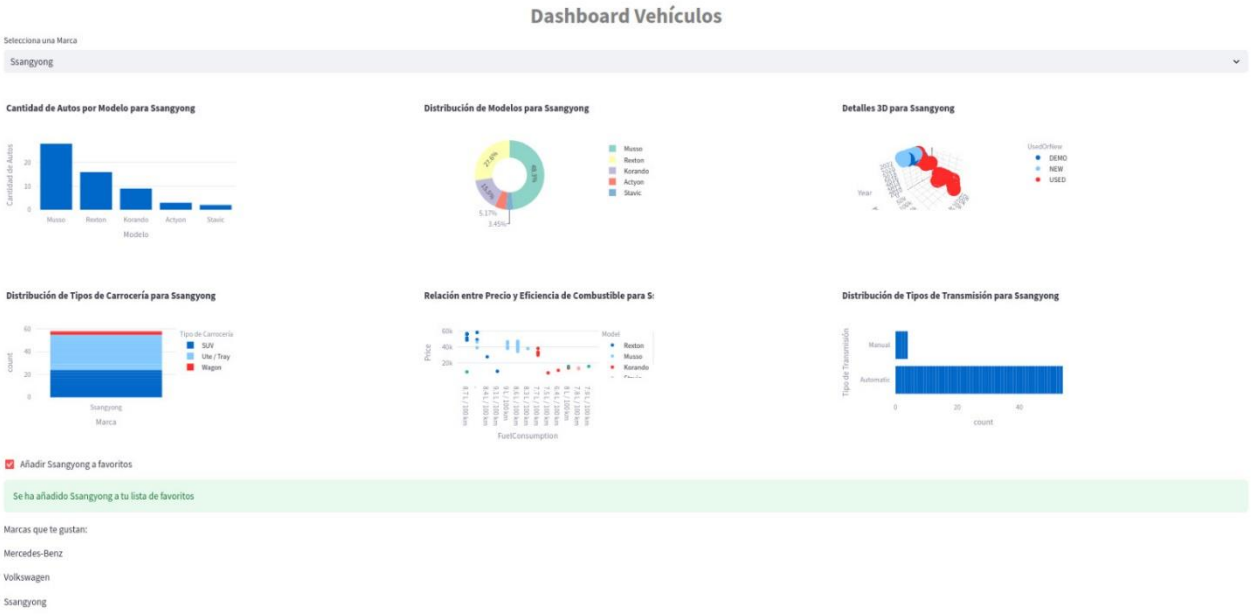
PRACTICA FINAL

Este dashboard de vehículos ofrece una herramienta dinámica para explorar y entender datos relacionados con una marca específica. Es especialmente útil para quien quiera encontrar información entre coches y para comparar ya que proporciona una visión detallada de la distribución de modelos, tipos de carrocería, relaciones entre precio y eficiencia de combustible, y más de cada marca. La funcionalidad de marcar favoritos es perfecta para los usuarios que quieran realizar un seguimiento de sus modelos preferidos.



<u>Cantidad de Autos por Modelo:</u>

Este gráfico proporciona una representación visual de la distribución de coches según sus modelos para la marca seleccionada. Cada barra vertical representa un modelo específico, mientras que la altura de la barra indica la cantidad de coches correspondientes a ese modelo.

Este análisis visual facilita la identificación de modelos más populares, tendencias de ventas y ayuda en la gestión de inventario al ofrecer una visión clara de la composición del parque automotor.

Distribución de modelos:

Este gráfico ofrece una representación visual que destaca la proporción de cada modelo en el rango de vehículos de la marca seleccionada. En este gráfico circular, cada segmento representa un modelo específico, y su tamaño relativo indica la proporción de coches que ese modelo contribuye al total. Este tipo de visualización es útil para identificar de manera rápida y efectiva la participación de cada modelo en el conjunto de vehículos, permitiendo a los usuarios visualizar patrones de popularidad o rareza. En resumen, este gráfico proporciona una perspectiva intuitiva sobre la distribución de modelos.

Dispersión 3D con Precio, Kilómetros y Año del Modelo:

El "Gráfico de Dispersión 3D con Precio, Kilómetros y Año del Modelo" proporciona una representación visual tridimensional de la relación entre el precio, los kilómetros recorridos y el año de fabricación de los coches de la marca seleccionada. Cada punto en el espacio tridimensional representa un coche específico, donde el eje x representa el precio, el eje y representa los kilómetros y el eje z representa el año del modelo. Este tipo de gráfico permite identificar patrones de dispersión, como la posible correlación entre el precio y el año del modelo, así como la dispersión de kilómetros recorridos. Es especialmente útil para visualizar la distribución de características clave en el contexto tridimensional, facilitando la identificación de tendencias de mercado, la evaluación de la relación entre variables y la toma de decisiones informadas en la compra o venta de coches.

Distribución de Tipos de Carrocería:

El "Gráfico de Barras Apiladas para Distribución de Tipos de Carrocería" muestra la distribución de diferentes tipos de carrocería de coches para la marca seleccionada, presentando cada tipo como una barra apilada para resaltar la proporción de cada categoría.

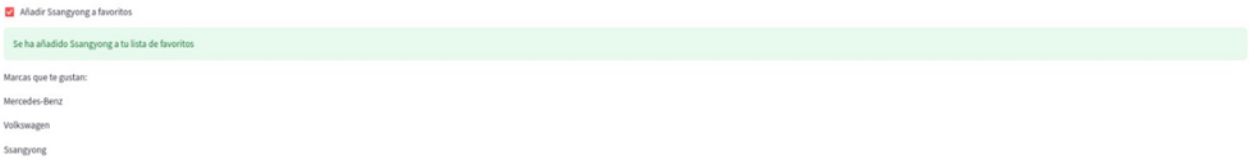
Relación entre Precio y Eficiencia de Consumo de Combustible:

El "Gráfico de Dispersión para Relación entre Precio y Eficiencia de Consumo de Combustible" visualiza la relación entre el precio y la eficiencia de consumo de combustible de coches, proporcionando insights sobre cómo estos dos factores están correlacionados. Este análisis es esencial para compradores y vendedores al considerar la relación entre costos y eficiencia.

Distribución de Tipos de Transmisión:

El "Gráfico de Barras para Distribución de Tipos de Transmisión" presenta la frecuencia de cada tipo de transmisión en el rango de coches, siendo útil para entender las preferencias de transmisión de los clientes y adaptar estrategias de marketing. En conjunto, estos gráficos

ofrecen una visión completa y detallada de la composición y características de coches de la marca seleccionada, siendo esenciales para la toma de decisiones estratégicas y la comprensión del mercado.



En cuanto a las acciones relacionadas con la interacción del usuario, la función "Guardar Marca en Favoritos" implementa una lógica donde la marca seleccionada se registra como favorita, lo que puede implicar operaciones de almacenamiento en una base de datos o en memoria. Por otro lado, la función "Eliminar Marca de Favoritos" permite la eliminación de una marca de la lista de favoritos, involucrando lógica de manejo de datos y la interacción con una base de datos. La "Lista de Favoritos" simplemente recupera y muestra las marcas que han sido registradas como favoritas por el sistema.

FASTAPI

En la carpeta de FastAPI se encuentran tres elementos clave para la práctica:

Dockerfile:

Este archivo se utiliza para construir una imagen de contenedor para la aplicación FastAPI. Los pasos son los siguientes:

1. Se basa en la imagen ‘tiangolo/uvicorn-gunicorn’, optimizada para FastAPI.
2. Crea el directorio ‘/fastapi’ en el contenedor.
3. Copia ‘requirements.txt’ al directorio ‘/fastapi’.
4. Establece el directorio de trabajo como ‘/fastapi’.
5. Instala las dependencias de Python.
6. Copia el código de la aplicación al directorio ‘/fastapi’.
7. Expone el puerto 8000.
8. Define el comando predeterminado para ejecutar la aplicación FastAPI con Uvicorn y Gunicorn en el puerto 8000.

Requirements:

El archivo “requirements.txt” enumera las dependencias necesarias para la aplicación:

1. ‘pydantic==1.10.4’: Validación y serialización de datos.
2. ‘Fastapi’: Framework web rápido para construir APIs.
3. ‘Pandas’: Biblioteca para manipulación y análisis de datos tabulares.

Server:

El archivo ‘server.py’ contiene el código de la aplicación FastAPI:

1. Importa FastAPI y BaseModel de Pydantic.
2. Inicializa una instancia de FastAPI.
3. Define el modelo 'InputData' utilizando la clase BaseModel de Pydantic, con un campo 'name' de tipo cadena.
4. La ruta "/save_brand/" maneja solicitudes POST y guarda la marca proporcionada en la lista de favoritos, devolviendo un mensaje.
5. La ruta "/delete_brand/" maneja solicitudes POST y elimina la marca proporcionada de la lista de favoritos, devolviendo un mensaje.

Este conjunto de archivos facilita la construcción de un contenedor que ejecuta una aplicación FastAPI. La aplicación ahora ofrece funciones para agregar y eliminar marcas de una lista de favoritos, y las rutas "/save_brand/" y "/delete_brand/" realizan estas acciones respectivamente. Tanto el Dockerfile como el archivo de requisitos aseguran que todas las dependencias estén disponibles en el entorno del contenedor para un despliegue sin problemas.

STREAMLIT

En la carpeta de Streamlit vemos los siguientes elementos:

Dashboard:

Este archivo Python llamado "dashboard" utiliza la biblioteca Streamlit para crear un panel de control interactivo basado en datos de vehículos almacenados en un archivo CSV.

- **Propósito:** Crea un panel interactivo para explorar datos de vehículos desde un archivo CSV.
- **Importaciones:** Usa Streamlit, Pandas, Plotly Express, SQLAlchemy, datetime y logging.
- **Diseño y Datos:** Configura el diseño de la aplicación, carga datos desde "Australian_Vehicle_Prices.csv".
- **Selección y Registro:** Permite seleccionar marcas, registra preferencias de usuario en una base de datos SQLite.
- **Visualizaciones:** Utiliza Plotly Express para crear gráficos interactivos.
- **Consulta de Preferencias:** Recupera y muestra marcas preferidas desde la base de datos.

En resumen, este archivo Python utiliza Streamlit para crear un panel de control interactivo que permite al usuario explorar y analizar datos de vehículos, al tiempo que registra las marcas que le gustan en una base de datos SQLite.

Dockerfile:

El Dockerfile es un archivo de configuración esencial en Docker. Define los pasos para construir una imagen de contenedor, asegurando que la aplicación tenga todas las dependencias necesarias. Simplifica la implementación al proporcionar un entorno consistente y portable.

- **Propósito:** Construye un contenedor para la aplicación Streamlit.
- **Base de Imagen:** Utiliza Python 3.10 en su versión liviana (slim).

- **Configuración del Contenedor:**
    - Crea un directorio ‘/streamlit’.
    - Copia ‘requirements.txt’ y todos los archivos al directorio ‘/streamlit’.
    - Establece el directorio de trabajo como ‘/streamlit’.
    - Instala las dependencias especificadas en ‘requirements.txt’.
    - Expone el puerto 8501.
    - Establece el comando predeterminado para ejecutar la aplicación desde "dashboard.py".

Requirements:

El archivo requirements.txt especifica las dependencias de Python para la aplicación Streamlit:

- **Streamlit:** Framework web interactivo.
- **Streamlit-aggrid:** Extensión para integrar Ag-Grid.
- **Requests:** Biblioteca para realizar solicitudes HTTP.
- **Matplotlib:** Herramienta de visualización de gráficos.
- **Seaborn:** Biblioteca para gráficos estadísticos.
- **Plotly:** Biblioteca para gráficos interactivos.
- **Pandas:** Biblioteca de análisis de datos.
- **SQLAlchemy:** Biblioteca para interactuar con bases de datos SQL.

Estas dependencias son esenciales para ejecutar la aplicación Streamlit y garantizar su funcionalidad.

Este conjunto de archivos y configuraciones permite construir un contenedor Docker que ejecuta una aplicación Streamlit incorporada en un entorno FastAPI. La aplicación ofrece un panel interactivo para explorar datos de vehículos, registrando las preferencias del usuario en una base de datos SQLite. El Dockerfile asegura la correcta configuración del entorno, y el archivo ‘requirements.txt’ especifica las dependencias esenciales para la ejecución de la aplicación Streamlit.

DOCKERCOMPOSE

El docker-compose.yml define dos servicios: "fastapi" y "streamlit". El servicio "fastapi" se construye a partir del directorio "fastapi/" y expone el puerto 8000. El servicio "streamlit" se construye a partir del directorio "streamlit/" y depende del servicio "fastapi". Expone el puerto 8501. Los dos servicios comparten una red llamada "deploy_network" y utilizan un volumen llamado "mis_datos" para persistir datos entre contenedores.