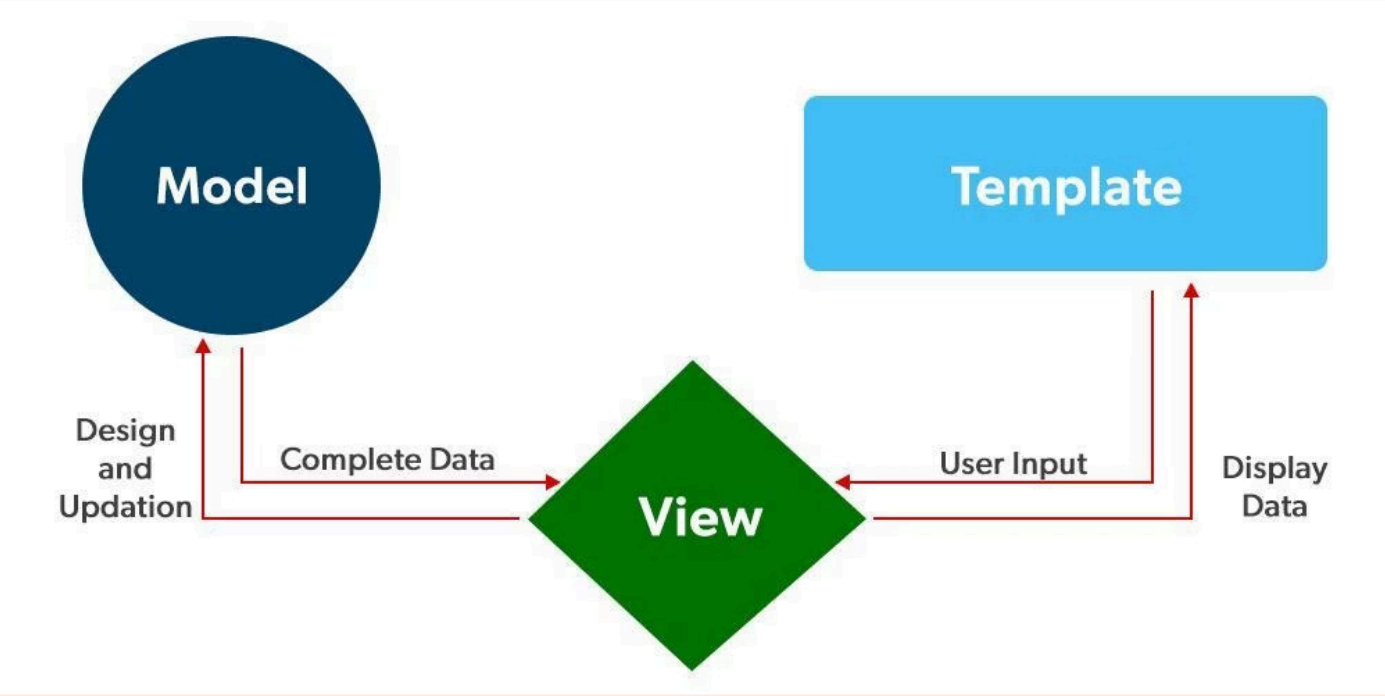
Arquitectura modular que permite crecer desde MVP hasta sistema empresarial complejo.

ORM Potente

Abstracción de base de datos que facilita migraciones y consultas complejas sin SQL directo.

Patrón MVT

Django implementa el patrón Model-View-Template para separar la lógica de negocio (los datos) de la lógica de presentación (cómo se ve):



Model: Define la estructura de la base de datos (tablas, campos) y cómo interactuar con ella.

View: Procesa las *requests* del usuario, interactúa con los *Models* para obtener o modificar datos, y luego decide qué *Template* usar.

Template: Es el **HTML** que recibe los datos de las *Views* y los muestra al usuario.

Al aplicar MTV, mantienes tu código **limpio** y **fácil de mantener**. Si quieres cambiar cómo se ve algo, solo tocas el **Template**. Si quieres cambiar cómo se guardan o procesan los datos, solo tocas el **Model**.

Arquitectura Profesional por Apps

Organización modular siguiendo el patrón de diseño empresarial de Django:

accounts/

Gestión completa de autenticación, usuarios, login y control de acceso por roles.

escuelas/

Lógica de negocio principal: registro, carga de datos y gestión de escuelitas deportivas.

core/ (redepse/)

Configuración central del proyecto, settings y utilidades compartidas entre apps.

> "Separación clara de responsabilidades: cada app maneja un dominio específico, facilitando mantenimiento y escalabilidad."

Django ORM

Django ORM permite definir modelos como clases Python, abstrayendo completamente la base de datos:

01

Modelos Declarativos

Escuela, Entrenadores, Alumnos, Canchas, Disciplinas y Documentos definidos como clases.

02

Relaciones Complejas

ForeignKey y ManyToMany para relaciones 1:N y N:N correctamente implementadas.

03

Migraciones Automáticas

Sistema de versionado de esquema que facilita cambios sin perder datos.

Componente	Archivo Clave	Función del ORM
Modelos (Clases)	models.py	**Define** la estructura de las tablas y las relaciones
Vistas (Controladores)	views.py	**Ejecuta** las operaciones CRUD sobre los datos.

ORM ¿Dónde lo utilizamos?

El ORM de Django nos permite interactuar con la base de datos a través de objetos Python, lo que simplifica el código y mejora la mantenibilidad. A continuación, se comparan operaciones comunes en SQL tradicional y su equivalente en Django ORM.

Operación	SQL Tradicional	Django ORM
1. Consultar un entrenador por DNI	SELECT * FROM entrenadores WHERE dni_ent = '12345678A';	Entrenador.objects.get(dni_ent='12345678A')
2. Crear una nueva escuela	INSERT INTO Escuela (nombre, direccion) VALUES ('Escuela Ejemplo', 'Calle Falsa 123');	Escuela.objects.create(nombre='Escuela Ejemplo', direccion='Calle Falsa 123')
3. Actualizar datos de un alumno	UPDATE Alumno SET nombre = 'Nuevo Nombre' WHERE id = 1;	Alumno.objects.filter(id=1).update(nombre='Nuevo Nombre')

Sistema de Formularios Django

Django Forms

Clases que definen campos, tipos y validaciones de forma declarativa.

Validación Automática

Validaciones client-side y server-side integradas, previniendo datos inconsistentes.

Protección CSRF

Tokens de seguridad automáticos en cada formulario contra ataques.

ModelForms

Generación automática desde modelos, reduciendo código repetitivo.

```
<form method="POST">
  {% csrf_token %}
  <input
    type="email"
    name="username"
    placeholder="Correo el
    required
  />
  <input
    type="password"
    name="password"
    placeholder="Contraseñ
    required
  />
  <input type="submit" val
</form>
<a href="{% url 'accounts:
<a href="{% url 'home' %}"
</div>
```

Autenticación y Seguridad

Sistema de Autenticación Integrado

Django proporciona un sistema completo de autenticación listo para usar:

- Hashing seguro de contraseñas con PBKDF2
- Gestión de sesiones y cookies seguras
- Sistema de permisos y grupos extensible
- Middleware de autenticación automático

Rol Escuela

Acceso a panel de carga de datos y documentación.

Rol Secretaría

Revisión, aprobación y gestión administrativa.

Arquitectura, Seguridad y Gobernanza del Proyecto

—> settings.py

Enfoque	Valor que Aporta a REDEPSE	Código de Implementación
Seguridad y Entornos	**Protección de datos sensibles** y separación clara entre Desarrollo (DEBUG=True) y Producción.	from decouple import config SECRET_KEY = config('SECRET_KEY', ...) DEBUG = config('DEBUG', ..., cast=bool)
Modularidad	El sistema está dividido en **aplicaciones especializadas**, logrando **escalabilidad** y un código altamente **mantenible**.	INSTALLED_APPS = [..., 'apps.usuarios', 'apps.escuelas', 'apps.secretarias', ...]
Integración DB	Configurado para la **portabilidad** y listo para integrarse con motores de DB robustos, como **MySQL.**	import pymysql DATABASES = {'default': {'ENGINE': config('DB_ENGINE'), ...}}

Configuración settings.py: Ejemplo

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'redepse',
    "apps.usuarios",
    "apps.escuelas",
    "apps.secretarias",
    "apps.accounts.apps.AccountsConfig",
]
```

Es la lista que le indica a Django qué **módulos** (aplicaciones) cargar al iniciar el proyecto. Demuestra el concepto de **Modularidad** , que es una base de la Arquitectura Empresarial.

Aqui se activan las funcionalidades **integradas** de Django, como la administración (`admin`), la seguridad (`auth`), sesiones, además de las aplicaciones personalizadas que contienen la lógica de negocio de REDEPSE.

Esto es la **Separación de Responsabilidades**. Cada *app* maneja un dominio de negocio específico

```
DATABASES = {
    'default': {
        'ENGINE': config('DB_ENGINE'),
        'NAME': config('DB_NAME'),
        'USER': config('DB_USER'),
        'PASSWORD': config('DB_PASSWORD'),
        'HOST': config('DB_HOST'),
        'PORT': config('DB_PORT', cast=int),
    }
}
```

Define la **conexión única** de Django con la base de datos. Este es el punto de encuentro entre el código Python y el motor de base de datos.

Las credenciales (nombre, usuario, contraseña), todos esos datos se leen de una **variable de entorno** (dentro de .env). Garantiza la **portabilidad**. Si cambiamos de MySQL a PostgreSQL, solo modificamos el archivo `.env`, **sin tocar el código** en `settings.py`. Asegura la **confidencialidad.**

Valor Técnico del Proyecto

REDEPSE: Sistema Real con Django

Más que un trabajo académico, es una aplicación profesional que demuestra:

Arquitectura Empresarial

Modularidad, separación de responsabilidades y código mantenible.

Seguridad Robusta

Autenticación, autorización y protección contra vulnerabilidades comunes.

Escalabilidad Real

Base sólida para agregar funcionalidades sin refactorizar estructura.

2

Apps

Modulares

100%

Django

Framework

> "Django permitió construir un sistema completo, seguro y escalable, aplicando patrones de diseño profesionales y mejores prácticas de la industria."

Integrantes del Proyecto

Desarrollo del Sistema REDEPSE con Django Framework

- Venier Rojas, Juan Cruz
- Perez Ledesma, Mariano
- Schuldt, Maximo
- Banegas Julián
- Ibañez, Juan