

# Dashboard Apple Prototype

Juan David Cotacio Sánchez Valery Dayana Triana Garcia

## Resumen—

### I. INTRODUCCIÓN

En el mundo de la tecnología, Apple se destaca como un líder indiscutible, y con nuestro proyecto de Dashboard de Ventas de Apple, hemos combinado la potencia de Python, Django, Pandas, Bootstrap y Plotly para ofrecer una visión integral y dinámica del rendimiento de ventas de esta icónica marca.

#### I-A. Como empezar el prototipo

Para la creación de este prototipo empecé en la instalación de Python, Django, Plotly y Pandas en sistema operativo windows para crear el entorno virtual en el cual iba trabajar mis Dashboard de ventas, además de la creación de mi servidor local con Django donde podemos leer la documentación para crear un Admin para que nuestra información en nuestra DB no sea de fácil acceso.

```
.\Users\nexus\Documents\DashboardVentas>python manage.py runserver localhost:3000
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 06, 2024 - 04:22:36
Django version 5.0.3, using settings 'DashboardVentas.settings'
Starting development server at http://localhost:3000/
Quit the server with CTRL-BREAK.
```

Figura 1. StarUp del Local Server

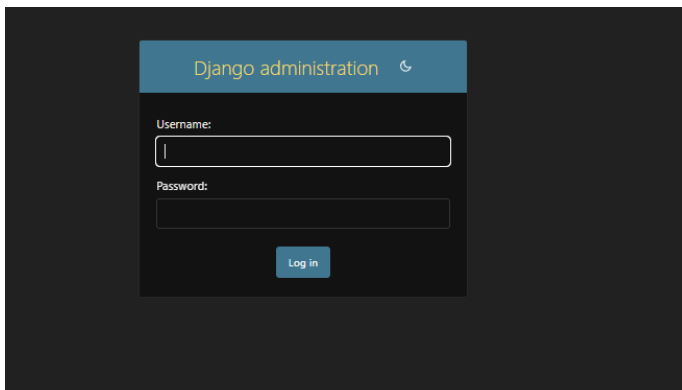


Figura 2. Admin Django

#### I-B. Gestion de Grupos y Usuarios

Después del acceso a mi gestor de datos necesite crear un grupo con ciertos privilegios pues se necesita que la información solo sea administrada por lo encargado de las ventas para después crear usuarios que hagan parte de estos grupos y hagan la gestión de datos.

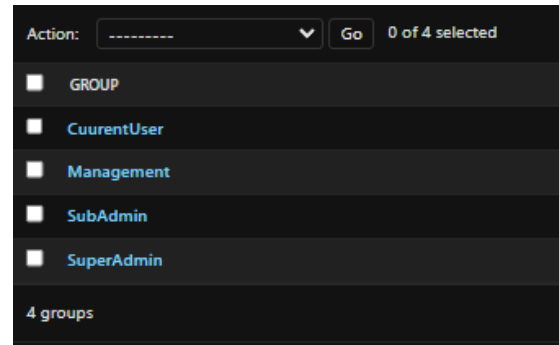


Figura 3. Grupo Django

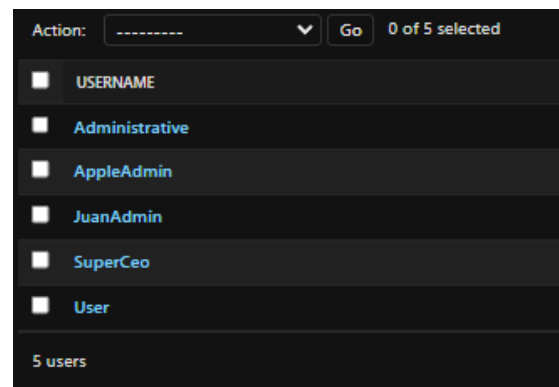


Figura 4. Usuarios Django

#### I-C. Creación de nuestra DB

Para nuestra creación de las ventas realizadas necesitamos empezar a jugar con nuestra base creada al momento de instalar Django, en este caso necesitamos irnos al models.py para poner nuestros tipos de datos y la cantidad de estos mismos decí crear dos tipos de ventas, uno que sean sus ventas mensuales por continentes y otros por productos con sus respectivos tipos de datos.

También en la consola realice la debida migración de mis datos a mi db en este caso realice 3 migraciones pues realice ciertos respectivos cambios para esto es necesario cuando cambiamos cosas de nuestro modelo.

Para la organización de nuestros datos en Django lo que hice fue jugar con nuestro admin.py, en la documentación nos brindan un array listDisplay para poder mostrar de mejor manera nuestros datos.

#### I-D. Implementación del Views

Las views fueron una de las partes más complejas en este proyecto pues es donde básicamente vamos a mostrar los datos

```

ventas > models.py
1  from django.db import models
2
3  class VentasApple(models.Model):
4      month = models.CharField(max_length=120)
5      value = models.IntegerField()
6      ubicacion = models.CharField(max_length= 120)
7
8  class VentasProductos(models.Model):
9      product_type = models.CharField(max_length = 120)
10     amount = models.IntegerField()
11     # Create your models here.
12

```

Figura 5. Modelo Python

```

ventas > admin.py
1  from django.contrib import admin
2  from ventas.models import VentasApple
3  from ventas.models import VentasProductos
4  class VentasAdmin(admin.ModelAdmin):
5      list_display = ["month", "value", "ubicacion"]
6
7  admin.site.register(VentasApple, VentasAdmin)
8
9  class VentasAdmin(admin.ModelAdmin):
10     list_display = ["product_type", "amount"]
11
12  admin.site.register(VentasProductos, VentasAdmin)
13  # Register your models here.
14

```

Figura 7. Admin Python

```

migrations
> __pycache__
> __init__.py
> 0001_initial.py
> 0002_rename_ventas_ventasapple.py
> 0003_ventasproductos.py
> static

```

Figura 6. Migraciones

Figura 8. Datos de la ventas Mensuales

de nuestra DB de manera dinamica, en este punto es donde usamos Plotly para nuestras graficas, para tener una mejor visión de nuestros datos y dashboards complementandose con Pandas. Basicamente en este punto necesitaremos realizar varias instancias de nuestros datos en la base de datos para así traerlos todos y empezar la gestión, para poder empezar a interactuar con nuestros datos y las graficas de Plotly necesitaremos usar DataFrames con los datos que necesitaremos, en este momento hubieron graficas que yo solo queria traer algunos datos entonces se pasaban esos parametros, ademas usaremos HTML y CSS para poder mostrar en una pagina web, por lo que necesitaremos retornar esta información a nuestro HTML.

#### I-E. Integración de nuestro HTML Y CSS

Para poder mostrar un Dashboard con nuestra DB necesitaremos crear nuestros templates en una carpeta Static para poder importarlo en nuestro HTML, hacemos la estructura basica de nuestro HTML, importamos bootstrap para poder darle un mejor diseño, una de las cosas investigativas fue la manera en como importaba mis sources y css al html, para eso usaremos nuestra carpeta static y la cargaremos al inicio y es seguir la estructura que seguiremos por defecto.

#### I-F. Vista General del Prototipo

En la vista general de mi prototipo se logro la gestión de nuestros datos por Django y además de eso usar nuestras herramientas classicas con nuevos frameworks y librerias, como lo son Plotly y Pandas, cargar los datos de manera dinamica.

#### I-G. Instructivo

Para que el juego se pueda disfrutar en terminos general se debe ajustar siempre valores, para que el contenido se adapte

a la pantalla y no se vea desproporcionando se pueda utilizar la combinación de teclas `ctrl + +` en su defecto `ctrl + -`, estos valores son adaptables al gusto del usuario pero para resoluciones 1920 x 1080, el 80 por ciento queda perfectamente anclado.

Este dashboard maneja un navbar para tener un diseño similar al de Apple es interactivo pero para mas comodidad del usuario iniciar el servidor en el puerto 3000 para no tener ningun inconveniente.

Product	Region	Month	Value
iPhone	North America	January	10000
iPhone	North America	February	11000
iPhone	North America	March	12000
iPhone	North America	April	13000
iPhone	North America	May	14000
iPhone	North America	June	15000
iPhone	North America	July	16000
iPhone	North America	August	17000
iPhone	North America	September	18000
iPhone	North America	October	19000
iPhone	North America	November	20000
iPhone	North America	December	21000
iPhone	Europe	January	8000
iPhone	Europe	February	8500
iPhone	Europe	March	9000
iPhone	Europe	April	9500
iPhone	Europe	May	10000
iPhone	Europe	June	10500
iPhone	Europe	July	11000
iPhone	Europe	August	11500
iPhone	Europe	September	12000
iPhone	Europe	October	12500
iPhone	Europe	November	13000
iPhone	Europe	December	13500
iPhone	Asia	January	6000
iPhone	Asia	February	6500
iPhone	Asia	March	7000
iPhone	Asia	April	7500
iPhone	Asia	May	8000
iPhone	Asia	June	8500
iPhone	Asia	July	9000
iPhone	Asia	August	9500
iPhone	Asia	September	10000
iPhone	Asia	October	10500
iPhone	Asia	November	11000
iPhone	Asia	December	11500

Figura 9. Datos de ventas por productos

```

1 from django.shortcuts import render
2 from ventas.models import VentasApple
3 from ventas.models import VentasProductos
4 import plotly.express as px
5 import pandas as pd
6
7 def gestionar(request):
8     sells = VentasApple.objects.all()
9     sellss = VentasProductos.objects.all()
10
11     df = pd.DataFrame(list(sells.values()))
12     dff = pd.DataFrame(list(sellss.values()))
13     dfff = pd.DataFrame(list(sells.values("month", "value")))
14     dffff = pd.DataFrame(list(sellss.values("value")))
15
16     grafico = px.bar(df, x="month", y="value", color="ubicacion", title="Month Sales by Region")
17     torta = px.pie(dff, values='amount', names='product_type', title="Monthly Product Sales")
18     finacial = px.line(dfff, x="month", y="value", title="Profit Lines Per Month")
19     histogram = px.histogram(dffff, x="value", title="Total Sales")
20
21     myhtml = grafico.to_html(full_html=False)
22     grafico_t = torta.to_html(full_html=False)
23     grafico_f = finacial.to_html(full_html=False)
24     grafico_h = histogram.to_html(full_html=False)
25
26     context = {
27         "ventas": sells,
28         "graf": myhtml,
29         "Circular": grafico_t,
30         "Financias": grafico_f,
31         "Histograma": grafico_h,
32     }
33
34     return render(request, "ventas/index.html", context)

```

Figura 10. Implementació Views

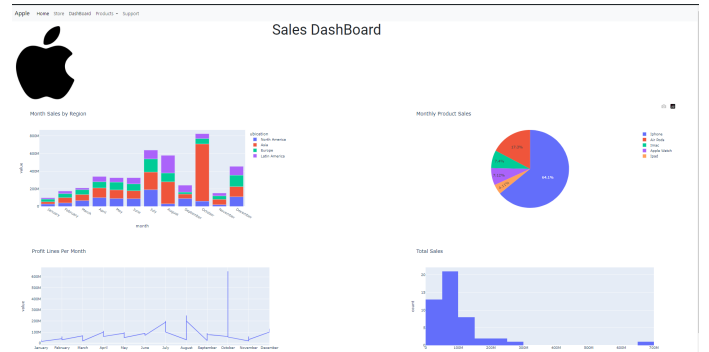


Figura 13. Vista Final Del Prototipo

```

{% load static %}

<doctype html>
<html lang="en">

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>DashboardVentas</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-QJTX2jPEjIS/Sha8U9OFeRpk6Yct0YMD0qNLY728330837mJYBNH4LEWlH" crossorigin="anonymous">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-YpccrV9pccrY3H880W0x559FDVZL5SA5A55NDz0y96k1ds1k1e7N6j1eHz"
  crossorigin="anonymous"></script>
<link rel="stylesheet" href="{% static 'main.css' %}">

```

Figura 11. Llamado Carpeta Static

```

</div>
<div class="container">
  <div class="graf" id="plotly-chart">
    {{graf | safe }}</div>
</div>

<div>
  <div class="graficoT" id="plotly-circle">
    {{Circular | safe}}
  </div>
</div>

<div>
  <div class="graficoF" id="plotly-line">
    {{Financias | safe}}
  </div>
</div>

<div>
  <div class="graficoH" id="plotly-line">
    {{Histograma | safe}}
  </div>
</div>

```

Figura 12. Manera de traer nuestros datos de views al HTML

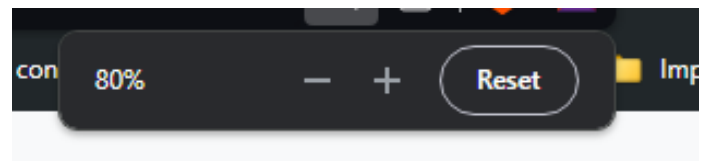


Figura 14. Zoom para resoluciones 1920 x 1080