eb12f06e-d560-4c9c-9837-aadab006aeca-Copy1

November 6, 2024

1 SPRINT 11 Proyecto integrado 2 (tienda de alimentos)

2 Indice

2.1 Introducción

En el dinámico mundo de las aplicaciones móviles, comprender el comportamiento de los usuarios es fundamental para el éxito de cualquier producto. En este estudio, nos adentramos en el análisis del comportamiento de los usuarios de nuestra aplicación móvil de venta de productos alimenticios. Nuestro objetivo principal es optimizar la experiencia del usuario y aumentar la tasa de conversión, desde la primera interacción hasta la finalización de la compra.

Para lograr esto, realizamos un análisis exhaustivo del embudo de ventas, identificando los puntos críticos donde los usuarios abandonan el proceso de compra. Además, llevamos a cabo un experimento A/A/B para evaluar el impacto de un cambio en el diseño de la interfaz, específicamente en las fuentes utilizadas en toda la aplicación. Al comparar dos grupos de control con un grupo experimental, pudimos determinar si el nuevo diseño de fuente afectaba significativamente el comportamiento de los usuarios.

A través de este análisis, buscamos responder a preguntas clave como: ¿Cuáles son las etapas del embudo de ventas que presentan mayores desafíos para los usuarios? ¿El cambio en las fuentes tiene un impacto positivo o negativo en la tasa de conversión? Los resultados de este estudio nos permitirán tomar decisiones informadas para mejorar la usabilidad de nuestra aplicación y aumentar las ventas.

2.2 Preparación de datos

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
```

```
from scipy.stats import mannwhitneyu, shapiro
```

[2]:
```
df = pd.read_csv("/datasets/logs_exp_us.csv", sep= '\s+', header=0,␣
↪encoding='utf-8', names=['EventName', 'DeviceIDHash', 'EventTimestamp',␣
↪'ExpId'])

df = df.rename(columns={
    'EventName': 'event_name',
    'DeviceIDHash' : 'device_id_hash',
    'EventTimestamp' : 'event_time',
    'ExpId' : 'exp_id'})

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244126 entries, 0 to 244125
Data columns (total 4 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   event_name      244126 non-null  object
 1   device_id_hash  244126 non-null  int64
 2   event_time      244126 non-null  int64
 3   exp_id          244126 non-null  int64
dtypes: int64(3), object(1)
memory usage: 7.5+ MB
```

[3]:
```
df.isna().sum()
```

[3]:
```
event_name        0
device_id_hash    0
event_time        0
exp_id            0
dtype: int64
```

[4]:
```
def cambiar_nombres_grupos(df):
    mapping = {246: 'A1', 247: 'A2', 248: 'B'}

    df['grupo'] = df['exp_id'].map(mapping)

    df.drop('exp_id', axis=1, inplace=True)

    return df

df = cambiar_nombres_grupos(df)
print(df.head())
```

```
event_name       device_id_hash  event_time grupo
```

```
0         MainScreenAppear  4575588528974610257  1564029816    A1
1         MainScreenAppear  7416695313311560658  1564053102    A1
2  PaymentScreenSuccessful  3518123091307005509  1564054127     B
3         CartScreenAppear  3518123091307005509  1564054127     B
4  PaymentScreenSuccessful  6217807653094995999  1564055322     B
```

[5]:
```
df['event_time'] = pd.to_datetime(df['event_time'], unit='s')
df.head()
```

[5]:
```
                event_name       device_id_hash          event_time grupo
0         MainScreenAppear  4575588528974610257 2019-07-25 04:43:36    A1
1         MainScreenAppear  7416695313311560658 2019-07-25 11:11:42    A1
2  PaymentScreenSuccessful  3518123091307005509 2019-07-25 11:28:47     B
3         CartScreenAppear  3518123091307005509 2019-07-25 11:28:47     B
4  PaymentScreenSuccessful  6217807653094995999 2019-07-25 11:48:42     B
```

[6]:
```
df['fecha'] = df['event_time'].dt.date
df.head()
```

[6]:
```
                event_name       device_id_hash          event_time grupo  \
0         MainScreenAppear  4575588528974610257 2019-07-25 04:43:36    A1
1         MainScreenAppear  7416695313311560658 2019-07-25 11:11:42    A1
2  PaymentScreenSuccessful  3518123091307005509 2019-07-25 11:28:47     B
3         CartScreenAppear  3518123091307005509 2019-07-25 11:28:47     B
4  PaymentScreenSuccessful  6217807653094995999 2019-07-25 11:48:42     B

        fecha
0  2019-07-25
1  2019-07-25
2  2019-07-25
3  2019-07-25
4  2019-07-25
```

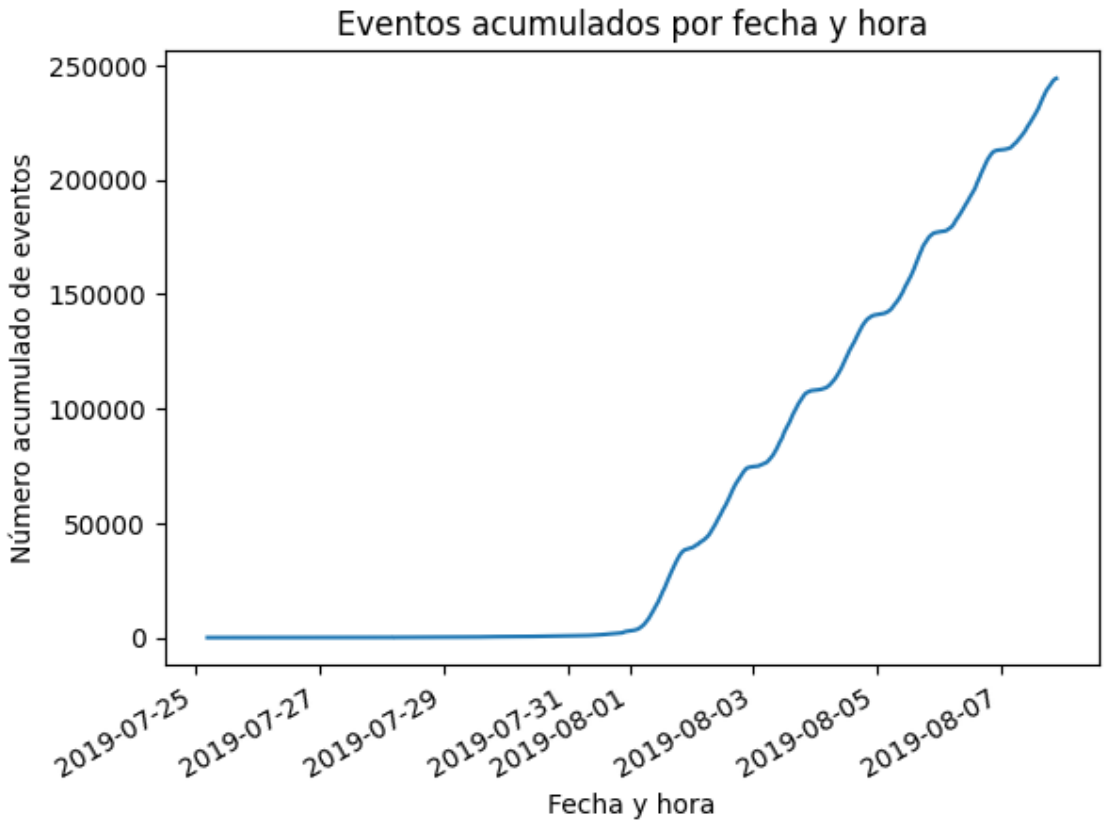
2.3 Estudiar y comprobar los datos

[7]:
```
# 1. Número total de eventos
total_eventos = df.shape[0]
print("Número total de eventos:", total_eventos)

# 2. Número total de usuarios únicos
total_usuarios = df['device_id_hash'].nunique()
print("Número total de usuarios únicos:", total_usuarios)

# 3. Promedio de eventos por usuario
promedio_eventos_por_usuario = total_eventos / total_usuarios
print("Promedio de eventos por usuario:", promedio_eventos_por_usuario)
```

```
Número total de eventos: 244126
Número total de usuarios únicos: 7551
Promedio de eventos por usuario: 32.33028737915508
```

[8]:
```
fecha_minima = df['fecha'].min()
fecha_maxima = df['fecha'].max()

print("Fecha más temprana:", fecha_minima)
print("Fecha más reciente:", fecha_maxima)

df.groupby(df['event_time']).size().cumsum().plot()
plt.xlabel('Fecha y hora')
plt.ylabel('Número acumulado de eventos')
plt.title('Eventos acumulados por fecha y hora')
plt.show()
```

```
Fecha más temprana: 2019-07-25
Fecha más reciente: 2019-08-07
```



La fecha de inicio o minima es el 25 de julio de 2019 y la fecha maxima es el 07 de agosto. un total de 13 dias, pero como se observa en la grafica a partir del 01 de agosto de 2019 es el momento en

el que los datos comienzan a estar completos

[9]:
```
# Agrupar por evento y contar el número de usuarios únicos
usuarios_por_evento = df.groupby('event_name')['device_id_hash'].nunique()

usuarios_por_evento
```

[9]:
```
event_name
CartScreenAppear           3749
MainScreenAppear           7439
OffersScreenAppear         4613
PaymentScreenSuccessful    3547
Tutorial                    847
Name: device_id_hash, dtype: int64
```

El orden de eventos es el siguiente : pagina principal > pagina de ofertas > carrito > pago > tutorial nuestro interes es la cantidad de usuarios unicos que llegan hasta la etapa de compra que en este caso son 3.547. Mientras que a la pagina principal llegan 7.439. estamos hablando que solo el 52% de los usuarios que ingresan a la pagina terminan realizando una compra.

[10]:
```
eventos_totales_por_evento = df.groupby('event_name').size()
eventos_totales_por_evento
```

[10]:
```
event_name
CartScreenAppear           42731
MainScreenAppear          119205
OffersScreenAppear         46825
PaymentScreenSuccessful    34313
Tutorial                    1052
dtype: int64
```

Al calcular la cantidad de usuarios en su totalidad que visitan la pagina. Se puede notar una gran diferencia. Donde al evento 'MainScreenAppear' llegan 119.205. Mientras que a 'PaymentScreen-Successful' llegan solo el 29%. Sin embargo. se puede observar un dato claro de los usuarios que llegan a la pagina principal y los que pasan a la siguiente instancia que es la "pagina de ofertas", hay una diferencia del 58%, por alguna razon una cantidad de usuarios signifcativa solo llega hasta la pagina principal.

para seguir con el estudio, tomaremos todos los eventos menos el "tutorial"

[11]:
```
df = df[df['event_name'] != 'Tutorial']
df['event_name'].unique()
```

[11]:
```
array(['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear',
       'OffersScreenAppear'], dtype=object)
```

[12]:
```
fecha_inicio_a_eliminar = '2019-07-25'
fecha_fin_a_eliminar = '2019-08-01'
```

```
df_filtrado = df[~((df['event_time'] >= fecha_inicio_a_eliminar) &␣
 ↪(df['event_time'] <= fecha_fin_a_eliminar))].reset_index()
df_filtrado
```

[12]:

	index	event_name	device_id_hash	event_time	\
0	2829	MainScreenAppear	3737462046622621720	2019-08-01 00:08:00	
1	2830	MainScreenAppear	3737462046622621720	2019-08-01 00:08:55	
2	2831	OffersScreenAppear	3737462046622621720	2019-08-01 00:08:58	
3	2832	MainScreenAppear	1433840883824088890	2019-08-01 00:08:59	
4	2833	MainScreenAppear	4899590676214355127	2019-08-01 00:10:15	
…	…	…	…	…	
240254	244121	MainScreenAppear	4599628364049201812	2019-08-07 21:12:25	
240255	244122	MainScreenAppear	5849806612437486590	2019-08-07 21:13:59	
240256	244123	MainScreenAppear	5746969938801999050	2019-08-07 21:14:43	
240257	244124	MainScreenAppear	5746969938801999050	2019-08-07 21:14:58	
240258	244125	OffersScreenAppear	5746969938801999050	2019-08-07 21:15:17	

	grupo	fecha
0	A1	2019-08-01
1	A1	2019-08-01
2	A1	2019-08-01
3	A2	2019-08-01
4	A2	2019-08-01
…	…	…
240254	A2	2019-08-07
240255	A1	2019-08-07
240256	A1	2019-08-07
240257	A1	2019-08-07
240258	A1	2019-08-07

[240259 rows x 6 columns]

El periodo que representa realmente los datos es desde el 01 de agosto de 2019 al 07 de agosto del 2019. Se elimino el periodo previo a ese y el evento “tutorial” asi podremos seguir con el analisis.

La diferencia de eventos previo a estos cambios no es muy grande, tomando en cuenta que se excluyeron 6 dias

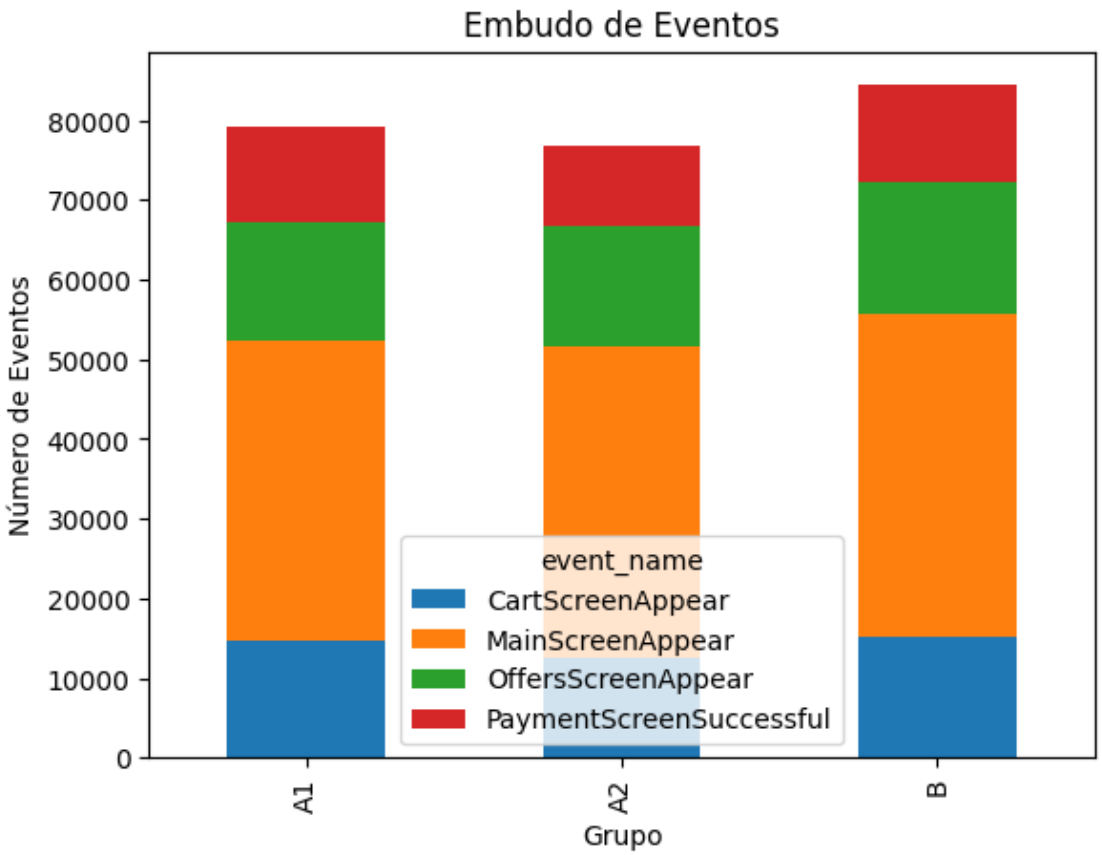
Como a partir del 01-08-2019 los datos comienzan a estar completos y esos 6 dias previos no representan cambios significativos para el analisis. considero que la mejor opcion es excluirlos del estudio.

2.4 Embudo de eventos

[13]:
```
# tabla pivot para contar los eventos por grupo y tipo de evento
pivot_table = df_filtrado.pivot_table(index='grupo', columns='event_name',␣
 ↪aggfunc='size', fill_value=0)
print(pivot_table)
```

```python
columna_inicio = 'MainScreenAppear'
columna_conversion = 'PaymentScreenSuccessful'

# Calcular las tasas de conversión para cada grupo
for grupo in pivot_table.index:
    total_usuarios = pivot_table.loc[grupo, columna_inicio]
    conversiones = pivot_table.loc[grupo, columna_conversion]
    tasa_conversion = (conversiones / total_usuarios) * 100
    print(f"Tasa de conversión para el grupo {grupo}: {tasa_conversion:.2f}%")

pivot_table.plot(kind='bar', stacked=True)
plt.title('Embudo de Eventos')
plt.xlabel('Grupo')
plt.ylabel('Número de Eventos')
plt.show()
```

```
event_name  CartScreenAppear  MainScreenAppear  OffersScreenAppear  \
grupo
A1                     14711             37708               14773
A2                     12456             39123               15182
B                      15198             40600               16395

event_name  PaymentScreenSuccessful
grupo
A1                           11910
A2                           10043
B                            12160
Tasa de conversión para el grupo A1: 31.58%
Tasa de conversión para el grupo A2: 25.67%
Tasa de conversión para el grupo B: 29.95%
```



Todos los grupos muestran un alto número de visualizaciones de la pantalla principal, lo que sugiere una buena atracción inicial de usuarios.Existen diferentes tasas de conversión, a pesar de que todos los grupos muestran un número similar de usuarios agregando productos al carrito, la tasa de conversión a pago varía significativamente, siendo el Grupo B el que presenta un mejor desempeño.

Los grupos A1 y A2 presentan oportunidades para mejorar la tasa de conversión, ya que una parte significativa de los usuarios que agregan productos al carrito no completan la compra.

[14]:
```
# Agrupar por evento y contar el número de usuarios únicos
usuarios_por_evento = df_filtrado.groupby('event_name')['device_id_hash'].
  ↪nunique()

# Ordenar los eventos por el número de usuarios de forma descendente
usuarios_por_evento = usuarios_por_evento.sort_values(ascending=False)

# proporción de usuarios que realizaron cada evento al menos una vez
total_usuarios = df['device_id_hash'].nunique()
proporcion_por_evento = usuarios_por_evento / total_usuarios

print("Usuarios por evento :\n", usuarios_por_evento)
```

```
print("\nUsuarios que realizaron cada evento al menos una vez:\n",␣
 ↪proporcion_por_evento)
```

```
Usuarios por evento :
 event_name
MainScreenAppear           7419
OffersScreenAppear         4593
CartScreenAppear           3734
PaymentScreenSuccessful    3539
Name: device_id_hash, dtype: int64

Usuarios que realizaron cada evento al menos una vez:
 event_name
MainScreenAppear           0.983040
OffersScreenAppear         0.608586
CartScreenAppear           0.494766
PaymentScreenSuccessful    0.468928
Name: device_id_hash, dtype: float64
```

La mayoría de los usuarios inician la aplicación al visualizar la pantalla principal ("MainScreenAppear"), seguida de las ofertas ("OffersScreenAppear"), el carrito de compras y finalmente, la pantalla de pago exitosa.

Un alto porcentaje de usuarios (98.3%) visualiza la pantalla principal, lo que indica que la aplicación está siendo utilizada activamente. La proporción de usuarios que agregan productos al carrito y completan la compra es considerablemente menor, lo que sugiere áreas de oportunidad para mejorar la conversión.

Los datos muestran un embudo de conversión, donde la mayoría de los usuarios inician la aplicación pero solo una fracción menor completa una compra. Es importante analizar los puntos de fricción en el embudo, por ejemplo, como la transición de "OffersScreenAppear" a "CartScreenAppear". ¿Por qué los usuarios no agregan productos al carrito después de ver las ofertas?

La alta frecuencia de "MainScreenAppear" indica un buen nivel de engagement inicial. Sin embargo, es importante analizar si los usuarios regresan a la aplicación con frecuencia y si exploran otras secciones.

La cantidad de usuarios que visualizan las ofertas es significativa un 60,8%, lo que sugiere que las ofertas son relevantes para los usuarios. Sin embargo, la tasa de conversión a compra podría ser mayor.

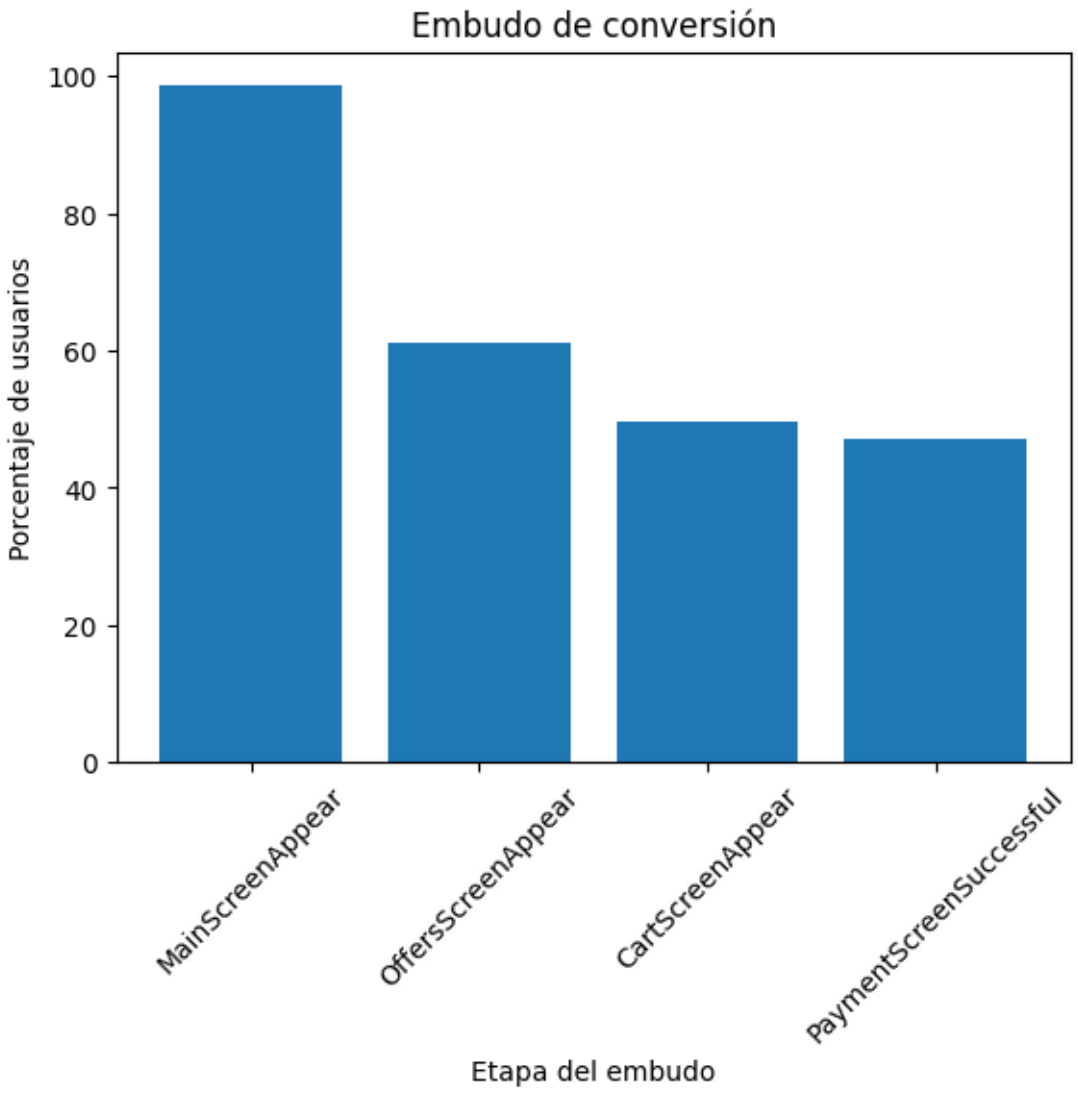
Como recomendacion los datos pueden ayudar a identificar áreas donde se puede mejorar la experiencia del usuario, como simplificar el proceso de compra, ofrecer recomendaciones de productos más personalizadas o mejorar la visibilidad de las ofertas más atractivas.

Ya que tenemos una columna timestamp nos permite saber la secuencia de los eventos dentro de la pagina.

[15]:
```
# Agrupar por usuario y ordenar por tiempo
df_ordenado = df_filtrado.sort_values(['device_id_hash', 'event_time'])
```

```
# Crear una nueva columna para el orden de los eventos
df_ordenado['event_order'] = df_ordenado.groupby('device_id_hash').cumcount()

# DataFrame pivot para visualizar el embudo
embudo = pd.pivot_table(df_ordenado, index='device_id_hash',␣
 ↪columns='event_name', values='event_order', aggfunc='min')

orden = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear',␣
 ↪'PaymentScreenSuccessful']

# porcentaje de usuarios en cada etapa
porcentajes = embudo.count() / embudo.shape[0] * 100
print(porcentajes)

porcentajes = porcentajes.reindex(orden)
plt.bar(porcentajes.index, porcentajes.values)
plt.xlabel('Etapa del embudo')
plt.ylabel('Porcentaje de usuarios')
plt.title('Embudo de conversión')
plt.xticks(rotation=45)
plt.show()
```

```
event_name
CartScreenAppear           49.588313
MainScreenAppear           98.525896
OffersScreenAppear         60.996016
PaymentScreenSuccessful    46.998672
dtype: float64
```



El análisis del porcentaje de usuarios en cada etapa del embudo nos ha proporcionado una visión general del recorrido del usuario en nuestra plataforma. Observamos que el 98.53% de los usuarios llegan a la pantalla principal, lo que indica un alto nivel de tráfico inicial. Sin embargo, el porcentaje de usuarios que avanzan a las siguientes etapas presenta una disminución progresiva.

En particular, llama la atención que solo el 60.99% de los usuarios llegan a la pantalla de ofertas y el 46.99% completa la compra. Estos datos sugieren que existe una pérdida significativa de usuarios entre las diferentes etapas del embudo.

Posibles causas de esta disminución podrían ser:

*Las ofertas presentadas podrían no ser lo suficientemente atractivas o relevantes para los usuarios.

*El proceso de navegación entre las diferentes pantallas puede ser confuso o demasiado complejo.

*El proceso de pago puede ser lento, poco seguro o presentar errores.

[16]:
```
# Ordenar por usuario y tiempo
df_ordenado = df.sort_values(['device_id_hash', 'event_time'])

# Crear una nueva columna para el orden de los eventos
df_ordenado['event_order'] = df_ordenado.groupby('device_id_hash').cumcount()

# Definir las etapas del embudo
etapas = ['MainScreenAppear', 'OffersScreenAppear', 'CartScreenAppear',␣
 ↪'PaymentScreenSuccessful']

# Crear un diccionario para almacenar las proporciones
proporciones = {}

# Iterar sobre todas las posibles transiciones
for i in range(len(etapas) - 1):
    etapa_actual = etapas[i]
    etapa_siguiente = etapas[i+1]

    # Filtrar los eventos que corresponden a la transición
    transicion = df_ordenado[(df_ordenado['event_name'] == etapa_siguiente) &
                             (df_ordenado['event_order'] == 1) &
                             (df_ordenado.
 ↪groupby('device_id_hash')['event_name'].shift() == etapa_actual)]

    # Calcular el total de usuarios en la etapa actual
    total_etapa_actual = df_ordenado[df_ordenado['event_name'] == etapa_actual].
 ↪shape[0]

    # Calcular la proporción
    proporcion = transicion.shape[0] / total_etapa_actual

    # Almacenar la proporción en el diccionario
    proporciones[f"{etapa_actual} -> {etapa_siguiente}"] = proporcion

# Crear un DataFrame con las proporciones
df_proporciones = pd.DataFrame.from_dict(proporciones, orient='index',␣
 ↪columns=['Proporción'])
print(df_proporciones)
```

```
                                          Proporción
MainScreenAppear -> OffersScreenAppear      0.010193
OffersScreenAppear -> CartScreenAppear      0.000705
CartScreenAppear -> PaymentScreenSuccessful    0.001872
```

El análisis de las tasas de conversión ha revelado puntos críticos significativos en nuestro embudo de ventas. Observamos que solo el 1.01% de los usuarios que llegan a la pantalla principal avanzan a la sección de ofertas, mientras que un porcentaje aún menor (0.07%) procede a agregar productos al carrito. La tasa de conversión más baja se encuentra en la transición de carrito a pago, con

apenas un 0.18% de los usuarios completando la compra.

Estos bajos porcentajes sugieren que estamos enfrentando desafíos en las etapas iniciales de nuestro embudo, así como en el proceso de finalización de la compra. Es probable que factores como por ejemplo: diseño poco intuitivo, ofertas poco atractivas, problemas en el proceso de pago estén influyendo en el abandono de los usuarios.

Para abordar estas problemáticas, propongo las siguientes acciones:

*Simplificar el diseño, destacar los beneficios clave y facilitar la navegación hacia las ofertas.

*Asegurarnos de que las ofertas sean atractivas y estén alineadas con las necesidades de los usuarios.

*Reducir el número de pasos, ofrecer múltiples opciones de pago y garantizar la seguridad de la transacción.

*Implementar estrategias de remarketing: Recuperar a los usuarios que abandonaron el carrito mediante emails personalizados y ofertas especiales.

Al implementar estas mejoras, esperamos aumentar significativamente las tasas de conversión en cada etapa del embudo y, en consecuencia, incrementar las ventas.

**¿En qué etapa se pierden más usuarios?

Basándonos en los resultados de ambos códigos, podemos concluir que la mayor pérdida de usuarios se produce entre la etapa de "OffersScreenAppear" (Pantalla de ofertas) y "CartScreenAppear" (Añadir al carrito).

Tasas de conversión: El porcentaje de usuarios que avanzan de la pantalla de ofertas al carrito es extremadamente bajo (0.07%).

Porcentaje de usuarios en cada etapa: Aunque un alto porcentaje de usuarios llega a la pantalla de ofertas, hay una disminución significativa en el número de usuarios que llegan al carrito.

Esto sugiere que existe un obstáculo importante que impide a los usuarios agregar productos a su carrito.

**¿Qué porcentaje de usuarios hace todo el viaje desde su primer evento hasta el pago?

Para responder a esta pregunta, debemos calcular el porcentaje de usuarios que llegan a la etapa final del embudo "PaymentScreenSuccessful".

Basándonos en el código 2, el 46.99% de los usuarios completa todo el viaje desde su primer evento hasta el pago. Esto significa que casi la mitad de los usuarios que inician su recorrido en la plataforma llegan a realizar una compra.

Sin embargo, este porcentaje aún puede considerarse bajo y representa una oportunidad de mejora.

2.5 Resultados de prueba A/A/B

[17]:
```
# cantidad de usuarios únicos en cada grupo
conteo_usuarios_por_grupo = df_ordenado['grupo'].value_counts()

print(conteo_usuarios_por_grupo)
```

```
B     85369
A1    79980
A2    77725
Name: grupo, dtype: int64
```

[18]:
```
df_ordenado.head()
```

[18]:
```
                    event_name    device_id_hash          event_time grupo  \
197263         MainScreenAppear  6888746892508752 2019-08-06 14:06:34    A1
209196         MainScreenAppear  6909561520679493 2019-08-06 18:52:54    A2
209199  PaymentScreenSuccessful  6909561520679493 2019-08-06 18:52:58    A2
209200         CartScreenAppear  6909561520679493 2019-08-06 18:52:58    A2
209201         MainScreenAppear  6909561520679493 2019-08-06 18:52:58    A2

              fecha  event_order
197263  2019-08-06            0
209196  2019-08-06            0
209199  2019-08-06            1
209200  2019-08-06            2
209201  2019-08-06            3
```

[40]:
```
df_ordenado['tuvo_pago_exitoso'] = df_ordenado['event_name'].apply(lambda x: 1␣
 ↪if x == 'PaymentScreenSuccessful' else 0)

usuarios_con_pago_exitoso = df_ordenado.
 ↪groupby('device_id_hash')['tuvo_pago_exitoso'].max().reset_index()

usuarios_con_pago_exitoso = pd.merge(usuarios_con_pago_exitoso,␣
 ↪df_ordenado[['device_id_hash', 'grupo']], on='device_id_hash', how='left')

# grupos A1 y A2
group_a1 = usuarios_con_pago_exitoso[usuarios_con_pago_exitoso['grupo'] ==␣
 ↪'A1']['tuvo_pago_exitoso']
group_a2 = usuarios_con_pago_exitoso[usuarios_con_pago_exitoso['grupo'] ==␣
 ↪'A2']['tuvo_pago_exitoso']

# Prueba de Shapiro-Wilk para normalidad
statistic_a1, p_value_a1 = shapiro(group_a1)
statistic_a2, p_value_a2 = shapiro(group_a2)

print(f"Prueba de Shapiro-Wilk para A1: statistic = {statistic_a1}, p-value =␣
 ↪{p_value_a1}")
print(f"Prueba de Shapiro-Wilk para A2: statistic = {statistic_a2}, p-value =␣
 ↪{p_value_a2}")

# Selección de la prueba estadística
if p_value_a1 > 0.01 and p_value_a2 > 0.01:
```

```
    # Distribución normal: Prueba t de Student
    t_statistic, p_value = ttest_ind(group_a1, group_a2)
    print(f"Prueba t de Student: t-statistic = {t_statistic}, p-value =␣
 ↪{p_value}")
else:
    # Distribución no normal: Prueba de Mann-Whitney U
    U_statistic, p_value = mannwhitneyu(group_a1, group_a2)
    print(f"Prueba de Mann-Whitney U: U-statistic = {U_statistic}, p-value =␣
 ↪{p_value}")

alpha = 0.01
if p_value < alpha:
    print("Hay una diferencia estadísticamente significativa entre A1 y A2.")
else:
    print("No hay una diferencia estadísticamente significativa entre A1 y A2.")
```

```
Prueba de Shapiro-Wilk para A1: statistic = 0.5397093296051025, p-value = 0.0
Prueba de Shapiro-Wilk para A2: statistic = 0.5624679327011108, p-value = 0.0
Prueba de Mann-Whitney U: U-statistic = 3202559910.0, p-value =
2.1575616083158573e-42
Hay una diferencia estadísticamente significativa entre A1 y A2.
```

Para este análisis buscaba determinar si existe una diferencia estadísticamente significativa en la tasa de conversión a pago entre los grupos A1 y A2. Donde se encontro que Si hay una diferencia. Quiere decir que son mas los usuarios del grupo A1 que completan compras dentro de la aplicacion

[20]:
```
# datos para los grupos A1 y A2
df_A1_A2 = df[df['grupo'].isin(['A1', 'A2'])]


eventos_por_grupo = df_A1_A2.groupby(['grupo', 'event_name']).size().
 ↪reset_index(name='cantidad')

# total de usuarios por grupo
total_usuarios_por_grupo = df_A1_A2.groupby('grupo').size().
 ↪reset_index(name='total_usuarios_grupo')


resultados = pd.merge(eventos_por_grupo, total_usuarios_por_grupo, on='grupo')
resultados['proporcion'] = resultados['cantidad'] /␣
 ↪resultados['total_usuarios_grupo']

# evento más popular en cada grupo
eventos_mas_populares = resultados.groupby('grupo').apply(lambda x: x.
 ↪loc[x['cantidad'].idxmax()])
```

```
grupo_A1_popular = eventos_mas_populares.loc['A1', 'proporcion']
grupo_A2_popular = eventos_mas_populares.loc['A2', 'proporcion']

# Verificar normalidad para el grupo A1
stat, p = shapiro(resultados.loc[resultados['grupo'] == 'A1', 'proporcion'])
print(f"Prueba de Shapiro-Wilk para A1: statistic = {stat}, p-value = {p}")

# Verificar normalidad para el grupo A2
stat, p = shapiro(resultados.loc[resultados['grupo'] == 'A2', 'proporcion'])
print(f"Prueba de Shapiro-Wilk para A2: statistic = {p}")


u_statistic, p_value = stats.mannwhitneyu(grupo_A1_popular, grupo_A2_popular)

print("Evento más popular en A1:", eventos_mas_populares.loc['A1',␣
 ↪'event_name'])
print("Evento más popular en A2:", eventos_mas_populares.loc['A2',␣
 ↪'event_name'])
print("Proporción en A1:", grupo_A1_popular)
print("Proporción en A2:", grupo_A2_popular)
print("p-valor de la prueba de Mann-Whitney U:", p_value)
```

```
Prueba de Shapiro-Wilk para A1: statistic = 0.7255407571792603, p-value =
0.022166257724165916
Prueba de Shapiro-Wilk para A2: statistic = 0.0590088926255703
Evento más popular en A1: MainScreenAppear
Evento más popular en A2: MainScreenAppear
Proporción en A1: 0.47863215803950987
Proporción en A2: 0.5109038275972981
p-valor de la prueba de Mann-Whitney U: 1.0
```

El p-valor obtenido de la prueba de Mann-Whitney U es de 1.0. Un p-valor tan alto indica que no hay evidencia suficiente para rechazar la hipótesis nula. En otras palabras, no podemos concluir que exista una diferencia estadísticamente significativa entre las proporciones de los grupos A1 y A2 en cuanto al evento "MainScreenAppear".

[41]:
```
def analizar_eventos(df):

    # agrupar por grupo y evento
    eventos_por_grupo = df.groupby(['grupo', 'event_name']).size().
 ↪reset_index(name='total_usuarios')

    # total de usuarios por grupo
    total_usuarios_por_grupo = df.groupby('grupo').size().
 ↪reset_index(name='total_usuarios_grupo')
```

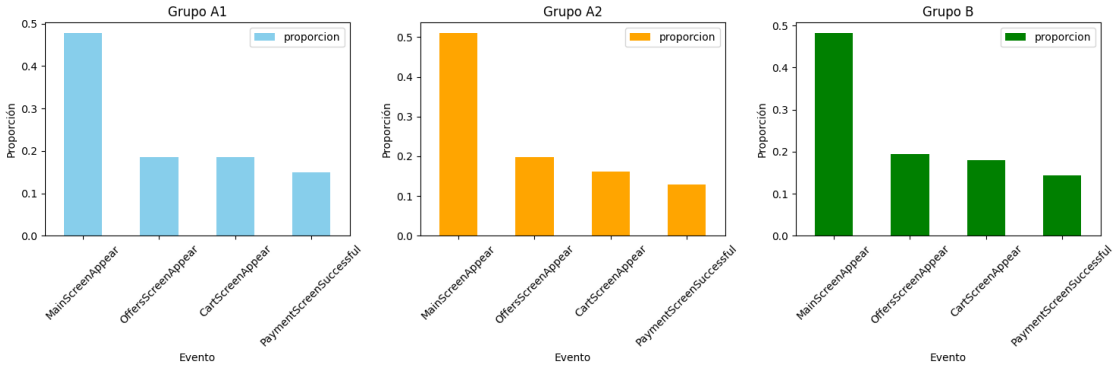
```
    resultados = pd.merge(eventos_por_grupo, total_usuarios_por_grupo,␣
 ↪on='grupo')
    resultados['proporcion'] = resultados['total_usuarios'] /␣
 ↪resultados['total_usuarios_grupo']

    return resultados


resultados_todos_eventos = analizar_eventos(df_ordenado)
resultados_todos_eventos
```

[41]:

	grupo	event_name	total_usuarios	total_usuarios_grupo	proporcion
0	A1	CartScreenAppear	14819	79980	0.185284
1	A1	MainScreenAppear	38281	79980	0.478632
2	A1	OffersScreenAppear	14910	79980	0.186422
3	A1	PaymentScreenSuccessful	11970	79980	0.149662
4	A2	CartScreenAppear	12570	77725	0.161724
5	A2	MainScreenAppear	39710	77725	0.510904
6	A2	OffersScreenAppear	15344	77725	0.197414
7	A2	PaymentScreenSuccessful	10101	77725	0.129958
8	B	CartScreenAppear	15342	85369	0.179714
9	B	MainScreenAppear	41214	85369	0.482775
10	B	OffersScreenAppear	16571	85369	0.194110
11	B	PaymentScreenSuccessful	12242	85369	0.143401

[37]:
```
# grafica de proporciones de eventos por grupo
resultados = analizar_eventos(df_ordenado)

resultados = resultados.sort_values(by='event_name', key=lambda x: x.
 ↪map({'MainScreenAppear': 0, 'OffersScreenAppear': 1, 'CartScreenAppear': 2,␣
 ↪'PaymentScreenSuccessful':3}))

# Filtrar los resultados para cada grupo
```

```
grupos = resultados['grupo'].unique()

colores = ['skyblue', 'orange', 'green']

fig, axes = plt.subplots(nrows=1, ncols=len(grupos), figsize=(15, 5))

for i, grupo in enumerate(grupos):
    resultados_grupo = resultados[resultados['grupo'] == grupo]
    resultados_grupo.plot(x='event_name', y='proporcion', kind='bar',␣
 ↪ax=axes[i], color=colores[i])
    axes[i].set_title(f'Grupo {grupo}')
    axes[i].set_xlabel('Evento')
    axes[i].set_ylabel('Proporción')
    axes[i].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```



El objetivo de este análisis fue comparar el comportamiento de los usuarios en diferentes grupos, evaluando la proporción de usuarios que completaron cada etapa del proceso de compra.

Los resultados muestran que el evento más popular en todos los grupos fue la visita a la página principal. Sin embargo, al analizar cada evento en detalle, se observaron diferencias significativas entre los grupos.

Página principal: Si bien la mayoría de los usuarios visitaron la página principal en todos los grupos, se observaron pequeñas variaciones en las proporciones.

Página de ofertas: El grupo A2, seguido del grupo B, mostraron una mayor proporción de usuarios que visitaron la página de ofertas, lo que sugiere una mayor sensibilidad a las promociones en este grupo.

Carrito: El grupo A1 presentó la proporción más alta de usuarios que agregaron productos al carrito, indicando una mayor intención de compra en este segmento.

Pago: Los resultados revelaron que el grupo A1 tuvo la tasa de conversión más alta, es decir, una

mayor proporción de usuarios completaron la compra.

En conclusión, los resultados sugieren que existen diferencias significativas en el comportamiento de los usuarios entre los diferentes grupos. Sim embargo, las diferencias no son tan grandes como para afirmar que el cambio de fuente en toda la aplicaion generen disgusto en los usuarios.

[23]:
```
# diferencias estadisticamente significativas
evento_seleccionado = "PaymentScreenSuccessful"


grupo_B_proporcion =␣
 ↪resultados_todos_eventos[(resultados_todos_eventos['grupo'] == 'B') &

                                                 ␣
 ↪(resultados_todos_eventos['event_name'] ==␣
 ↪evento_seleccionado)]['proporcion'].values[0]
grupo_A1_proporcion =␣
 ↪resultados_todos_eventos[(resultados_todos_eventos['grupo'] == 'A1') &

                                                 ␣
 ↪(resultados_todos_eventos['event_name'] ==␣
 ↪evento_seleccionado)]['proporcion'].values[0]

U1, p = stats.mannwhitneyu(grupo_B_proporcion, grupo_A1_proporcion)

print(f"Evento: {evento_seleccionado}")
print(f"Proporción en B: {grupo_B_proporcion}")
print(f"Proporción en A1: {grupo_A1_proporcion}")
print(f"p-valor: {p}")
```

```
Evento: PaymentScreenSuccessful
Proporción en B: 0.14340100036312947
Proporción en A1: 0.14966241560390098
p-valor: 1.0
```

Al comprar proporciones la mejor opcion es la prueba Mann Whitney U. Y como resultado obtuvimos que No existe una diferencia estadísticamente significativa entre los grupos B y A1 en términos de la proporción de usuarios que completan el proceso de pago. Esto sugiere que el cambio en las fuentes no ha tenido un impacto significativo en la tasa de conversión de pagos.

2.6 Conclusion general

El análisis realizado sobre el comportamiento de los usuarios en la aplicación ha revelado patrones interesantes y oportunidades de mejora. Aunque el cambio en las fuentes no ha demostrado tener un impacto estadísticamente significativo en la tasa de conversión a pago, se han identificado otros factores clave que afectan el recorrido del usuario a través del embudo de ventas.

Embudo de conversión: Existe una clara disminución en el número de usuarios a medida que avanzan por el embudo, especialmente en la transición de la página de ofertas al carrito y del carrito al pago.

Cambio de las fuentes: El cambio de fuente no parece ser el factor determinante en la tasa de

conversión, lo que sugiere que otros elementos del diseño o la experiencia del usuario podrían estar influyendo más.

Oportunidades de mejora: Se han identificado varios puntos del embudo donde se pueden implementar mejoras para aumentar la tasa de conversión, como optimizar el diseño de las páginas de ofertas, simplificar el proceso de pago y personalizar las recomendaciones de productos.

Recomendaciones:

*Optimizar el embudo de ventas: Centrarse en mejorar la experiencia del usuario en las etapas donde se produce una mayor pérdida de usuarios, especialmente en la transición de la página de ofertas al carrito.

*Personalización: Implementar estrategias de personalización para ofrecer a los usuarios recomendaciones de productos más relevantes y atractivas.

*Análisis de cohortes: Realizar un seguimiento del comportamiento de los usuarios a lo largo del tiempo para identificar patrones y tendencias.

*Segmentación de usuarios: Segmentar a los usuarios en función de características demográficas y de comportamiento para adaptar la experiencia del usuario.

¿Es recomendable continuar con el test A/A/B?

Dado que el cambio de fuente no ha mostrado un impacto significativo en la tasa de conversión, podría no ser prioritario continuar con este test en particular.

Sugerencia:

En lugar de centrarse en el cambio de fuente, se podrían realizar pruebas A/B en otros elementos del diseño, como la disposición de los elementos en la página, los botones de llamada a la acción o el lenguaje utilizado en las descripciones de los productos.

El análisis realizado ha proporcionado una visión valiosa sobre el comportamiento de los usuarios y ha identificado áreas de mejora. Al implementar las recomendaciones mencionadas anteriormente, se puede aumentar significativamente la tasa de conversión y mejorar la experiencia del usuario en general.