PeachStats

Diego Cuenca Arriols
IES TETUÁN DE LAS VICTORIAS

¿Qué es Peachstats?

- Una solución para equipos de tamaño mediano que necesitan gestionar sus estadísticas.
- Fácil de usar, sencilla de entender y muy personalizable.

Justificación y Beneficiarios

- Existen grandes empresas que mediante sensores toman miles de datos sobre un partido. Pero es extremadamente costoso.
- Manejar manualmente estadísticas es tedioso y mecánico.

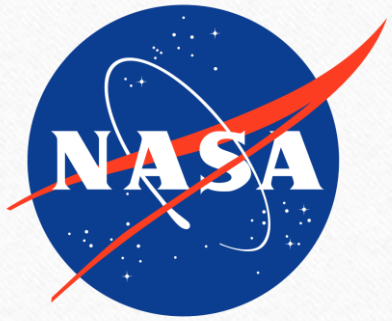
Usuarios

- Jugadores y equipo técnico: Pueden acceder a las estadísticas alojadas en la aplicación, ver las suyas propias, las de un equipo o un rival.
- Delegado: Es el administrador y encargado de las estadísticas, facilita su trabajo generando las estadísticas según una hoja de cálculo.

Tecnologías usadas

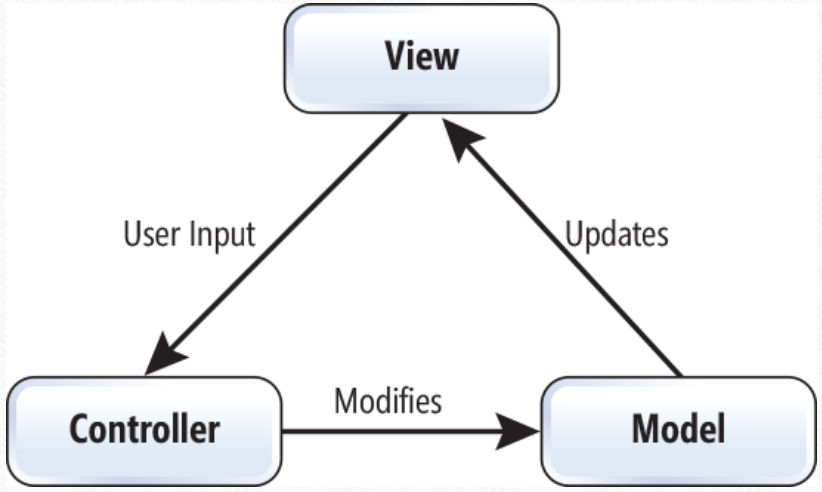
Django

- Framework para desarrollo web usando Python.
- Creado para el periódico World Company de Lawrence (Kansas) en 2005

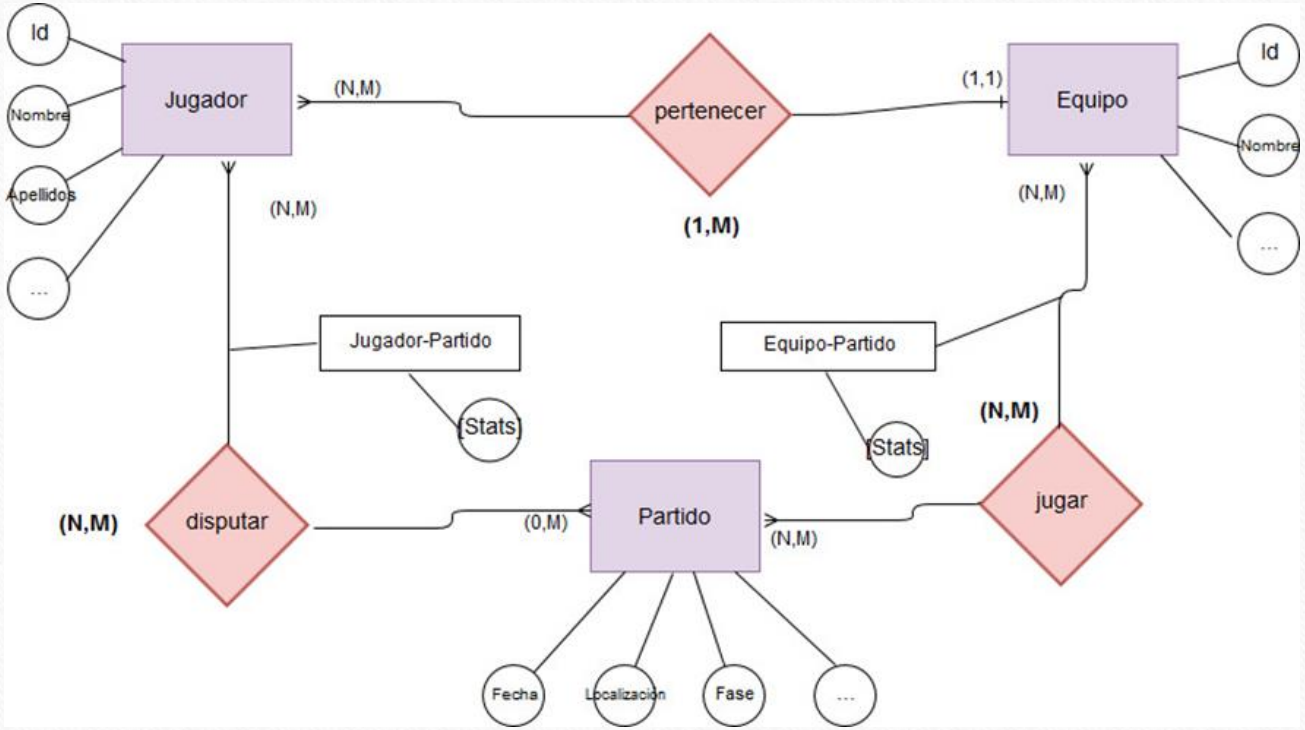


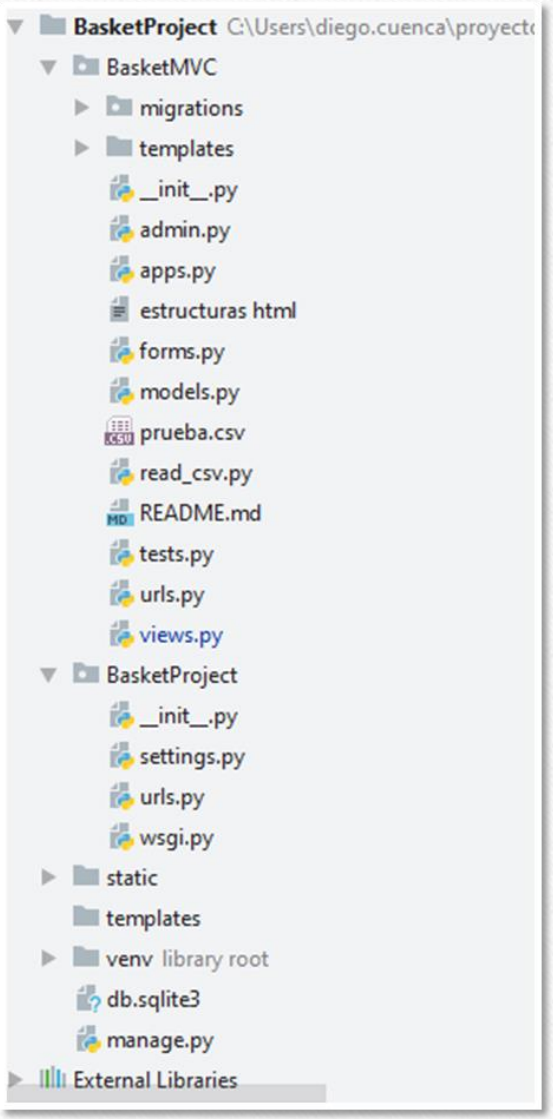
Django: MVT

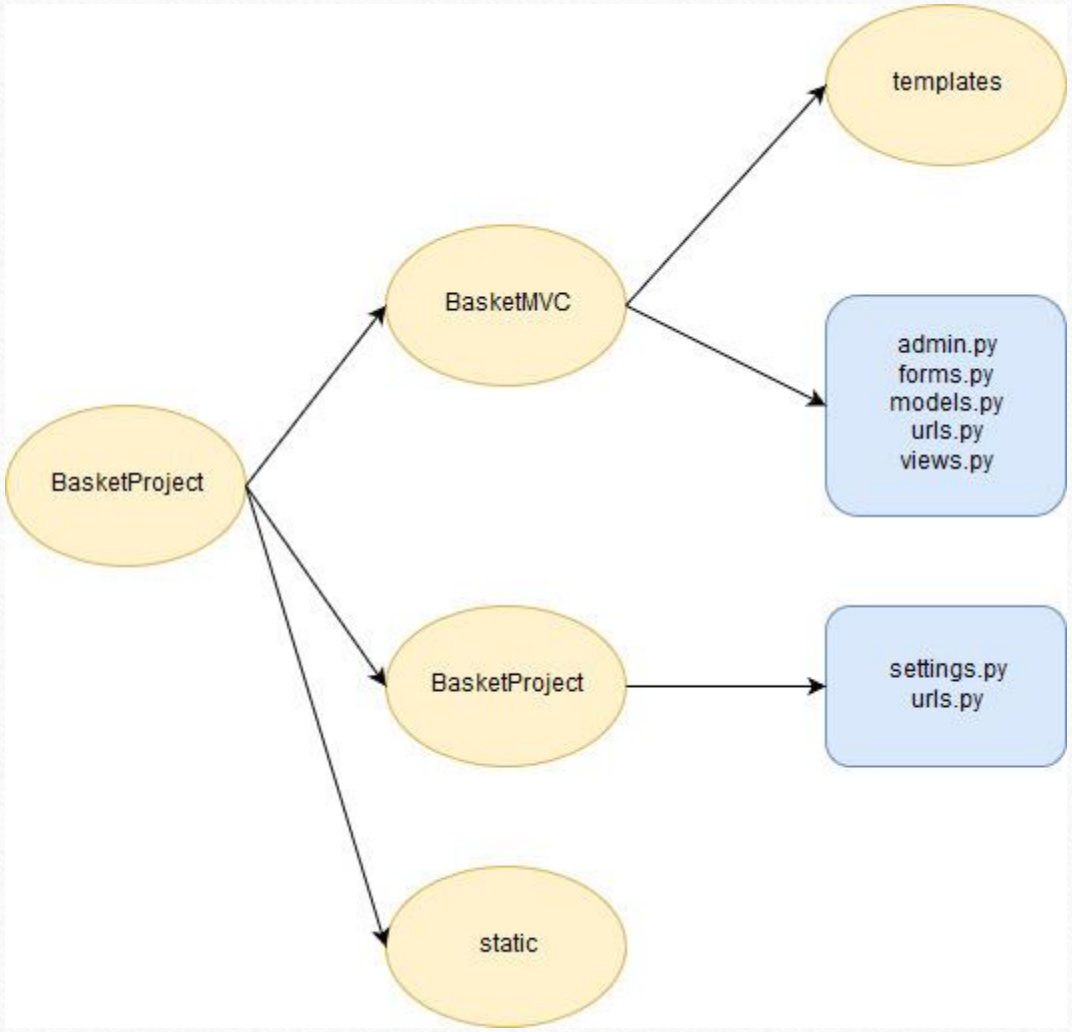
- Similar al patrón MVC.
- Cambia los controladores por “vistas”
- Las vistas son templates.



Análisis


Id
Nombre
Apellidos
...
Jugador
(N,M)
pertenecer
(1,1)
Equipo
Id
Nombre
...
(N,M)
(1,M)
(N,M)
Jugador-Partido
Stats
Equipo-Partido
Stats
(N,M)
(N,M)
disputar
(0,M)
Partido
(N,M)
jugar
Fecha
Localización
Fase
...


BasketProject C:\Users\diego.cuenca\proyecto
BasketMVC
migrations
templates
__init__.py
admin.py
apps.py
estructuras html
forms.py
models.py
prueba.csv
read_csv.py
README.md
tests.py
urls.py
views.py
BasketProject
__init__.py
settings.py
urls.py
wsgi.py
static
templates
venv library root
db.sqlite3
manage.py
External Libraries


templates
BasketMVC
admin.py
forms.py
models.py
urls.py
views.py
BasketProject
BasketProject
settings.py
urls.py
static

Modelos

- Cada clase corresponde a una de nuestras entidades
- Definimos las columnas como atributos de la clase

```
6   class equipo(models.Model):
7       id = models.AutoField(primary_key=True)
8       nombre = models.CharField('Nombre', max_length=30)
9       sede = models.CharField('Dirección', max_length=100)
10
11      # Estadísticas acumuladas
12
13      TC_P = models.IntegerField('TC%',default=0)
14      TC2 = models.IntegerField('TC2',default=0)
15      I_TC2 = models.IntegerField('2PI',default=0)
16      TC2_P = models.IntegerField('TC2%',default=0)
17
18      TC3 = models.IntegerField('TC3',default=0)
19      I_TC3 = models.IntegerField('3PI',default=0)
20      TC3_P = models.IntegerField('TC3%',default=0)
```

URLS y Vistas

```python
urlpatterns = [
    ## Inicio ##
    path('index', views.index, name = 'index'),

    ## Ver jugador ##
    path('jugador', views.mostrar_jugador, name = 'Jugador'),

    ## Ver jugadores ##
    path('mostrar_jugadores', views.Mostrar, name = 'Mostrar'),

    ## Ver listado de equipos ##
    path('equipos', views.mostrar_Equipos, name='Equipos'),

    ## Ver un equipo ##
    path('equipo', views.mostrar_equipo, name='Equipo'),

    ## Ver un partido ##
    path('partido', views.mostrar_partido, name='Partido'),

    ## Ver un listado de partidos ##
    path('partidos', views.mostrar_Partidos, name='Partidos'),

    ## Agregar un partido ##
    path('upload', views.formulario_partido, name='upload_partido'),
```

```python
## Muestra varios jugadores ##
@login_required
def Mostrar(request):
    return render(request, 'jugador.html', {'jugadores': jugador.objects.all()})

## Muestra un único jugador ##
@login_required
def mostrar_jugador(request):
    nombre2 = request.GET['nombre']
    id_equipo = request.GET['equipo']
    apellido2 = request.GET['apellido']
    id = jugador.objects.get(nombre=nombre2, equipo=id_equipo, apellido1__=apellido2).id
    return render(request, 'jugador.html', {
        'jugador': jugador.objects.get(nombre=nombre2, equipo=id_equipo, apellido1__=apellido2),
        'partidos': stats_jugador.objects.filter(id_jugador=id)
    })

## Muestra un equipo ##
@login_required
def mostrar_equipo(request):
    nombre2 = request.GET['nombre']
    id = equipo.objects.get(nombre__nombre2).id
    return render(request, 'equipo.html', {
        'equipo': equipo.objects.get(nombre=nombre2),
        'jugadores': jugador.objects.filter(equipo=id),
        'partidos': partido.objects.filter(equipo1__=id) | partido.objects.filter(equipo2__=id)})

## Muestra listado equipos ##
@login_required
def mostrar_Equipos(request):
    return render(request, 'list_equipos.html', {'equipos': equipo.objects.all()})

## Muestra un listado de partidos ##
@login_required
def mostrar_Partidos(request):
    return render(request, 'list_partidos.html', {'partidos': partido.objects.all()})
```

Agregar Partido

```python
from django import forms
from BasketMVC.models import equipo


## Formulario de añadir partidos ##
class PartidoForm(forms.Form):
    equipo1 = forms.ModelChoiceField(
        queryset = equipo.objects.values_list('nombre', flat=True)
    )

    equipo1.widget.attrs.update({'class': 'form-control'})
    equipo2 = forms.ModelChoiceField(
        queryset = equipo.objects.values_list('nombre', flat=True)
    )
    equipo2.widget.attrs.update({'class': 'form-control'})

    PFASE = 'Primera Fase'
    SFASE = 'Segunda Fase'
    FINAL = 'Fase Final'
    ASCENSO = 'Fase Ascenso'

    FASES = (
        (PFASE, 'Primera Fase'),
        (SFASE, 'Segunda Fase'),
        (FINAL, 'Fase Final'),
        (ASCENSO, 'Fase de Ascenso')
    )

    fase = forms.ChoiceField(choices=FASES)
    fase.widget.attrs.update({'class': 'form-control'})
    file = forms.FileField()
    file.widget.attrs.update({'class': 'form-control'})
```

```python
## Agrega un partido ##
def formulario_partido(request):

    ## Obtenemos la petición y mostramos el formulario ##
    if request.method == 'POST':
        form = PartidoForm(request.POST, request.FILES)
        resultado = upload_partido(request.FILES['file'], request.POST['equipo1'], request.POST['equipo2'], request.POST['fase'])
        return render(request, 'prueba.html', {'resultado': resultado})
    else:
        form = PartidoForm()
        return render(request, 'formulario_partido.html', {'form': form})

```

Templates

```
{% extends 'base.html' %}
{% load static %}

{% block content %}
    <h1> <img src="/static/images/{{jugador.equipo.abrev}}.png" height="100" width="100"></h1>
    <h1>{{jugador.nombre}} {{jugador.apellido1}}</h1>
    <h2> Número {{jugador.dorsal}} de {{jugador.equipo.nombre}}</h2>

    <hr>
    <h3>Estadísticas Acumuladas</h3>
            <table class="table table-stripped">
            <thead>
                <th scope="col">Partidos</th>
                <th scope="col"><abbr title="Puntos anotados">pts</abbr></th>
                <th scope="col"><abbr title="Puntos por partido">p/p</abbr></th>
                <th scope="col"><abbr title="Porcentaje Tiros Campo">TC%</abbr></th>
                <th scope="col"><abbr title="Tiros de 2">TC2</abbr></th>
                <th scope="col"><abbr title="Porcentaje Tiros de 2">%TC2</abbr></th>
                <th scope="col"><abbr title="Tiros de 3">TC3</abbr></th>
                <th scope="col"><abbr title="Porcentaje Tiros de 3">%TC3</abbr></th>
                <th scope="col"><abbr title="Tiros libres">TL</abbr></th>
                <th scope="col"><abbr title="Porcentaje tiros libres">%TL</abbr></th>
            </thead>
            <tbody>
                <tr>
                    <td scope="row">{{ jugador.partidos }}</td>
                    <td>{{ jugador.pts }}</td>
                    <td>{{ jugador.p_p }}</td>
                    <td>{{ jugador.TC_P }}</td>
                    <td>{{ jugador.TC2 }}/{{ jugador.I_TC2 }}</td>
                    <td>{{ jugador.TC2_P }}</td>
                    <td>{{ jugador.TC3 }}/{{ jugador.I_TC3 }}</td>
                    <td>{{ jugador.TC3_P }}</td>
                    <td>{{ jugador.TL }}/{{ jugador.I_TL }}</td>
                    <td>{{ jugador.TL_P }}</td>
                </tr>
            </tbody>
        </table>
        <hr>
```

Posibles Mejoras

- Separar por completo back y front: convirtiendo el back en una API Rest y el front en una app de Angular por ejemplo, que consume dichos servicios.
- Obtener más fuentes de datos: pdf, actas, webscrapping…
- Exportar los datos en informes personalizados según las exigencias del entrenador.
- Crear un panel de administrador propio.

Fin