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 empece 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 facil acceso.

Figura 1. StarUp del Local Server

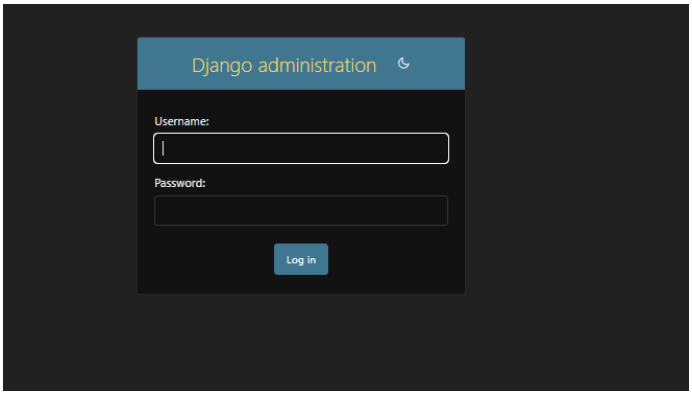

Figura 2. Admin Django

I-B. Gestion de Grupos y Usuarios

Depues del acceso ami gestor de datos necesite crear un grupo con cierto privilegios pues se necesita que la información solo sea administrada por lo encargador de las ventas para despues crear usuarios que hagan parte de estos grupos y hagan la gestion de datos.

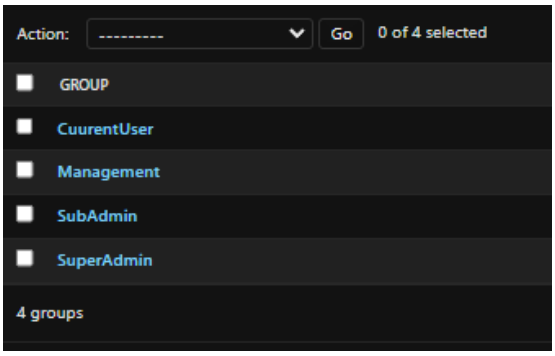

Figura 3. Grupo Django

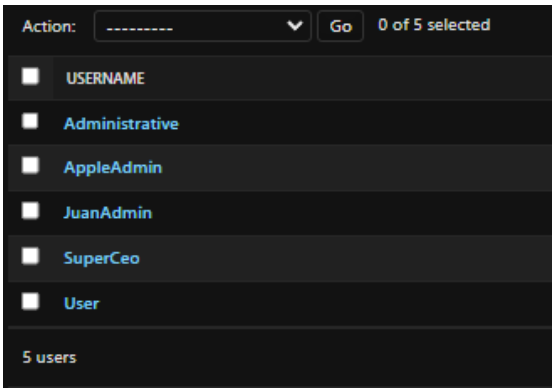

Figura 4. Usuarios Django

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 irno al models.py para poner nuestros tipos de datos y la cantidad de estos mismos decici crear dos tipos de ventas, uno que sean sus ventas mensuales por continentes y otros por porductos con sus respectivos tipos de datos.

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

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

I-D. Implementación del Views

La views fue una de las partes mas complejas en ese proyecto pues es donde basicamente vamos a mostrar los datos

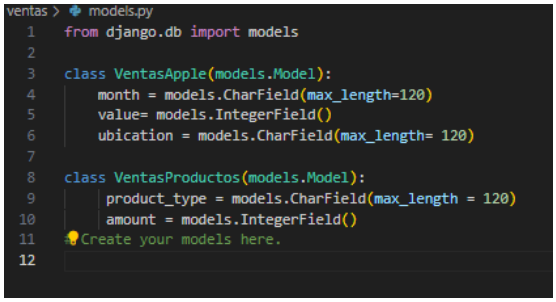

Figura 5. Modelo Python

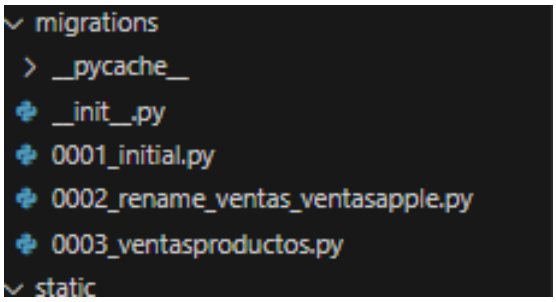

Figura 6. Migraciones

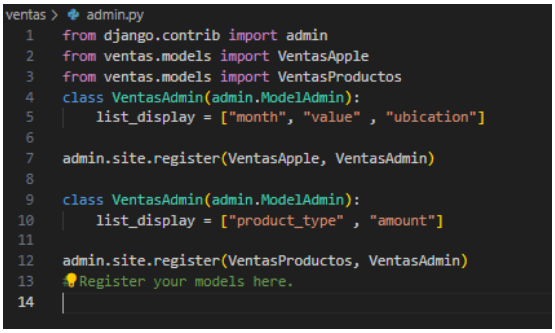

Figura 7. Admin Python

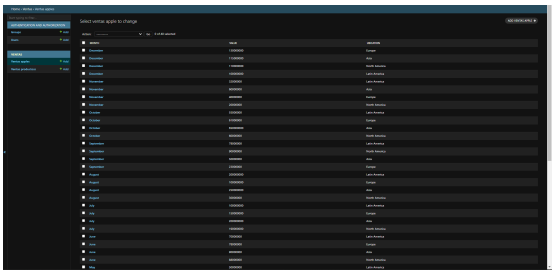

Figura 8. Datos de la ventas Mensuales

de nuestra DB de manera dinamica, en este punto es donde usamos Plotlys para nuestra graficas, para tener una mejor visión de nuestros datos y dashboards complementandose con Pandas. Basicamente en este punto necesitaremos realizar varias intancias de nuestros datos en la base datos para así traerlos todos y empezar la gestion, para poder empezar a interactuar nuestros datos y las graficas de Plotly necesitaremos usar DataFrames con los datos que necesitaremos, en este hubieron graficas que yo solo queria traer algunos datos entonces se pasaba 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 nuestro templates en una carpeta Statica para poder importarlo en nuestro HTML, hacemos la estructura basica de nuestro HTML, importamos boostrap para poder darle un mejor diseño, una de la cosas investigativas fue la manera en como importaba mis sources y css al html, para eso usaremos nuestra carpeta statica y la cargaremos al incicio y es seguir la estructura que seguirimos por defecto.Z

I-F. Vista General del Prototipo

En la vista general de mi protipo se logro la gestion de nuestros datos por Django y además de eso usar nuestras herramientas clasicas 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 desproporcioando se pueda utilizar la combinacion de teclas çtrl + +.º en su defecto çtrl + -", estos valores son adaptables al gusto del usuario pero para resoluciones 1920 x 1080, el 80 porciento queda perfectamente ancajado.

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 incoveniente.

Figura 9. Datos de ventas por productos

```python
from django.shortcuts import render
from ventas.models import VentasApple
from ventas.models import VentasProductos
import plotly.express as px
import pandas as pd

def gestionar(request):
    sells = VentasApple.objects.all()
    sellss = VentasProductos.objects.all()

    df = pd.DataFrame(list(sells.values()))
    dff = pd.DataFrame(list(sellss.values()))
    dfff = pd.DataFrame(list(sells.values("month", "value")))
    dffff = pd.DataFrame(list(sells.values("value")))

    grafico = px.bar(df, x="month", y="value", color="ubication", title="Month Sales by Region")
    torta = px.pie(dff, values='amount', names='product_type', title="Monthly Product Sales")
    financial = px.line(dfff, x='month', y='value', title="Profit Lines Per Month")
    histogram = px.histogram(dffff, x="value", title="Total Sales")

    myhtml = grafico.to_html(full_html=False)
    grafico_t = torta.to_html(full_html=False)
    grafico_f = financial.to_html(full_html=False)
    grafico_h = histogram.to_html(full_html=False)

    context = {
        "ventas": sells,
        "graf": myhtml,
        "Circular": grafico_t,
        "Financias": grafico_f,
        "Histograma": grafico_h,
    }

    return render(request, "ventas/index.html", context)
```

Figura 10. Implementació Views

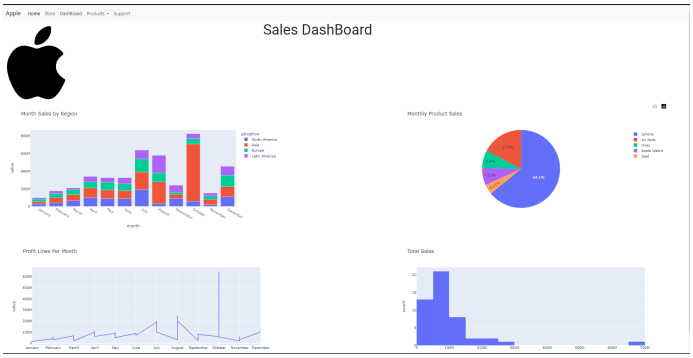

Figura 13. Vista Final Del Prototipo

```
{% 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-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
    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>
```

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

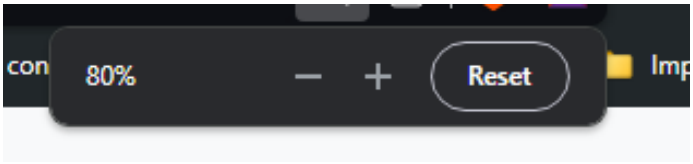

Figura 14. Zoom para resoluciones 1920 x 1080