Цель исследования: выделите группы пользователей мобильного приложения "Ненужные вещи"

Описание данных:

Датасет содержит данные о событиях, совершенных в мобильном приложении "Ненужные вещи". В нем пользователи продают свои ненужные вещи, размещая их на доске объявлений. В датасете содержатся данные пользователей, впервые совершивших действия в приложении после 7 октября 2019 года. 1 retention rate, 2 время, проведённое в приложении, 3 частота действий, 4 конверсия в целевое действие — просмотр контактов.Проведем исследовательский анализ данных, сегментируем пользователей на основе действий,проверьте статистические гипотезы

Шаг 1: загрузим данные и изучим общую информацию о них

- Изучим общую информацию о датасете
- Что можно сказать о каждом столбце каждой таблицы?
- Каких типов данные присутствуют в столбцах?
- Сколько пользователей представлено?
- Какие есть источники, с которых пользователи установили приложение?
- Какие виды действий пользователей существуют?

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

In [2]:
```
dataset = pd.read_csv('/Users/Olgaolga/Desktop/mobile_dataset 2.csv')
```

In [3]:
```
sourses = pd.read_csv('/Users/Olgaolga/Desktop/mobile.csv')
```

In [4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74197 entries, 0 to 74196
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   event.time  74197 non-null  object
 1   event.name  74197 non-null  object
 2   user.id     74197 non-null  object
dtypes: object(3)
memory usage: 1.7+ MB
```

In [5]:
```
sourses.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4293 entries, 0 to 4292
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   userId  4293 non-null   object
 1   source  4293 non-null   object
dtypes: object(2)
memory usage: 67.2+ KB
```

In [6]:
```
dataset.head()
```

Out[6]:

	event.time	event.name	user.id
0	2019-10-07 00:00:00.431357	advert_open	020292ab-89bc-4156-9acf-68bc2783f894
1	2019-10-07 00:00:01.236320	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
2	2019-10-07 00:00:02.245341	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c
3	2019-10-07 00:00:07.039334	tips_show	020292ab-89bc-4156-9acf-68bc2783f894
4	2019-10-07 00:00:56.319813	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c

In [7]:
```
sourses.head()
```

Out[7]:

	userId	source
0	020292ab-89bc-4156-9acf-68bc2783f894	other
1	cf7eda61-9349-469f-ac27-e5b6f5ec475c	yandex
2	8c356c42-3ba9-4cb6-80b8-3f868d0192c3	yandex
3	d9b06b47-0f36-419b-bbb0-3533e582a6cb	other
4	f32e1e2a-3027-4693-b793-b7b3ff274439	google

In [8]:
```
dataset.describe()
```

Out[8]:

	event.time	event.name	user.id
count	74197	74197	74197
unique	74197	16	4293
top	2019-10-07 00:00:00.431357	tips_show	cb36854f-570a-41f4-baa8-36680b396370
freq	1	40055	478

In [9]:
```
dataset.rename(columns = {'event.time':'event_time', 'event.name':'event_
```

In [10]:
```
sourses.rename(columns = {'userId':'user_id'},inplace = True)
```

In [11]:
```
sourses.describe()
```

Out[11]:

	user_id	source
count	4293	4293
unique	4293	3
top	020292ab-89bc-4156-9acf-68bc2783f894	yandex
freq	1	1934

In [12]:
```
sourses['source'].unique()
```

Out[12]:
```
array(['other', 'yandex', 'google'], dtype=object)
```

In [13]:
```
dataset['event_name'].unique()
```

Out[13]:
```
array(['advert_open', 'tips_show', 'map', 'contacts_show', 'search_4',
       'search_5', 'tips_click', 'photos_show', 'search_1', 'search_2',
       'search_3', 'favorites_add', 'contacts_call', 'search_6',
       'search_7', 'show_contacts'], dtype=object)
```

Выводы шаг 1:

Мы изучили общую информацию о датасете. Что можно сказать о каждом столбце каждой таблицы? В каждом столбце обеих таблиц содержатся данные типа object. В таблице dataset содержится информация о времени события, совершенного пользователем, наименование события, идентификатор пользователя. В sourses содержится информация об идентификаторах пользоавтелей и о источниках. В таблицах представлено 4293 уникальных пользователей. Существуют следующие источники, с которых пользователи установили приложение:'other' (другие), 'yandex', 'google'. Также представлена информация о действиях пользователей, таких как: advert_open — открыл карточки объявления, photos_show — просмотрел фотографий в объявлении, tips_show — увидел рекомендованные объявления, tips_click — кликнул по рекомендованному объявлению, contacts_show и show_contacts — посмотрел номер телефона, contacts_call — позвонил по номеру из объявления, map — открыл карту объявлений, search_1 — search_7 — разные действия, связанные с поиском по сайту, favorites_add — добавил объявление в избранное. Также на шаге 1 я поменяла названия столбцов для удобства использования.

Шаг 2: выполним предобработку данных

- Изучим, есть ли дубликаты в данных
- Проверим наличие пропусков
- Заменим пропуски, если это требуется

- Посмотрим, присутствуют ли выбросы и аномальные значения

In [14]:
```
dataset.duplicated().sum()
```

Out[14]:
```
0
```

In [15]:
```
sourses.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
dataset.isna().sum()
```

Out[16]:
```
event_time    0
event_name    0
user_id       0
dtype: int64
```

In [17]:
```
sourses.isna().sum()
```

Out[17]:
```
user_id    0
source     0
dtype: int64
```

In [18]:
```
dataset['event_time'] = pd.to_datetime(dataset['event_time'])
```

In [19]:
```
dataset.groupby('event_name')['event_time'].count().sort_values(ascending
```

Out[19]:
```
event_name
tips_show        40055
photos_show      10012
advert_open       6164
contacts_show     4450
map               3881
search_1          3506
favorites_add     1417
search_5          1049
tips_click         814
search_4           701
contacts_call      541
search_3           522
search_6           460
search_2           324
search_7           222
show_contacts       79
Name: event_time, dtype: int64
```

In [20]:
```
dataset['event_name'] = dataset['event_name'].replace('contacts_show', 's
```

Выводы шаг 2:

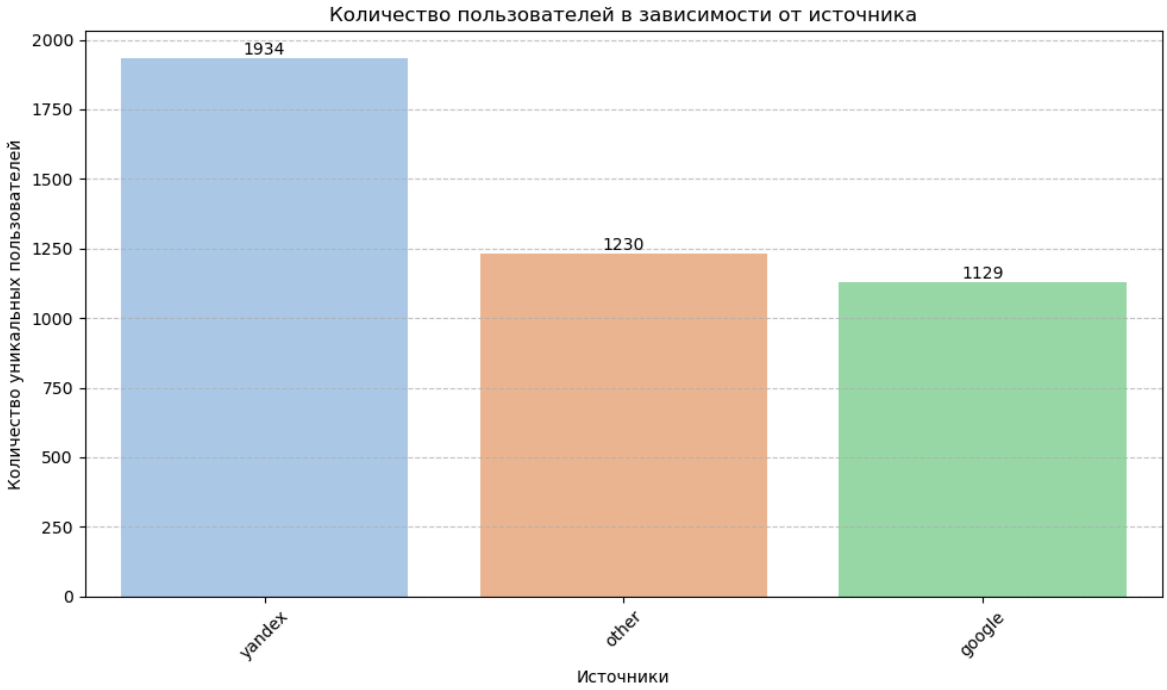
Дубликаты в таблицах отсутствуют, пропусков также нет. Я перевела значения столбца с временем события к типу datetime для удобства обращения с датой и временем. Также можно заметить, что в dataset присутствовали действия contacts_show и show_contacts, имеющие один и тот же смысл. для удобства я объединила их подназванием 'show_contacts'.

Шаг 3: проведем исследовательский анализ данных

- Какие источники являются наиболее популярными? Наименее популярными?
- Как распределены пользователи по действиям? Сколько пользователей теряется после каждого этапа?
- Сколько времени пользователь проводит на каждом действии?
- Определим конверсию в целевое действие — просмотр контактов
- Посчитаем retention rate
- найдем частоту совершения событий
- Определим время, проводимое пользователем в приложении

Изучим популярность источников:

In [21]:
```
sourses_counts = sourses.groupby('source')['user_id'].nunique().sort_valu
plt.figure(figsize=(10, 6))
sns.barplot(x='source', y='sourses_count', data=sourses_counts, palette='
plt.xlabel('Источники')
plt.ylabel('Количество уникальных пользователей')
plt.title('Количество пользователей в зависимости от источника')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
for index, value in enumerate(sourses_counts['sourses_count']):
    plt.text(index, value, f'{value}', ha='center', va='bottom')
plt.tight_layout()
plt.show()
```



Наиболее популярным является Яндекс, затем прочие источники, на третьем месте находится Гугл

In [22]:
```
sourses_counts
```

Out[22]:

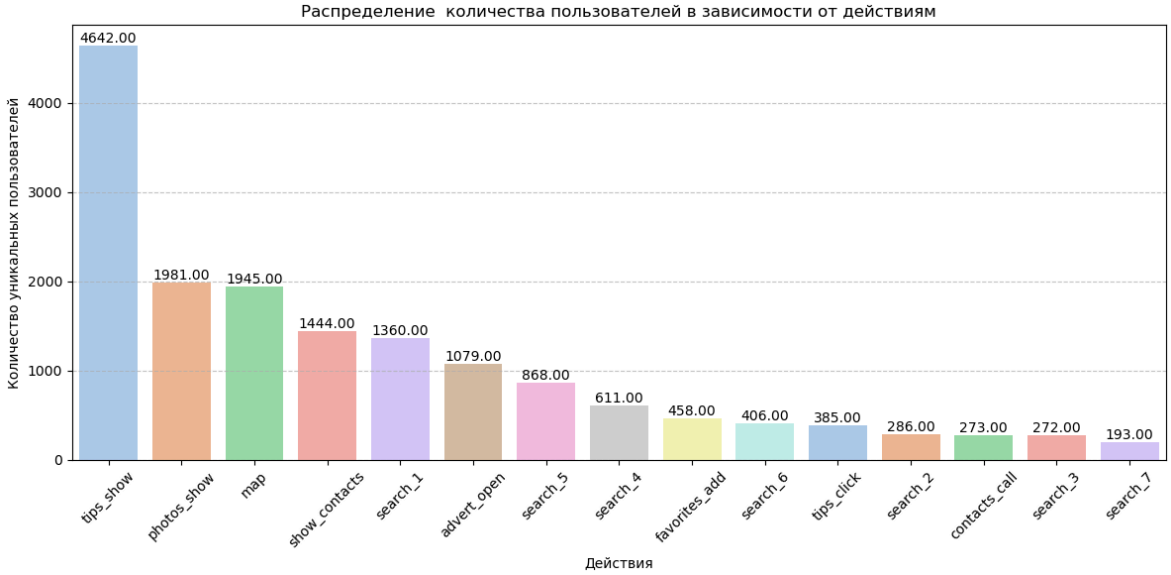
	source	sourses_count
0	yandex	1934
1	other	1230
2	google	1129

Выделим сессии пользователей

In [23]:
```
dataset['session_date'] = dataset['event_time'].dt.date
```

Посмотрим на количество пользователей по действиям

In [24]:
```
user_counts = dataset.groupby(['event_name','session_date'])['user_id'].n
user_counts = user_counts.groupby('event_name')['user_count'].sum().reset
user_counts = user_counts.sort_values(by='user_count', ascending=False)
plt.figure(figsize=(12, 6))
sns.barplot(x='event_name', y='user_count', data=user_counts, palette='pa
plt.xlabel('Действия')
plt.ylabel('Количество уникальных пользователей')
plt.title('Распределение  количества пользователей в зависимости от дейст
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
for index, value in enumerate(user_counts['user_count']):
    plt.text(index, value, f'{value:.2f}', ha='center', va='bottom')
plt.tight_layout()
plt.tight_layout()
plt.show()
```



Что касается распределения действий пользователей по количеству, можем сделать следующие выводы: большая часть пользователей содержится на действии с показом рекомендованных объявлений (4642 пользователя). Почти в три раза меньше человек смотрят фотограмии (1981), чуть поменьше - 1945 - пользуются просмотром карты. Эти три действия - самые многочисленные по числу уникальных пользователей, совершаюших их.

In [25]:
```
user_counts['previous_user_count'] = user_counts['user_count'].shift(1)
user_counts['from_previous'] = user_counts['user_count'] - user_counts['p
user_counts.drop(columns=['previous_user_count'], inplace=True)
user_counts['from_previous'].fillna(0, inplace=True)
```

Также можно узнать различие в количестве пользователей между действиями

In [26]:
```
user_counts
```

Out[26]:

	event_name	user_count	from_previous
14	tips_show	4642	0.0
4	photos_show	1981	-2661.0
3	map	1945	-36.0
12	show_contacts	1444	-501.0
5	search_1	1360	-84.0
0	advert_open	1079	-281.0
9	search_5	868	-211.0
8	search_4	611	-257.0
2	favorites_add	458	-153.0
10	search_6	406	-52.0
13	tips_click	385	-21.0
6	search_2	286	-99.0
1	contacts_call	273	-13.0
7	search_3	272	-1.0
11	search_7	193	-79.0

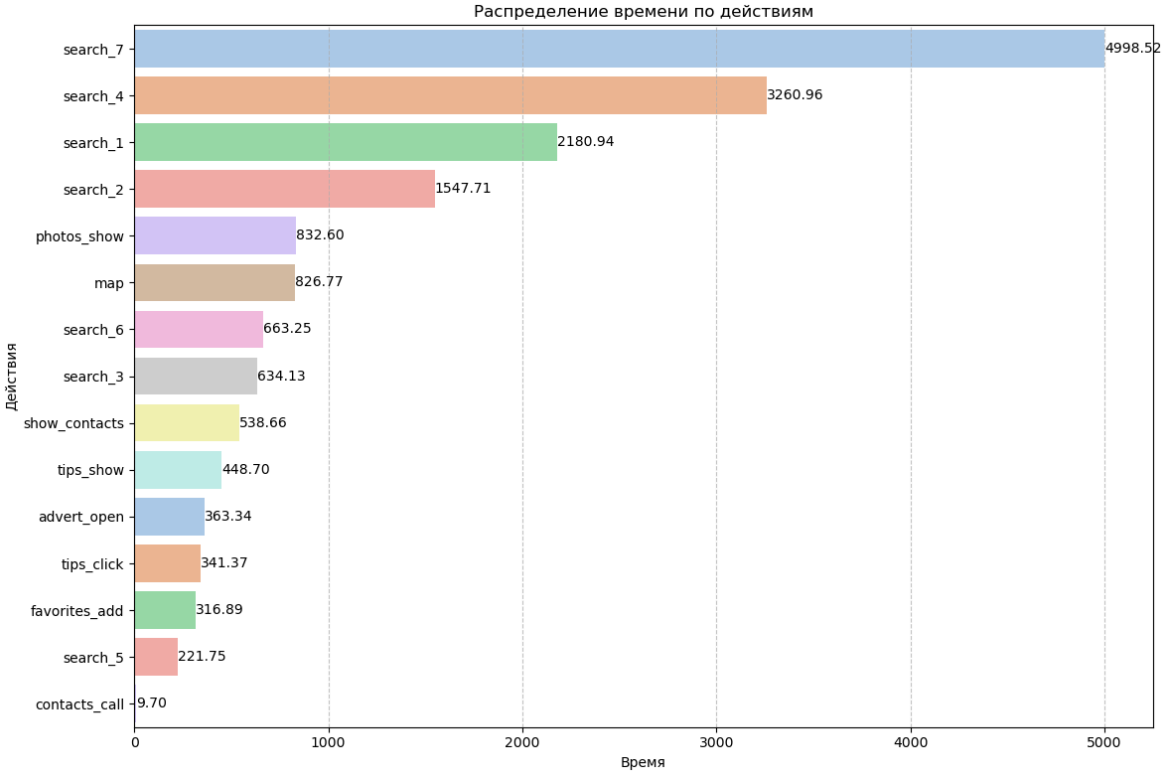
In [27]:
```
dataset['time_spent'] = dataset.groupby(['user_id', 'session_date'])['eve
time_action = dataset.groupby('event_name')['time_spent'].mean().reset_in
time_action = time_action.sort_values(by='time_spent', ascending=False)
time_action
```

Out[27]:

	event_name	time_spent
11	search_7	4998.517178
8	search_4	3260.958417
5	search_1	2180.935202
6	search_2	1547.712203
4	photos_show	832.601948
3	map	826.767633
10	search_6	663.251401
7	search_3	634.131542
12	show_contacts	538.662669
14	tips_show	448.699658
0	advert_open	363.335365
13	tips_click	341.372886
2	favorites_add	316.886520
9	search_5	221.746542
1	contacts_call	9.701261

Посмотрим, сколько времени пользователи тратят на определенные действия

In [28]:
```
plt.figure(figsize=(12, 8))
sns.barplot(x='time_spent', y='event_name', data=time_action, palette='pa
plt.xlabel('Время')
plt.ylabel('Действия')
plt.title('Распределение времени по действиям')
plt.grid(axis='x', linestyle='--', alpha=0.7)
for index, value in enumerate(time_action['time_spent']):
    plt.text(value, index, f'{value:.2f}', va='center')
plt.tight_layout()
plt.show()
```



Также посмотрим распределение пользователей по времени на действиях. Наибольшее время у пользователей содержат действия, связанные с поиском. Какой поиск - не уточняется, но в целом это логично, так кака пользователь заходит с каким-либо запросом в приложение и ищет то, что ему нужно. Также достаточно много времени пользования содержит действие с показом фотографий. Действительно, купить товар, не посмотрев на него, нельзя, поэтому люди тратят время, чтобы изучить всё визуально. Влего лишь на несколько еденицы отличается действие с показом карты. Скорее всего, покупатель старается выбирать товар, который находится ближе к нему. Наименее продолжительным являются действия с показом контактов и с еще одним видом поиска

Посчитаем конверсию в просмотр контактов и возвращаемость пользователей:

In [29]:
```
unique = dataset['user_id'].nunique()
target = dataset[dataset['event_name'] == 'show_contacts']['user_id'].nun
conversion_rate = target / unique
print(f"Конверсия в просмотр контактов: {conversion_rate:.2%}")
```

```
Конверсия в просмотр контактов: 22.85%
```

Как связаны дата и частота событий? Ответим на этот вопрос графиком

In [30]:
```
daily_events = dataset.groupby('session_date').size().reset_index(name='e
daily_events['day_of_week'] = pd.to_datetime(daily_events['session_date']
daily_events['is_weekend'] = daily_events['day_of_week'].isin([5, 6])

plt.figure(figsize=(12, 8))
sns.lineplot(x='session_date', y='event_count', data=daily_events, label=
sns.lineplot(x='session_date', y='event_count', data=daily_events[daily_e
             label='Выходные', linestyle='--', color=sns.color_palette('p
sns.lineplot(x='session_date', y='event_count', data=daily_events[~daily_
```

```
             label='Будни', linestyle='--', color=sns.color_palette('past

plt.xlabel('Дата')
plt.ylabel('Количество событий')
plt.title('Частота событий в зависимости от даты')
plt.legend()
plt.show()
```

```
/var/folders/j2/5vsypwjx25x_h3d_1msym38c0000gn/T/ipykernel_67020/225263733
0.py:6: UserWarning: Ignoring `palette` because no `hue` variable has been
assigned.
  sns.lineplot(x='session_date', y='event_count', data=daily_events, label
='Всего', color='b', palette='pastel')
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: Futu
reWarning: use_inf_as_na option is deprecated and will be removed in a fut
ure version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

С частотой совершения событий ситуация следующая:кроме общего количества событий по дням также будет интересно сравнить активность пользователей в будние дни и в выходные. С 9 октября наблюдалась общая тенденция к уменьшению числа событий. Затем к 13 октября начался рост. Вслед за этим прослеживалась тенденция на уменьшение до 21 сентября, после чего последовал скачок. До 29 октября наблюдались небольшие колебаниЯ, закончившиеся резким падением к 1 ноября. Затем количество событий снова скачкообразно выросло. Можем заметить, что количество событий, совершаемых в будние дни, гораздо стабильнее, чем в выходные, причем стабильно в положительном смысле.

Учитывая, что одну сессию мы считаем одним календарным днем, посмотрим, сколько времени пользователи проводят в приложении. Затем уточним некоторые детали по полученной таблице

In [31]:
```
durations_minutes = dataset.groupby(['user_id', 'session_date']).agg(sess
```

In [32]:
```
durations_minutes
```

Out[32]:

	user_id	session_date	session_start	session_end
0	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	2019-10-07	2019-10-07 13:39:45.989359	2019-10-07 13:49:41.716617
1	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	2019-10-09	2019-10-09 18:33:55.577963	2019-10-09 18:42:22.963948
2	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	2019-10-21	2019-10-21 19:52:30.778932	2019-10-21 20:07:30.051028
3	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	2019-10-22	2019-10-22 11:18:14.635436	2019-10-22 11:30:52.807203
4	00157779-810c-4498-9e05-a1e9e3cedf93	2019-10-19	2019-10-19 21:34:33.849769	2019-10-19 21:59:54.637098
...	...	...	...	...
7812	fffb9e79-b927-4dbb-9b48-7fd09b23a62b	2019-10-29	2019-10-29 13:58:47.865084	2019-10-29 16:13:00.681459
7813	fffb9e79-b927-4dbb-9b48-7fd09b23a62b	2019-10-30	2019-10-30 00:15:43.363752	2019-10-30 11:31:45.886946
7814	fffb9e79-b927-4dbb-9b48-7fd09b23a62b	2019-11-01	2019-11-01 00:24:31.162871	2019-11-01 00:24:53.473219
7815	fffb9e79-b927-4dbb-9b48-7fd09b23a62b	2019-11-02	2019-11-02 01:16:48.947231	2019-11-02 19:30:50.471310
7816	fffb9e79-b927-4dbb-9b48-7fd09b23a62b	2019-11-03	2019-11-03 14:32:55.956301	2019-11-03 16:08:25.388712

7817 rows × 4 columns

In [33]:
```
durations_minutes['session_duration'] = (durations_minutes['session_end']
mean = durations_minutes['session_duration'].mean()
median = durations_minutes['session_duration'].median()
print(f'Средняя продолжительность сессии: {mean:.2f} минут')
print(f'Медианная продолжительность сессии: {median:.2f} минут')
long = durations_minutes[durations_minutes['session_duration'] >= 90]
short = durations_minutes[durations_minutes['session_duration'] <= 1]
print(f'Количество длительных сессий: {len(long)}')
print(f'Количество коротких сессий: {len(short)}')
```

```
Средняя продолжительность сессии: 83.03 минут
Медианная продолжительность сессии: 11.81 минут
Количество длительных сессий: 1448
Количество коротких сессий: 1477
```

Посмотрим, как распределены пользователи с короткими сессиями по источниам

In [34]:
```
short_users = short['user_id'].unique()
short_sources = sourses[sourses['user_id'].isin(short_users)]
short_counts = short_sources['source'].value_counts()
print("Распределение источников для пользователей с короткими сессиями:")
print(short_counts)
```

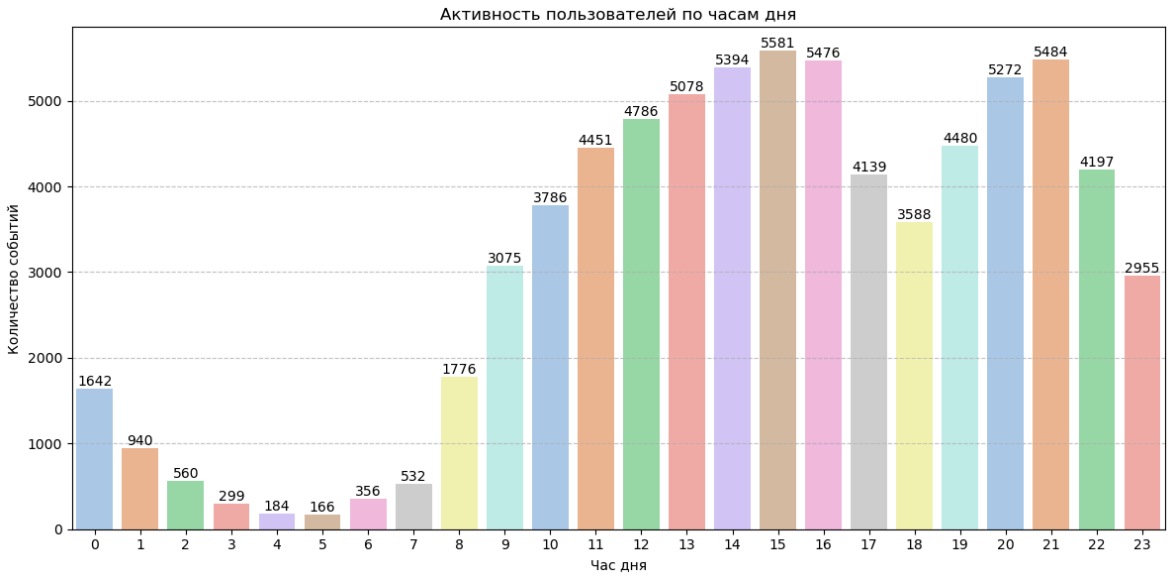
```
Распределение источников для пользователей с короткими сессиями:
source
yandex    401
other     333
google    250
Name: count, dtype: int64
```

Лидер по числу пользователей с самыми короткими сессиями - Яндекс. Возможно, стоит обратить внимание на пользователей этого источника с целью придумать что-либо для того, чтобы побудить их вернуться к использованию приложения. Насколько я знаю, такое иногда используют маркетплейсы. Анализируют активность пользователей и тем пользователям, которые, к примеру, совершают меньше сессий, делают персональные предложения, чтобы они заходили в приложение снова и снова.

Изменяется ли авктивность пользователей в течение дня?

In [35]:
```
dataset['hours'] = dataset['event_time'].dt.hour
hourly_activity = dataset.groupby('hours').size().reset_index(name='event
hourly_activity = hourly_activity.sort_values(by='event_count', ascending
plt.figure(figsize=(12, 6))
sns.barplot(x='hours', y='event_count', data=hourly_activity, palette='pa
plt.xlabel('Час дня')
plt.ylabel('Количество событий')
plt.title('Активность пользователей по часам дня')
plt.grid(axis='y', linestyle='--', alpha=0.7)
for index, row in hourly_activity.iterrows():
    plt.text(row['hours'], row['event_count'], f'{row["event_count"]}', h

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



Посмотрим на активность пользователей по часам дня: можем заметить, что большинство пользователей используют приложение в периоде 12:00-16:00, а также 20:00 - 21:00.

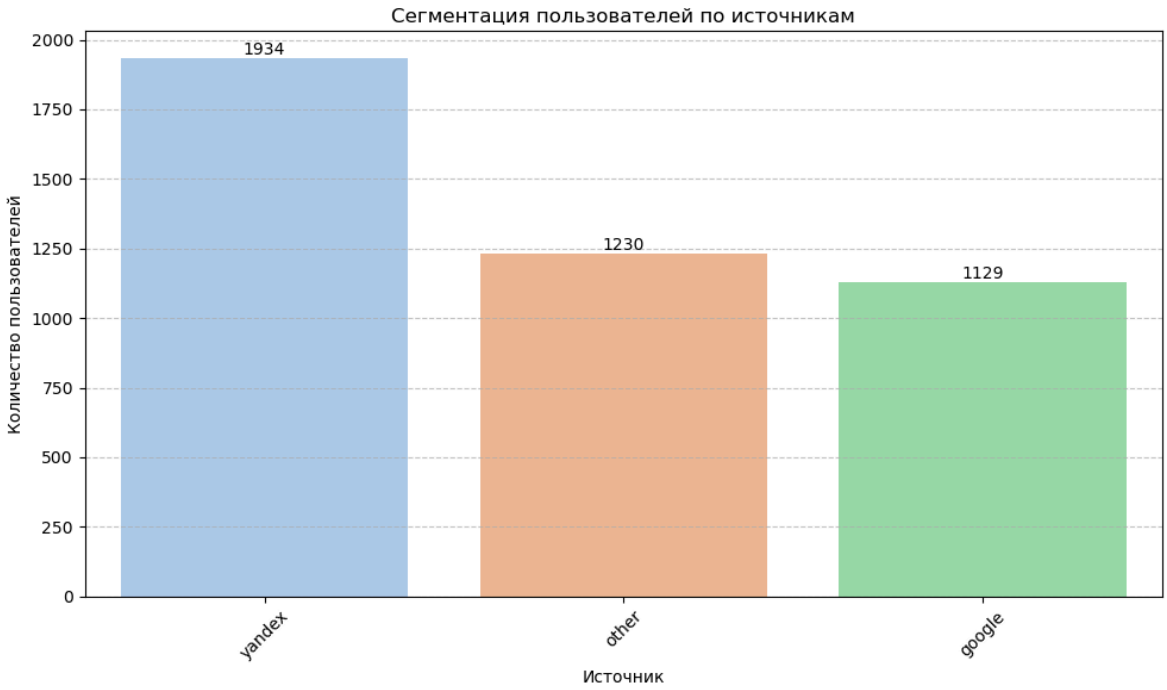
Выводы шаг 3:

Наиболее популярным источником является Яндекс, затем "другие" и Гугл. Также посмотрим распределение пользователей по времени на действиях. Наибольшее время у пользователей содержат действия, связанные с поиском. Какой поиск - не уточняется, но в целом это логично, так кака пользователь заходит с каким-либо запросом в приложение и ищет то, что ему нужно. Также достаточно много времени пользования содержит действие с показом фотографий. Действительно, купить товар, не посмотрев на него, нельзя, поэтому люди тратят время, чтобы изучить всё визуально. Влего лишь на несколько еденицы отличается действие с показом карты. Скорее всего, покупатель старается выбирать товар, который находится ближе к нему. Наименее продолжительным являются действия с показом контактов и с еще одним видом поиска. Что касается распределения действий пользователей по количеству, можем сделать следующие выводы: большая часть пользователей содержится на действии с показом рекомендованных объявлений (4642 пользователя). Почти в три раза меньше человек смотрят фотограмии (1981), чуть поменьше - 1945 - пользуются просмотром карты. Эти три действия - самые многочисленные по числу уникальных пользователей, совершаюших их. Конверсия в целевое действие — просмотр контактов - 22.85%. По тепловой карте удержания можем сделать следующие выводы: с удержанием пользователей дела идут не очень хорошо. После первого дня дял группы пользователей, присоединившихся к приложению 7 октября, на второй денб осталось лишь 18%. К 27-му дню остается 1%. Теряется большое число пользователей. С частотой совершения событий ситуация следующая:кроме общего количества событий по дням также будет интересно сравнить активность пользователей в будние дни и в выходные. С 9 октября наблюдалась общая тенденция к уменьшению числа событий. Затем к 13 октября начался рост. Вслед за этим прослеживалась тенденция на уменьшение до 21 сентября, после чего последовал скачок. До 29 октября наблюдались небольшие колебаниЯ, закончившиеся резким падением к 1 ноября. Затем количество событий снова скачкообразно выросло. Можем заметить, что количество событий, совершаемых в будние дни, гораздо стабильнее, чем в выходные, причем стабильно в положительном смысле.Также определили время , проводимое пользователем в приложении. За сессию в среднем пользователь проводит в нем около 83.03 минут. Медианное значение - 11.81 минут. Также посмотрим на активность пользователей по часам дня: можем заметить, что большинство пользователей используют приложение в периоде 12:00-16:00, а также 20:00 - 21:00.

Шаг 4: сегметация

In [36]:
```
segmentation = sourses.groupby('source').size().reset_index(name='user_co
segmentation = segmentation.sort_values(by='user_count', ascending=False)
print(segmentation)
plt.figure(figsize=(10, 6))
sns.barplot(x='source', y='user_count', data=segmentation, palette='paste
plt.xlabel('Источник')
plt.ylabel('Количество пользователей')
plt.title('Сегментация пользователей по источникам')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
for index, value in enumerate(segmentation['user_count']):
    plt.text(index, value, f'{value}', ha='center', va='bottom')
plt.tight_layout()
plt.show()
```

```
   source  user_count
2  yandex        1934
1   other        1230
0  google        1129
```



Выводы шаг 4:

Мы разбили пользователей по источникам. Такой вариант разбивки достаточно практичный, так как ориентируясь на источники пользователей, можно дорабатывать приложение под источник, привлекающий большую часть клиентов.Также количество человек, участвующих. в каждой группе при разбивке, не критично различается. Что касается результатов сегментации, можем сделать следующие выводы: наибольшим количеством пользователей обладает Яндекс, вслед за ним идут прочие источники, на третьем месте Гугл.

In [37]:
```
dataset
```

Out[37]:

	event_time	event_name	user_id	session_date	time_spent	hc
0	2019-10-07 00:00:00.431357	advert_open	020292ab-89bc-4156-9acf-68bc2783f894	2019-10-07	NaN	
1	2019-10-07 00:00:01.236320	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	2019-10-07	0.804963	
2	2019-10-07 00:00:02.245341	tips_show	cf7eda61-9349-469f-ac27-e5b6f5ec475c	2019-10-07	NaN	
3	2019-10-07 00:00:07.039334	tips_show	020292ab-89bc-4156-9acf-68bc2783f894	2019-10-07	5.803014	
4	2019-10-07 00:00:56.319813	advert_open	cf7eda61-9349-469f-ac27-e5b6f5ec475c	2019-10-07	54.074472	
...	...	...	...	...	...	
74192	2019-11-03 23:53:29.534986	tips_show	28fccdf4-7b9e-42f5-bc73-439a265f20e9	2019-11-03	4.572126	
74193	2019-11-03 23:54:00.407086	tips_show	28fccdf4-7b9e-42f5-bc73-439a265f20e9	2019-11-03	30.872100	
74194	2019-11-03 23:56:57.041825	search_1	20850c8f-4135-4059-b13b-198d3ac59902	2019-11-03	NaN	
74195	2019-11-03 23:57:06.232189	tips_show	28fccdf4-7b9e-42f5-bc73-439a265f20e9	2019-11-03	185.825103	
74196	2019-11-03 23:58:12.532487	tips_show	28fccdf4-7b9e-42f5-bc73-439a265f20e9	2019-11-03	66.300298	

74197 rows × 6 columns

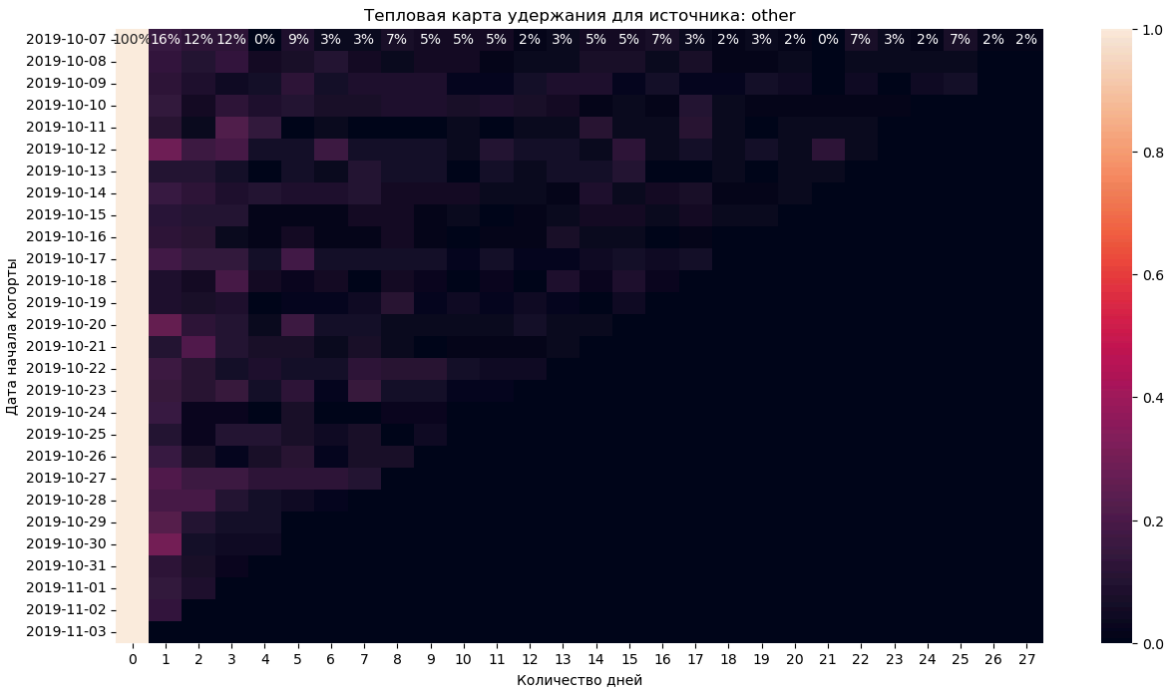
Шаг 5: основные вопросы исследования

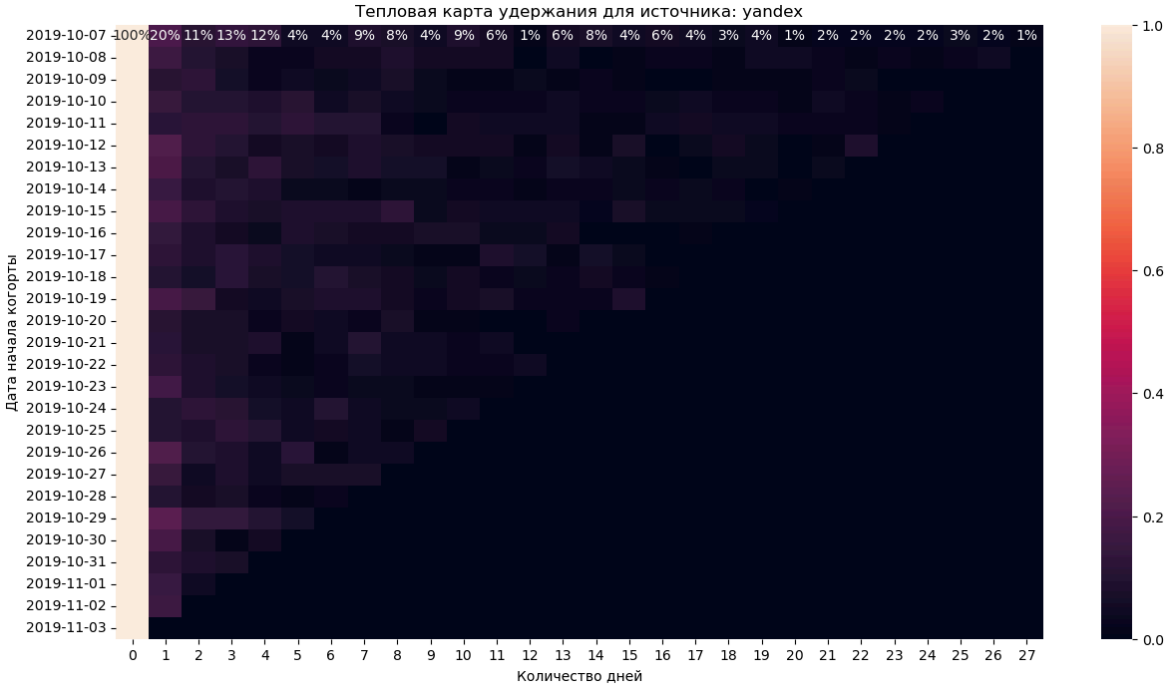
- Пользователи с какой группы склонны часто возвращаться в приложение?
- Пользователи какой группы часто делают целевое событие? (конверсия в целевое действие)

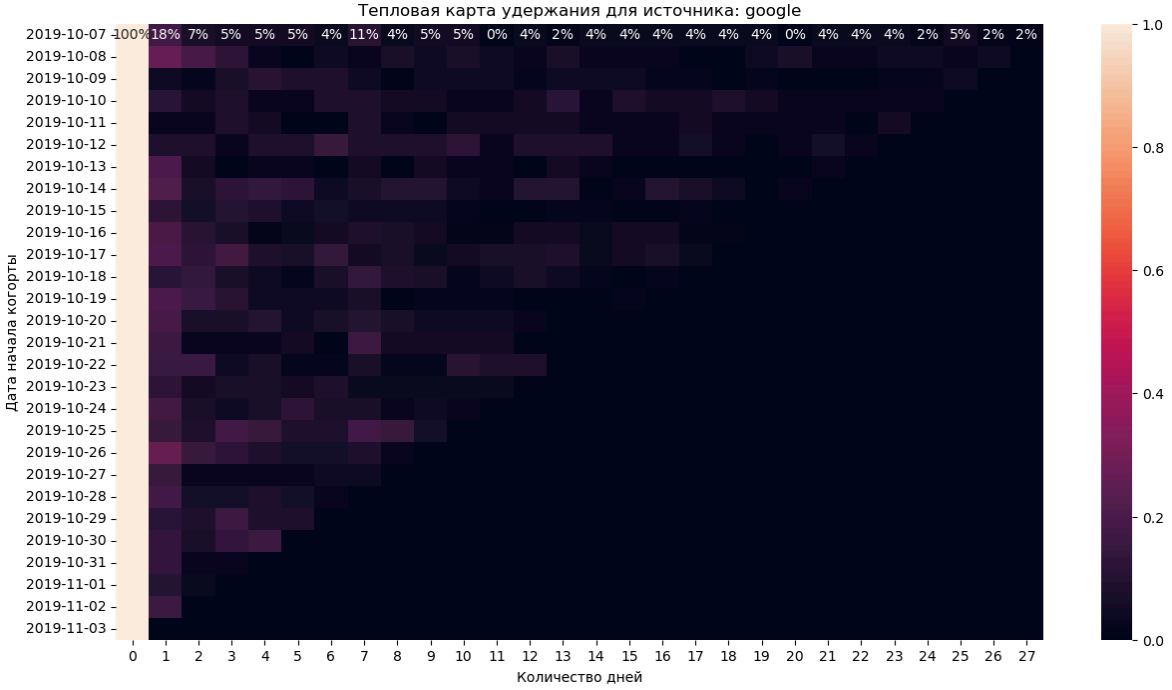
Для начала посмотрим на возвращаемость пользователей из различных источников

In [38]:
```
dataset_merged = dataset.merge(sourses, on='user_id', how='left')
dataset_merged['first'] = dataset_merged.groupby('user_id')['session_date
```

In [39]:
```
dataset_merged['session_date'] = pd.to_datetime(dataset_merged['session_d
dataset_merged['first'] = dataset_merged.groupby('user_id')['session_date
dataset_merged['after_first'] = (dataset_merged['session_date'] - dataset
sourcesu = dataset_merged['source'].unique()
for source in sourcesu:
    source_data = dataset_merged[dataset_merged['source'] == source]
    groups = source_data.groupby(['first', 'after_first']).agg({'user_id'
    groups_groups = groups.pivot(index='first', columns='after_first', va
    size = groups_groups[0]
    retention_matrix = groups_groups.divide(size, axis=0)
    retention_matrix.index = retention_matrix.index.strftime('%Y-%m-%d')
    retention_matrix = retention_matrix.fillna(0)
    plt.figure(figsize=(15, 8))
    sns.heatmap(retention_matrix, annot=True, fmt=".0%")
    plt.title(f'Тепловая карта удержания для источника: {source}')
    plt.xlabel('Количество дней')
    plt.ylabel('Дата начала когорты')
    plt.show()
```



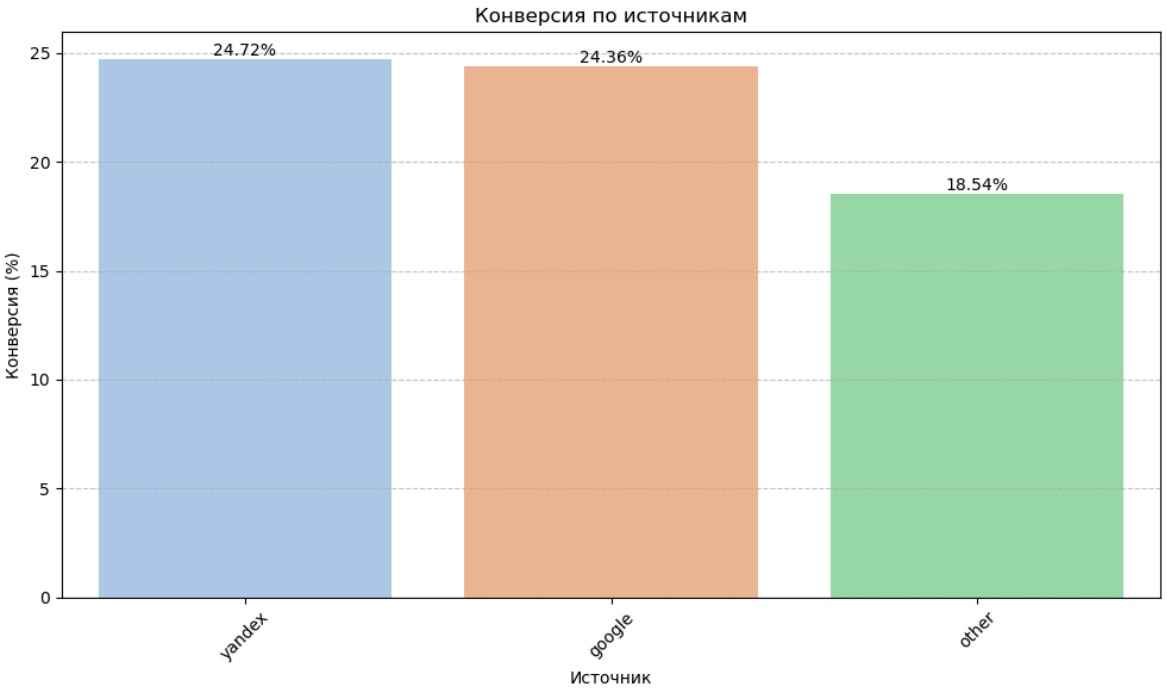




Выводы можем сделать следующие: наибольшее значение удержания пользователей у источников категории "другие". Для "других" на второй день максимально осталось 30% пользователей, что является максимальным значением среди всех источников. У Яндекс максимальное число процентов удержания пользователей составляет 24%, у гугл - 27%. В целом везде пользователи удерживаются достаточно плохо. Надо исправлять ситуацию.

In [40]:
```
main = 'show_contacts'
main_events = dataset_merged[dataset_merged['event_name'] == main]
main_unique_users = main_events.groupby('source')['user_id'].nunique().re
total_unique_users = dataset_merged.groupby('source')['user_id'].nunique(
conversions = pd.merge(main_unique_users, total_unique_users, on='source'
conversions['conversion'] = (conversions['main_events_count'] / conversio
conversions = conversions.sort_values(by='conversion', ascending=False)
plt.figure(figsize=(10, 6))
sns.barplot(x='source', y='conversion', data=conversions, palette='pastel
```

```
plt.xlabel('Источник')
plt.ylabel('Конверсия (%)')
plt.title('Конверсия по источникам')
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
for index, row in enumerate(conversions.itertuples()):
    plt.text(index, row.conversion, f'{row.conversion:.2f}%', ha='center'
plt.tight_layout()
plt.show()
```



Выводы шаг 5:

Пользователи из "прочих" источников возвращаются чаще, чем из Яндекс и Гугл.Для "других" на второй день максимально осталось 30% пользователей, что является максимальным значением среди всех источников. У Яндекс максимальное число процентов удержания пользователей составляет 24%, у гугл - 27%. В целом везде пользователи удерживаются достаточно плохо. Надо исправлять ситуацию. Пользователи из Яндекс имеют более высокую конверсию в просмотр контактов, что является целевым действием. На втором месте находится Гугл.

Шаг 6: проверка гипотез

H0: Конверсия у пользователей Гугл и Яндекс одинакова

H1: Конверсия у пользователей Гугл и Яндекс различается

In [41]:
```
conversions
```

Out[41]:

	source	main_events_count	total_events_count	conversion
2	yandex	478	1934	24.715615
0	google	275	1129	24.357839
1	other	228	1230	18.536585

Используем z-тест для сравния двух независимых групп пользователей и проверим, существует ли статистически значимая разница в их конверсиях.

In [42]:
```
yandex_users = sourses[sourses['source'] == 'yandex']['user_id'].unique()
google_users = sourses[sourses['source'] == 'google']['user_id'].unique()
#посчитаем общее количество уникальных пользователей гугл/яндекс
yandex_total = len(yandex_users)
google_total = len(google_users)
#посчитаем количество пользователей, совершивших целевое действие
yandex_merged = dataset_merged[(dataset_merged['user_id'].isin(yandex_use
]['user_id'].nunique()
google_merged = dataset_merged[(dataset_merged['user_id'].isin(google_use
]['user_id'].nunique()
#рассчет конверсии гугл/яндекс
p_yandex = yandex_merged / yandex_total
p_google = google_merged / google_total
#скомбинированная конверсия
p_comb = (yandex_merged + google_merged) / (yandex_total + google_total)
#разность конверсии у гугл/яндекс
difference = p_yandex - p_google
#z-статистика
z_stat = difference / (p_comb * (1 - p_comb) * (1/yandex_total + 1/google
#p-value
distribution = st.norm(0, 1)
p_value = (1 - distribution.cdf(abs(z_stat))) * 2
print(f'p-value: {p_value:.4f}')
if p_value < 0.05:
    print('Отвергаем нулевую гипотезу: разница между конверсиями у гугл и
else:
    print('Не  отвергаем нулевую гипотезу, разница между конверсиями отсу
print(f'Конверсия пользователей яндекс: {p_yandex:.4f}')
print(f'Конверсия пользователей гугл: {p_google:.4f}')
```

```
p-value: 0.8244
Не  отвергаем нулевую гипотезу, разница между конверсиями отсутствует
Конверсия пользователей яндекс: 0.2472
Конверсия пользователей гугл: 0.2436
```

H0: Пользователи, открывшие карту объявлений, не совершают действия поиска и не открывают карточки объявлений чаще, чем те, кто карту не открывал

H1: Пользователи, открывшие карту объявлений, совершают действия поиска и открывают карточки объявлений чаще, чем те, кто карту не открывал

Логика следующая: карта объявлений - достаточно необычное нововведение в приложении. Посмотрим, стимулирует ли это действие пользователей на такие важные шаги как совершение поиска товаров и открытие карточек объявлений?

In [43]:
```
#создадим столбец, показывающий, относится ли действие к поиску 1-7
dataset['search'] = dataset['event_name'].str.startswith('search')
#отберем уникальных пользователей, совершивших действие с картой
map_users = dataset.loc[dataset['event_name'] == 'map', 'user_id'].unique
#датафрейм в котором содержатся все записи пользователей, открывших карту
opened_map = dataset[dataset['user_id'].isin(map_users)]
#наоборот не открывшие карту пользователи
not_opened_map = dataset[~dataset['user_id'].isin(map_users)]
#количество действий пользователя, открывшего карту
search_map = opened_map[opened_map['search']].groupby('user_id').size()
#количество действий пользователя , не открывшего карту
search_nomap = not_opened_map[not_opened_map['search']].groupby('user_id'
#количество открытий карточек объявлений для каждого пользователя, открыв
advert_map = opened_map[opened_map['event_name'] == 'advert_open'].groupb
#количество открытий карточек объявлений для каждого пользователя, не отк
advert_nomap = not_opened_map[not_opened_map['event_name'] == 'advert_ope
search_results = st.ttest_ind(search_map, search_nomap, equal_var=False)
advert_open_results = st.ttest_ind(advert_map, advert_nomap, equal_var=Fa
print('p-value для  действий поиска:', search_results.pvalue)
print('p-value для карточек объявлений:', advert_open_results.pvalue)
if search_results.pvalue < 0.05 and advert_open_results.pvalue < 0.05:
    print("Отвергаем нулевую гипотезу: пользователи, открывшие карту, чаш
else:
    print("Не получилось отвергнуть нулевую гипотезу: пользователи, откры
```

```
p-value для  действий поиска: 0.8157662454883214
p-value для карточек объявлений: 0.05322374183731996
Не получилось отвергнуть нулевую гипотезу: пользователи, открывшие карту,
совершают действия поиска и открывают карточки объявлений не чаще
```

Выводы шаг 6:

Мы проверяли 2 гипотезы:

-Пользователи, открывшие карту объявлений, не совершают действия поиска и не открывают карточки объявлений чаще, чем те, кто карту не открывал

-Конверсия у пользователей Гугл и Яндекс одинакова

Обе гипотезы подтвердились. Можем прийти к выводу, что Гугл и Яндекс эквиваленты но конверсии пользователей. Также можно сделать вывод, Что карта объявлений не мотивирует пользователей совершать другие действия, такие как поиск и просмотр карточек.

Общие выводы:

1. Структура данных:

Данные представлены в таблицах, содержат информацию о событиях пользователей и источниках. Пропуски и дубликаты отсутствуют, а временные метки приведены к типу datetime для удобства анализа.

2. Анализ данных:

Источники установки: Наиболее популярным источником является Яндекс, за ним следуют «другие» и Гугл. Пользователи из «других» источников чаще возвращаются в приложение.

3. Действия пользователей:

Наиболее популярные действия — просмотр рекомендованных объявлений, фотографий и карты. Однако, карта объявлений не стимулирует пользователей к дальнейшим действиям, таким как поиск и открытие карточек.

4. Конверсия и удержание:

Конверсия в просмотр контактов составляет 22.85%, а retention rate достаточно низкий везде. Пользователи из Яндекс показывают более высокую конверсию в целевое действие.

5. Активность пользователей:

Пользователи наиболее активны в период с 12:00 до 16:00 и с 20:00 до 21:00. Активность в будние дни более стабильна положительная по сравнению с выходными.

6. Гипотезы:

Первая гипотеза о том, что открытие карты не увеличивает частоту поиска и открытия карточек, подтвердилась.

Вторая гипотеза о равенстве конверсии пользователей из Яндекса и Гугла также подтвердилась.

7. Выводы и рекомендации:

Приложение следует оптимизировать с учетом наиболее активных временных интервалов пользователей. К примеру, можно ввести "темную тему" для тех, кто использует его вечером, чтобы было комфортнее читать в темноте.

Стоит обратить внимание на улучшение функционала карты, чтобы она могла стимулировать дальнейшие действия пользователей.

Так как конверсии пользователей Яндекса и Гугла равны, стоит обратить внимание на увеличение удержания пользователей Яндекса, по причине того, что оно ниже.