Table of Contents

Сборный проект-2

Проект для стартапа, который продаёт продукты питания. Необходимо разобраться, как ведут себя пользователи нашего мобильного приложения.

Изучим воронку продаж, узнаем, как пользователи доходят до покупки. Выявим, сколько пользователей доходит до покупки, а сколько — «застревает» на предыдущих шагах и на каких именно.

Исследуем результаты A/A/B-эксперимента: пользователей разбили на 3 группы: 2 контрольные (им показывают старые шрифты) и одну экспериментальную (новые шрифты). Выясним, какой шрифт лучше.

Описание данных

Работаем с датафреймом logs. Каждая запись - это действие пользователя (событие).

- EventName — название события;
- DeviceIDHash — уникальный идентификатор пользователя;

- EventTimestamp — время события;
- ExpId — номер эксперимента: 246 и 247 — контрольные группы, а 248 — экспериментальная.

ПЛАН РАБОТЫ:

1. Откроем файл с данными и изучим общую информацию

2. Подготовим и предобработаем данные:

- заменим названия столбцов
- проверим пропуски, дубликаты, корректность типов данных
- добавим необходимые столбцы даты и времени, а также отдельный столбец дат

3. Изучим и проверим данные. Выявим:

- сколько всего событий в логе?
- сколько всего пользователей в логе?
- сколько в среднем событий приходится на пользователя?
- данными за какой период мы располагаем? Найдем максимальную и минимальную дату, определим, с какого момента данные полные и отбросим более старые?
- как много событий и пользователей мы потеряем, отбросив старые данные?
- проверим, что у нас есть пользователи из всех трёх экспериментальных групп.

4. Изучим воронку событий

- посмотрим какие события есть в логах, как часто они встречаются
- посчитаем, сколько пользователей совершали каждое из этих событий. Посчитаем долю пользователей, которые хоть раз совершали событие
- предположим, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Отбросим не встраивающуюся в воронку события
- по воронке событий посчитаем, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем)
- выясним, на каком шаге теряем больше всего пользователей
- определим, какая доля пользователей доходит от первого события до оплаты

5. Изучим результаты эксперимента

- выявим, сколько пользователей в каждой экспериментальной группе. Проверим, находят ли статистические критерии разницу между выборками 246 и 247
- выберем самое популярное событие, посчитаем число пользователей, совершивших это событие в каждой из контрольных групп, посчитаем долю таких пользователей. проверим, будет ли отличие между группами статистически достоверным. Проделаем то же самое ждя всех других событий. Выясним, можно ли сказать, что разбиение на группы работает корректно
- совершим аналогичные действия с экспериментальной группой. Сравним результаты с каждой из контрольных групп в отдельности по каждому событию, сравним результаты с объединённой контрольной группой. Сделаем выводы из эксперимента
- проверим гипотезы с разным уровнем статистической значимости.

Знакомство с данными, общая информация

In [1]:
```
# импорт библиотек
import pandas as pd
```

```
from scipy import stats as st
import numpy as np
import datetime as dt
import math as mth

import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from plotly import graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio
pio.renderers.default = "png"
svg_renderer = pio.renderers["png"]
svg_renderer.scale = 1.2

plt.style.use('seaborn-poster')
%config InlineBackend.figure_format = 'retina'
```

In [2]:
```
# сохранение датафрейма logs в одноименной переменной
try:
    logs = pd.read_csv(r'\Users\Hp\Desktop\Практикум\sbor-2\logs_exp.csv', sep='\t')
except:
    logs = pd.read_csv('https://code.s3.yandex.net/datasets/logs_exp.csv', sep='\t')
```

In [3]:
```
display(logs.head(10))
logs.info()
```

	EventName	DeviceIDHash	EventTimestamp	ExpId
0	MainScreenAppear	4575588528974610257	1564029816	246
1	MainScreenAppear	7416695313311560658	1564053102	246
2	PaymentScreenSuccessful	3518123091307005509	1564054127	248
3	CartScreenAppear	3518123091307005509	1564054127	248
4	PaymentScreenSuccessful	6217807653094995999	1564055322	248
5	CartScreenAppear	6217807653094995999	1564055323	248
6	OffersScreenAppear	8351860793733343758	1564066242	246
7	MainScreenAppear	5682100281902512875	1564085677	246
8	MainScreenAppear	1850981295691852772	1564086702	247
9	MainScreenAppear	5407636962369102641	1564112112	246

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

Имеем `244126 наблюдений`, пропусков нет. Приступаем к предобработке данных.

Подготовка и предобработка данных

- Замените названия столбцов на удобные для вас;
- Проверьте пропуски и типы данных. Откорректируйте, если нужно;

- Добавьте столбец даты и времени, а также отдельный столбец дат;

`Зададим столбцам удобные названия` с соблюдением snake_case:

In [4]:
```
logs.columns = ['event_name', 'user_id', 'datetime', 'group']
logs.columns
```

Out[4]:
```
Index(['event_name', 'user_id', 'datetime', 'group'], dtype='object')
```

Ранее выяснили, что пропусков в таблице нет, переходим к `корректировке типов данных:`

- столбец с датой приведем к типу datetime64
- добавим столбец 'date', который будет хранить только дату без времени
- посмотрим, какие значения хранит в себе столбец 'group'

In [5]:
```
logs['datetime'] = pd.to_datetime(logs['datetime'],unit='s')
logs['date'] = logs['datetime'].dt.date
logs['date'] = logs['date'].astype('datetime64')
print('Уникальные группы:', logs['group'].unique())
```

```
Уникальные группы: [246 248 247]
```

Никаких сюрпризов нет, в датасете встречается только три группы. Мы знаем, что 246 и 247 — контрольные, а 248 — экспериментальная. Чтобы избежать возможной путаницы, `заменим значения в этом столбце: 246-A1, 247- A2, 248 - B`

In [6]:
```
#функция для переименования групп
def rename_group(group):
    if group==246:
        return 'A1'
    elif group==247:
        return 'A2'
    elif group==248:
        return 'B'
    else:
        return 'UNKNOWN'

logs['group'] = logs['group'].apply(rename_group)
print('Уникальные группы:', logs['group'].unique())
logs.groupby('group')['event_name'].count()
```

Out[6]:
```
Уникальные группы: ['A1' 'B' 'A2']
group
A1    80304
A2    78075
B     85747
Name: event_name, dtype: int64
```

Теперь две контрольные группы носят названия A1 и A2, экспериментальная - B. Обе группы имеют достаточное число событий.

`Проверим наличие дубликатов`

In [7]:
```
print('Число полных дубликатов:', logs.duplicated().sum())
print('Процент дубликатов:', "{0:.2f}%".format(logs.duplicated().sum()/len(logs)*100))
```

```
Число полных дубликатов: 413
Процент дубликатов: 0.17%
```

К сожалению, в в логе встречается 413 дубликатов. Удалим их.

In [8]:
```
logs = logs.drop_duplicates().reset_index(drop=True)
print('Число полных дубликатов:', logs.duplicated().sum())
```

```
Число полных дубликатов: 0
```

Снова вызовем info для проверки внесенных изменений:

In [9]:
```
logs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243713 entries, 0 to 243712
Data columns (total 5 columns):
 #   Column      Non-Null Count   Dtype         
---  ------      --------------   -----         
 0   event_name  243713 non-null  object        
 1   user_id     243713 non-null  int64         
 2   datetime    243713 non-null  datetime64[ns]
 3   group       243713 non-null  object        
 4   date        243713 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(1), object(2)
memory usage: 9.3+ MB
```

Вывод

- переименованы столбцы
- исходный столбец с датой и временем приведен к корректному типу, добавлен новый столбец с датой
- группы A/A/B-теста переименованы, в каждой группе встречается достаточное количество наблюдений
- пропусков нет
- удалены дубликаты, теперь набор данных содержит **243713** наблюдения

Таблица первично предобработана и готова к дальнейшему изучению и анализу

Изучение и проверка данных

- Сколько всего событий в логе?
- Сколько всего пользователей в логе?
- Сколько в среднем событий приходится на пользователя?
- Данными за какой период вы располагаете? Найдите максимальную и минимальную дату. Постройте гистограмму по дате и времени. Можно ли быть уверенным, что у вас одинаково полные данные за весь период? Технически в логи новых дней по некоторым пользователям могут «доезжать» события из прошлого — это может «перекашивать данные». Определите, с какого момента данные полные и отбросьте более старые. Данными за какой период времени вы располагаете на самом деле?
- Много ли событий и пользователей вы потеряли, отбросив старые данные?
- Проверьте, что у вас есть пользователи из всех трёх экспериментальных групп.

Сколько всего событий в логе?

In [10]:
```
print('Всего событий:', len(logs))
print()
print('Уникальных событий:', list(logs['event_name'].unique()))
print()
print('Численное распределение по событиям:')
print(logs.groupby('event_name')['event_name'].count().sort_values(ascending=False))
```

```
Всего событий: 243713

Уникальных событий: ['MainScreenAppear', 'PaymentScreenSuccessful', 'CartScreenAppear', 'Offe
rsScreenAppear', 'Tutorial']

Численное распределение по событиям:
event_name
MainScreenAppear           119101
OffersScreenAppear          46808
CartScreenAppear            42668
PaymentScreenSuccessful     34118
Tutorial                     1018
Name: event_name, dtype: int64
```

Общее число событий сократилось до `243713`. Всего встреается `5 событий`: самое частое событие - MainScreenAppear, реже всего случается событие Tutorial

Сколько всего пользователей в логе? Сколько в среднем событий приходится на пользователя?

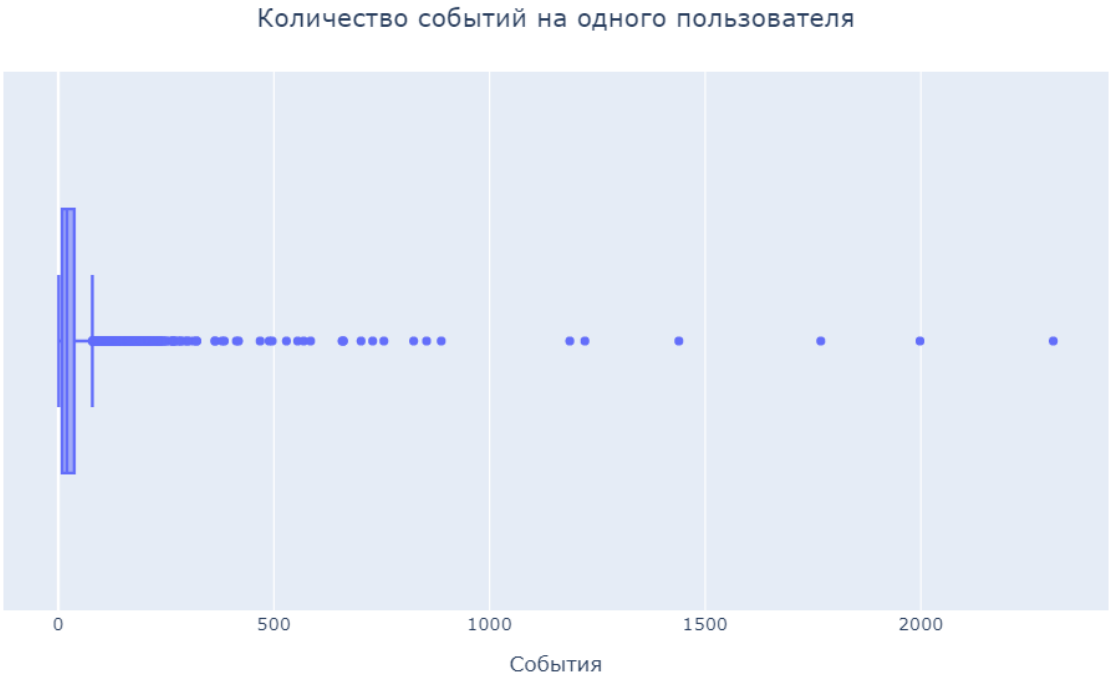
In [11]:
```
print('Уникальных пользователей:', len(logs['user_id'].unique()))
print()
print('Среднее число событий на пользователя:', round((len(logs) / len(logs['user_id'].unique
```

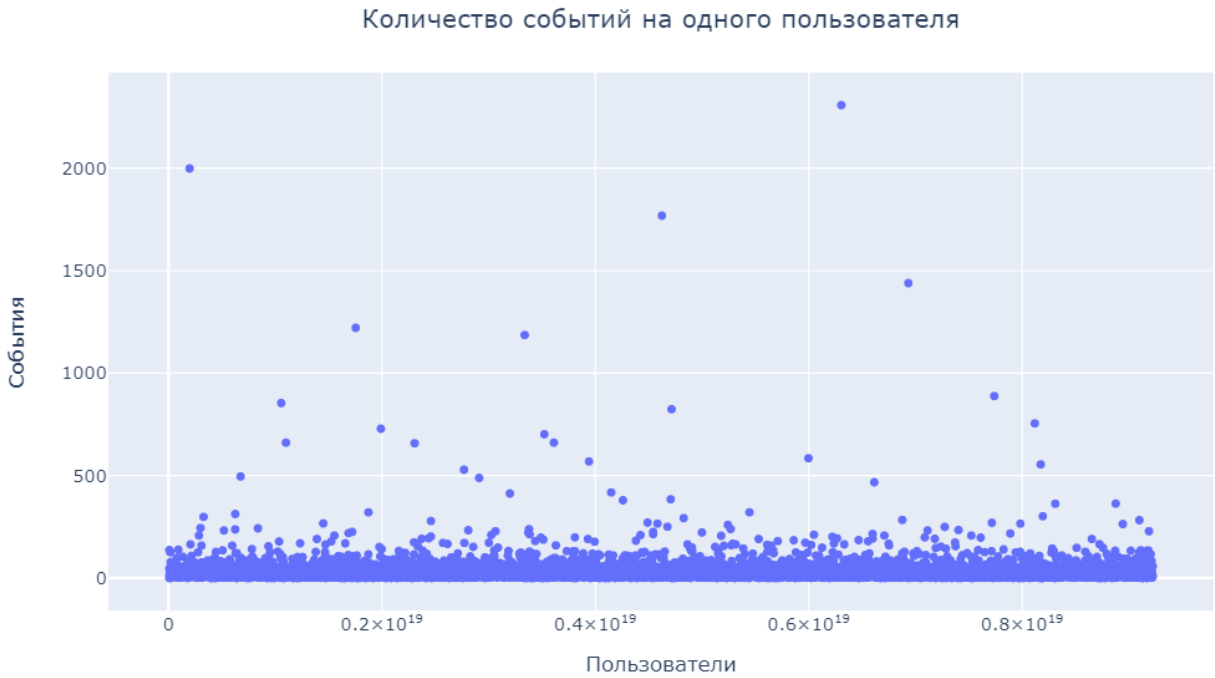
```
Уникальных пользователей: 7551

Среднее число событий на пользователя: 32.28
```

Среднее число часто зависит от аномалий и выбросов. Построим "ящик с усами", чтобы `проверить, насколько полученное нами среднее число зависит от экстремально высоких или низких значений:`

In [12]:
```
per_user_event = logs.pivot_table(index = 'user_id', values = 'event_name',  aggfunc = 'count
    .sort_values(by='event_name').reset_index()
print('Медианное число событий на пользователя:', per_user_event['event_name'].median())

fig = px.box(per_user_event, x="event_name")
fig.update_layout(title='Количество событий на одного пользователя',
    title_x = 0.5,
    showlegend=False,
    height=500, width=900)
fig.update_xaxes(title_text = "События")
fig.show()

fig = px.scatter(x=per_user_event['user_id'], y=per_user_event['event_name'])
fig.update_layout(title='Количество событий на одного пользователя',
    title_x = 0.5,
    showlegend=False,

    height=500, width=900)
fig.update_yaxes(title_text = "События")
fig.update_xaxes(title_text = "Пользователи")
fig.show()
```

```
Медианное число событий на пользователя: 20.0
```





Действительно, в наборе данных встречаются крайне "активные" пользователи: встречаются пользователи, совершившие более 1000 событий. Да и в целом по диаграмме размаха видим большое число выбросов. В таком случае лучше будем ориентироваться на медианное значение, куда меньше подверженное выбросам.

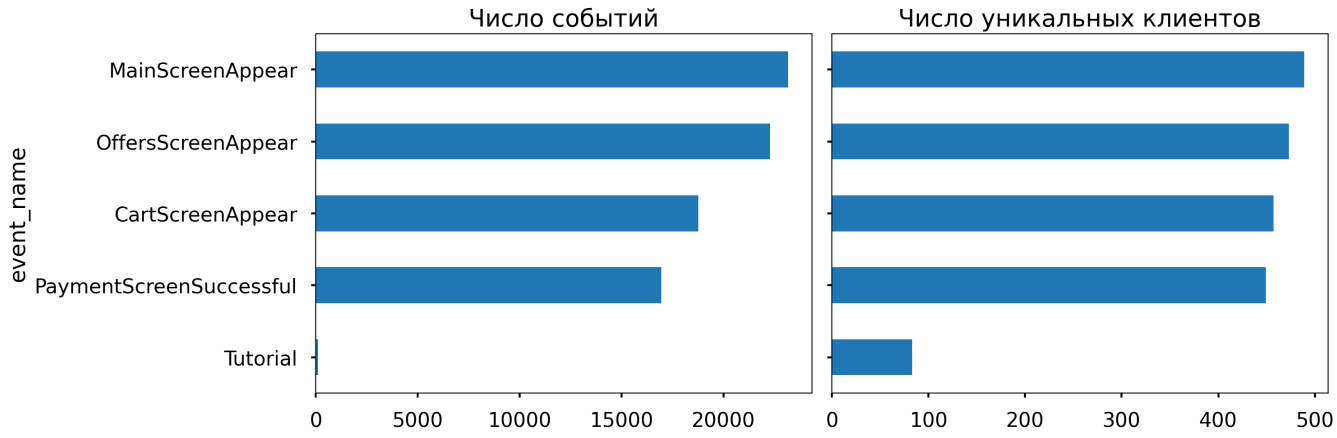
Медианное число событий на пользователя: 20

Дополнительно изучим "аномальных" пользователей. Верхняя граница "ящика" 79, посмотрим на пользователей, совершивших 80 и более событий за исследуемый период.

In [13]:
```
anomal_user = per_user_event.query('event_name > 79')['user_id'].tolist()
print('Пользователей, с числом событий > 79:', len(anomal_user))

plt.figure(figsize=(15, 5))
ax1 = plt.subplot(1, 2, 1)
logs.query('user_id in @anomal_user').groupby('event_name')['user_id'].count().sort_values().
plt.title('Число событий')
ax2 = plt.subplot(1, 2, 2, sharey=ax1)
```

```
logs.query('user_id in @anomal_user').groupby('event_name')['user_id'].nunique().sort_values(
plt.title('Число уникальных клиентов')

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

```
Пользователей, с числом событий > 79: 494
```



В принципе, графики выглядят нормально, нет сильного перевеса у какого-то одного события. `Но все же есть подозрения в ошибке сбора данных,` посмотрим отдельно на клиента, совершившего более 2000 событий

In [14]:
```
top_active_user = per_user_event.sort_values(by='event_name', ascending=False).head(1)['user_
top_active_user_df = logs.query('user_id in @top_active_user')
top_active_user_df.groupby(['date', 'event_name'])['event_name'].count()
```

Out[14]:
```
date        event_name
2019-08-01  CartScreenAppear            27
            MainScreenAppear             9
            OffersScreenAppear          10
            PaymentScreenSuccessful     22
2019-08-02  CartScreenAppear          1070
            MainScreenAppear            19
            OffersScreenAppear          38
            PaymentScreenSuccessful   1063
2019-08-03  MainScreenAppear             6
            OffersScreenAppear          12
2019-08-04  MainScreenAppear             2
            OffersScreenAppear           6
2019-08-05  MainScreenAppear             3
            OffersScreenAppear           2
2019-08-06  MainScreenAppear             2
            OffersScreenAppear           2
2019-08-07  CartScreenAppear             3
            MainScreenAppear             5
            OffersScreenAppear           6
Name: event_name, dtype: int64
```

Обратим внимание на события пользователя 02.08.2019. В обход главного экрана, пользователь 1070 раз посмотрел каталог и 1063 раза успешно оплатил заказ, что выглядит крайне неправдоподобно. `Есть повод усомниться в корректности сбора данных`

Определение корректного периода

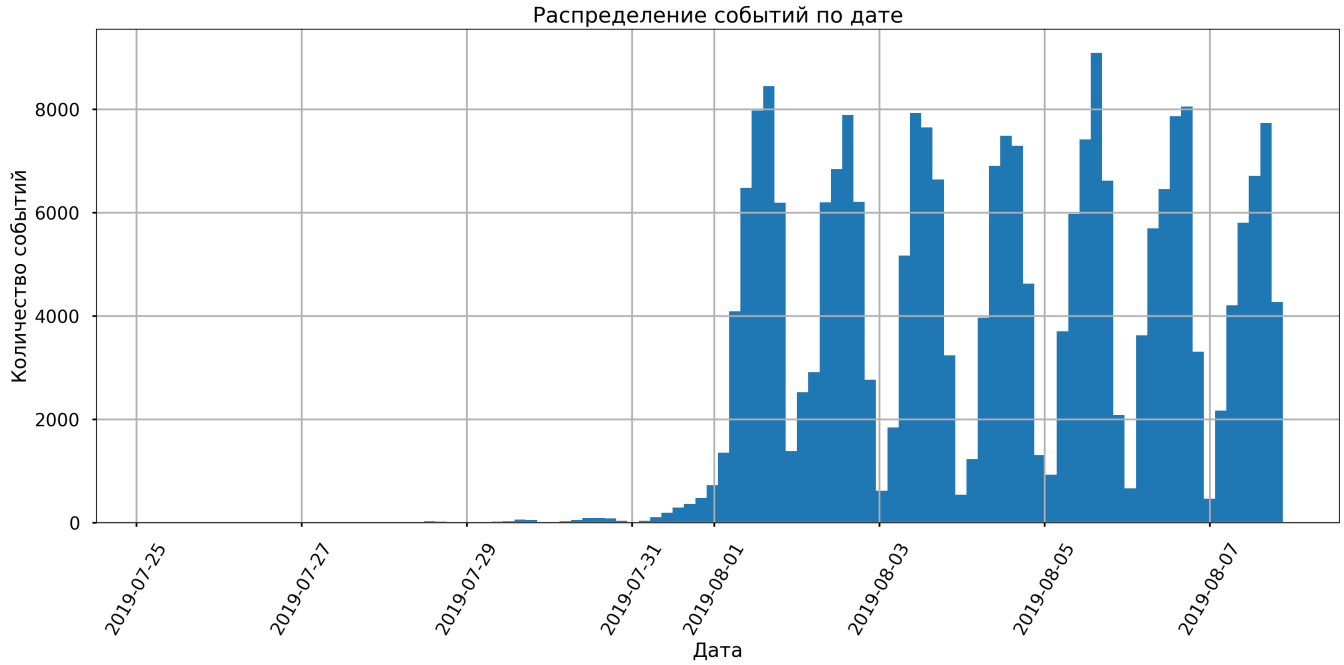
Для начала определим минимальную и максимальную даты в нашем наборе данных:

In [15]:
```
print('Минимальная дата:', logs['datetime'].min())
print('Максимальная дата:',logs['datetime'].max())
print('Временной промежуток:', logs['datetime'].max() - logs['datetime'].min())
```

```
Минимальная дата: 2019-07-25 04:43:36
Максимальная дата: 2019-08-07 21:15:17
Временной промежуток: 13 days 16:31:41
```

Располагаем данным с `25.07.19 по 07.08.2019, чуть менее 14 суток.` Построим гистограмму по дате и времени, посмотрим как распределены данные за весь изучаемый период:

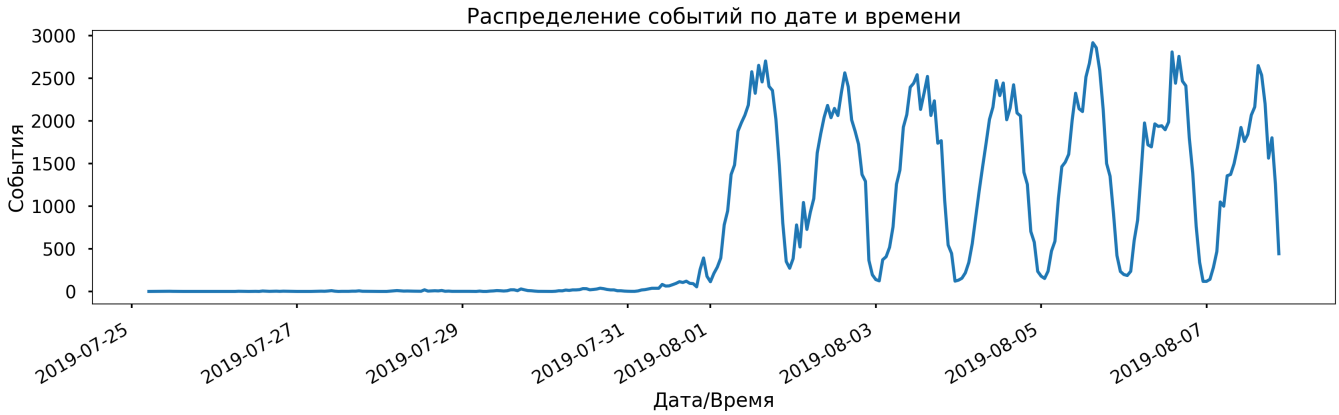
In [16]:
```
plt.figure(figsize = (20, 8))
logs['datetime'].hist(bins = 100,
                      grid=True,
                      legend=False)
plt.xticks(rotation=60);
plt.title('Распределение событий по дате')
plt.xlabel('Дата')
plt.ylabel('Количество событий');
```

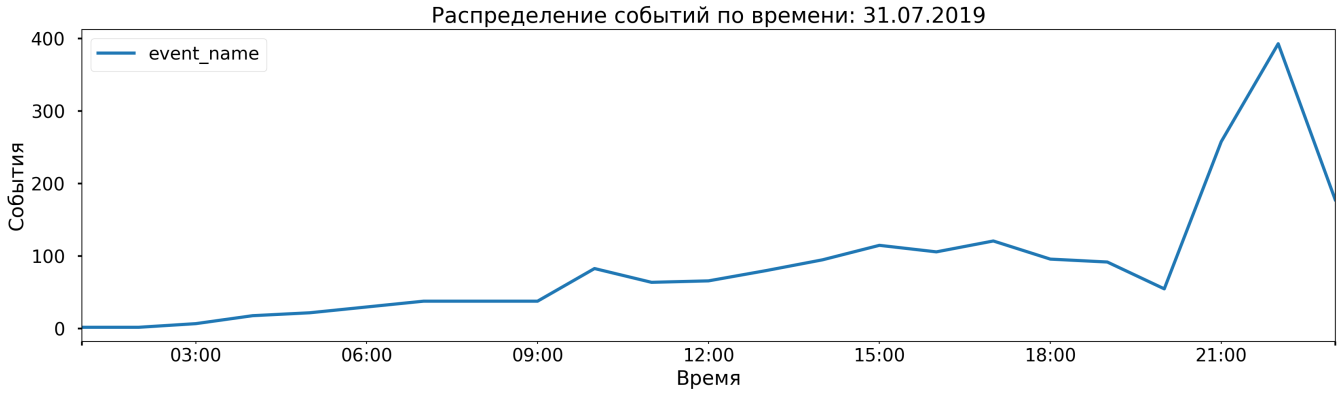


Судя по гистограмме, полными данными `мы располагаем только  начиная с 01.08.` Данных за последнюю неделю июля у нас недостаточно. Видим небольшой всплеск во второй половине дня 31.07.

Создадим столбец 'datetime_hour', в котором округлим дату до ближайшего часа, чтобы посмотреть на диаграмму в более "сглаженном" виде, а так же прицельно посмотрим на распределение событий 31.07, чтобы определить время первого пика

In [17]:
```
#создание столбца с округленной до ближайщего часа датой
logs['datetime_hour'] = logs['datetime'].dt.round('H')

hour_event = logs.groupby('datetime_hour')['event_name'].count().reset_index()
hour_event.plot(x='datetime_hour',
                y='event_name',
                figsize=(20,5),
               legend=False,
               title='Распределение событий по дате и времени',
               xlabel='Дата/Время',
               ylabel='События')
plt.show()
hour_event.query('"2019-07-31 00:00" < datetime_hour < "2019-08-01 00:00"')\
    .plot(x='datetime_hour',
          y='event_name',
          figsize=(20,5),
          title='Распределение событий по времени: 31.07.2019',
         xlabel='Время',
         ylabel='События')
plt.show()
```





`События в каждом августовском дне распределены нормально,` число событий растет к середине дня и спадает к концу суток. Резких отличий между днями нет.

В июльскую неделю имеем ничтожно малое число событий. Видимо, корректный сбор данных начался только в августе. Однако стоит отметить, что первый заметный `рост числа событий приходится на период после 21:00 07.31.2019.` Именно с этого времени можем считать данные достаточно полными, `более старые даты удалим`.

Перезапишем отфильтрованные логи в 'data' и посчитаем потери:

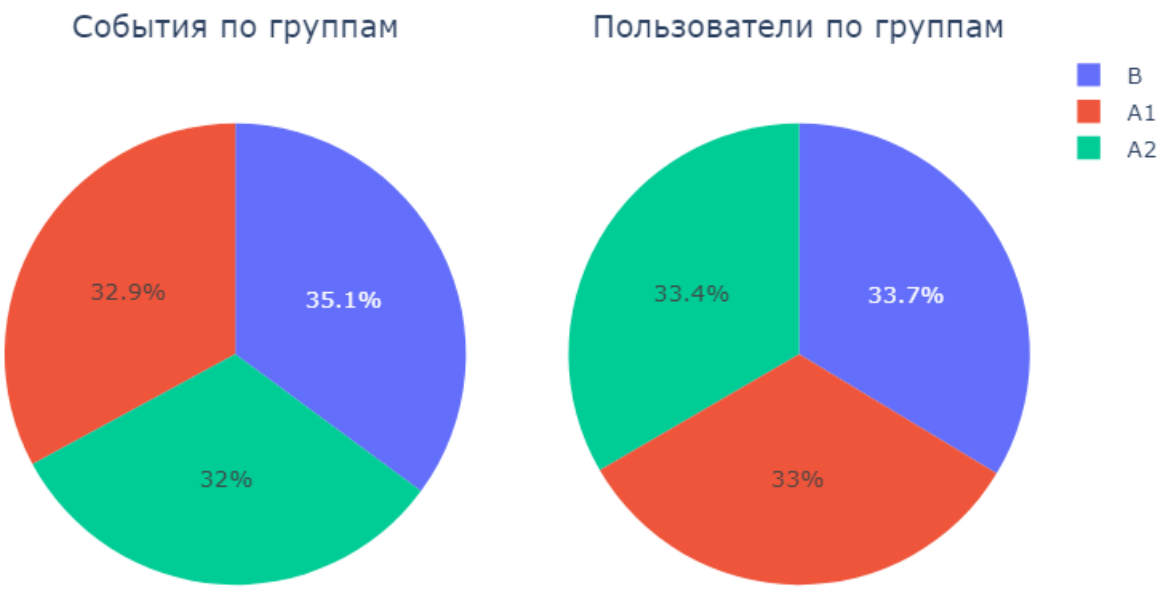
In [18]:
```
data = logs.query('datetime_hour >= "2019-07-31 21:00"')
delete_logs = logs.query('datetime_hour < "2019-07-31 21:00"')

print('СОБЫТИЯ:')
print('Всего:', len(logs))
print('Удалено событий:', len(delete_logs))
print('Осталось:', len(logs)-len(delete_logs))
print('Процент потери:', "{0:.2f}%".format(len(delete_logs)/len(logs)*100))
print()

print('ПОЛЬЗОВАТЕЛИ:')
print('Всего уникальных пользователей:', len(logs['user_id'].unique()))
print('Удалено пользователей:', len(logs['user_id'].unique())-len(data['user_id'].unique()))
print('Осталось:', len(data['user_id'].unique()))
print('Процент потери:', "{0:.2f}%".format((len(logs['user_id'].unique())-len(data['user_id']
                                            len(logs['user_id'].unique())*100))
print()
print('НОВЫЙ АКТУАЛЬНЫЙ ПЕРИОД:')
print('Минимальная дата:', data['datetime'].min())
print('Максимальная дата:', data['datetime'].max())
print('Временной промежуток:', data['datetime'].max() - data['datetime'].min())
```

```
СОБЫТИЯ:
Всего: 243713
Удалено событий: 1964
Осталось: 241749
Процент потери: 0.81%

ПОЛЬЗОВАТЕЛИ:
Всего уникальных пользователей: 7551
Удалено пользователей: 13
Осталось: 7538
Процент потери: 0.17%

НОВЫЙ АКТУАЛЬНЫЙ ПЕРИОД:
Минимальная дата: 2019-07-31 20:30:45
Максимальная дата: 2019-08-07 21:15:17
Временной промежуток: 7 days 00:44:32
```

"Отрезав" неактуальный период мы **потеряли всего 0.81% событий и 0.17% уникальных пользователей.**

Проверка групп на наличие пользователей

In [19]:
```
pie_data = data.pivot_table(index='group', values='user_id', aggfunc={'count', 'nunique'}).re
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'}, {'type':'domain'}]],
                    subplot_titles=['События по группам', 'Пользователи по группам'])

fig.add_trace(go.Pie(labels=pie_data['group'], values=pie_data['count']),
              1, 1)
fig.add_trace(go.Pie(labels=pie_data['group'], values=pie_data['nunique']),
              1, 2)
fig.update_traces(hoverinfo="all")
fig.show()
```



`Пользователи есть в каждой из трех групп, распределены по группам почти в равных долях.` Видим, что у группы B событий чуть больше, чем у остальных.

Вывод

- **Медианное число событий на пользователя: 20**. Среднее число: 32.3, сильно зависит от выбросов
- Набор данных включал в себя наблюдения с 25.07.19 по 07.08.2019. При анализе выявили, что данные за первую неделю неполные, **сократили исследуемый период до 2019-07-31 20:30:45 - 2019-08-07 21:15:17**. Потери при сокращении периода:
  - 13 пользователей (0.17%)
  - 1964 события (0.81%)
- Пользователи распределены по всем трем группам, практически в равных долях
- После удаления неактуального периода исследованию подлежит **241749 событий и 7538 уникальных пользователей**
- Набор данных содержит аномалии, данные по пользователям могут собираться некорректно

Воронка событий

- Посмотрите, какие события есть в логах, как часто они встречаются. Отсортируйте события по частоте.
- Посчитайте, сколько пользователей совершали каждое из этих событий. Отсортируйте события по числу пользователей.
- Посчитайте долю пользователей, которые хоть раз совершали событие.
- Предположите, в каком порядке происходят события. Все ли они выстраиваются в последовательную цепочку? Их не нужно учитывать при расчёте воронки.
- По воронке событий посчитайте, какая доля пользователей проходит на следующий шаг воронки (от числа пользователей на предыдущем). То есть для последовательности событий A → B → C посчитайте отношение числа пользователей с событием B к количеству пользователей с событием A, а также отношение числа пользователей с событием C к количеству пользователей с событием B.
- На каком шаге теряете больше всего пользователей?
- Какая доля пользователей доходит от первого события до оплаты?

Виды событий и их частота. Определение порядка воронки

Мы уже определили, что всего встречается 5 событий. Посмотрим, как часто каждое из событий встречается в наборе данных. Создадим таблицу event, сгруппированную по типу события, со столбцами:

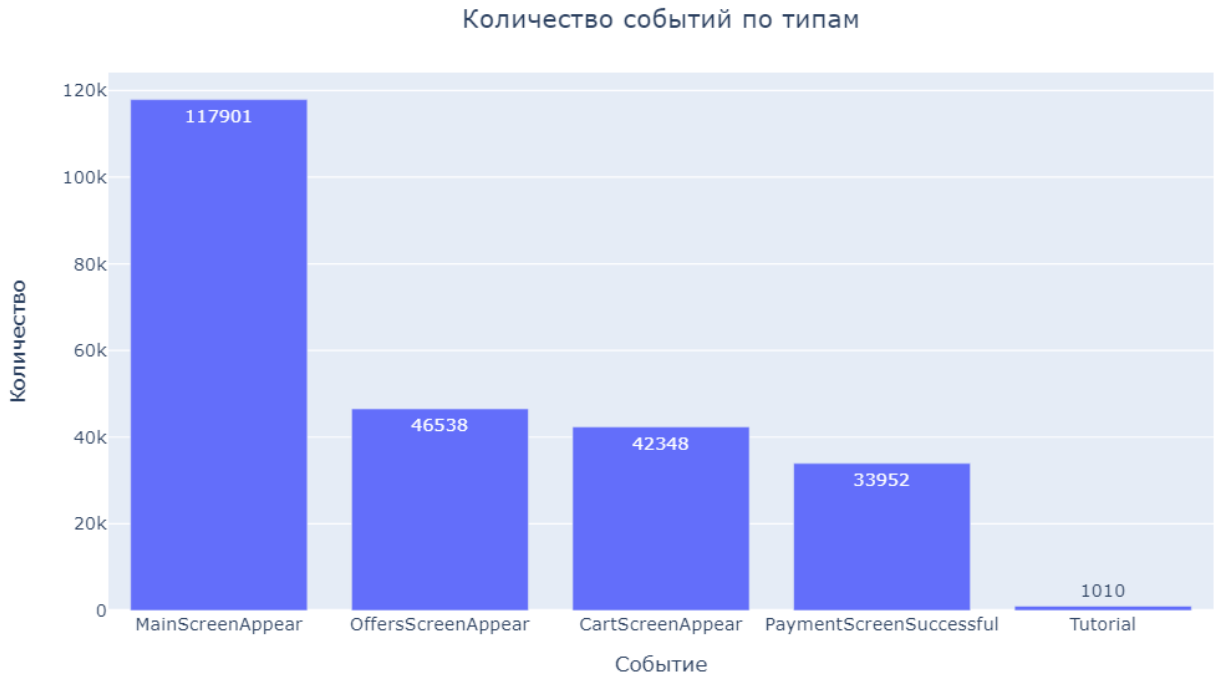
- total_event - событий всего
- user_count - число уникальных пользователей
- user_ratio - доля пользователей, хоть раз совершавших событие

In [20]:
```
event = data.pivot_table(index='event_name', values='user_id', aggfunc={'count', 'nunique'}).
event.columns = ['event_name', 'total_event', 'user_count']
event['user_ratio'] = round((event['user_count']/data['user_id'].nunique())*100, 2)
event=event.sort_values(by='total_event', ascending=False)
event
```

Out[20]:

	event_name	total_event	user_count	user_ratio
1	MainScreenAppear	117901	7423	98.47
2	OffersScreenAppear	46538	4597	60.98
0	CartScreenAppear	42348	3736	49.56
3	PaymentScreenSuccessful	33952	3540	46.96
4	Tutorial	1010	843	11.18

In [21]:
```
fig = px.bar(event.sort_values(by='total_event', ascending=False),
             x='event_name',
             y='total_event',
            text='total_event')
fig.update_xaxes(title_text = "Событие")
fig.update_yaxes(title_text = "Количество")
fig.update_layout(title='Количество событий по типам',
    title_x = 0.5,
    showlegend=False,
    height=500, width=900)
fig.show()
```



`Самое частое событие MainScreenAppear` - пользователи открывали стартовую страницу/ главный экран почти 120.000 раз. На страницу с обучением/ правилами пользования "Tutorial" заходят реже всего.

In [22]:
```
fig = make_subplots(rows=1, cols=2, subplot_titles=['Число пользователей', 'Доли пользователе

fig.add_trace(go.Bar(x=event['event_name'], y=event['user_count'], text=event['user_count']),
fig.add_trace(go.Bar(x=event['event_name'], y=event['user_ratio'], text=event['user_ratio']),
fig.update_traces(hoverinfo="all")
fig.update_layout(
    title="Распределение пользователей по событиям",
    title_x = 0.5,
    showlegend=False)
fig.show()
```

На основании частоты событий можем определить следующую воронку:

1. **MainScreenAppear** - 7423 пользователя, 98,5% - Главная/стартовая страница
2. **OffersScreenAppear** - 4597 пользователей, 61% - Страница предложений/каталог
3. **CartScreenAppear** - 3736 пользователей, 49,56% - Экран корзины/ Товары добавлены в корзину
4. **PaymentScreenSuccessful** - 3540 пользователей, 46,96% - Успешная оплата товара

Tutorial (руководство пользователя/обучение) не выстраивается в последовательную цепочку - во-первых, событие может наступить вне зависимости от других событий, во-вторых, судя по доле пользователей - оно необязательное, клиент может обратиться руководству пользователя в любой момент. Иными словами:

- событие Tutorial не вытекает из какого-то другого события (у него нет **обязательного** предыдущего события)
- событие Tutorial не влечет за собой наступление следующего события

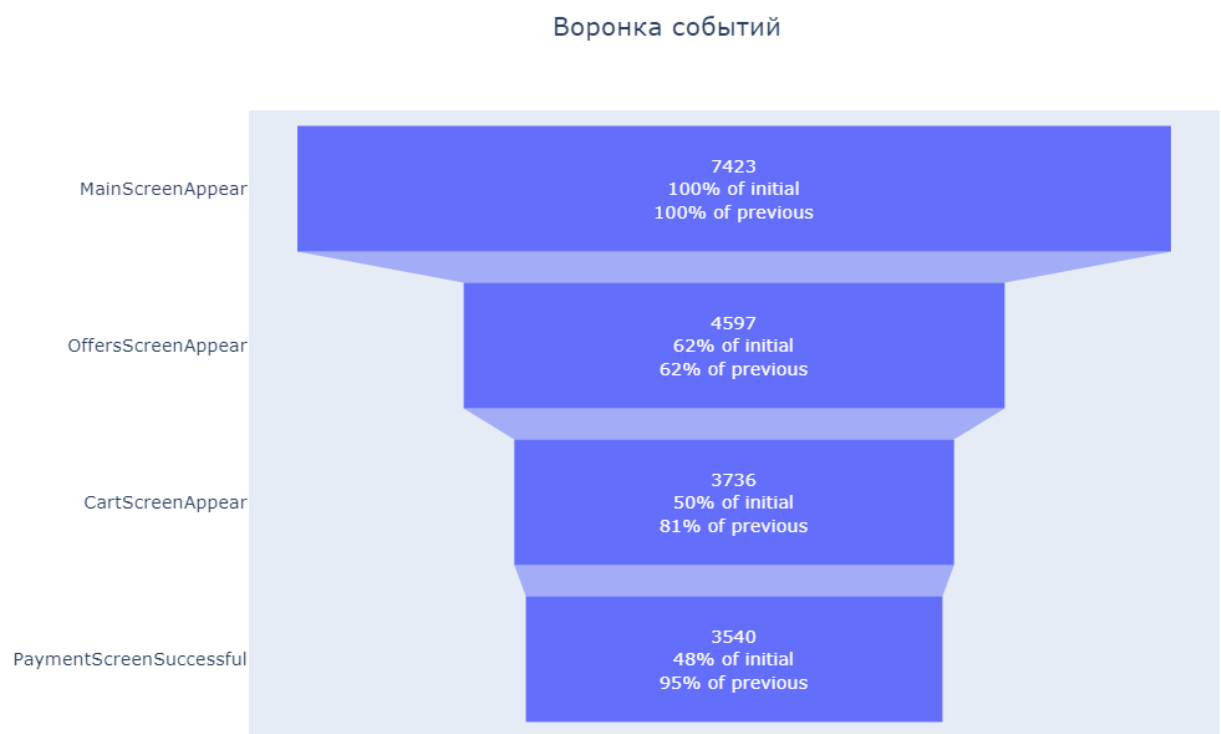
Tutorial не влияет на принятие решения о покупке (либо влияет очень косвенно), соответственно, не будет учитываться при расчете воронки. Однако заметим, что 11% все же обращаются на страницу руководства пользователя, что выглядит немного странным. Достаточно ли интуитивен интерфейс приложения, если 11 из 100 человек обращаются к этой странице?

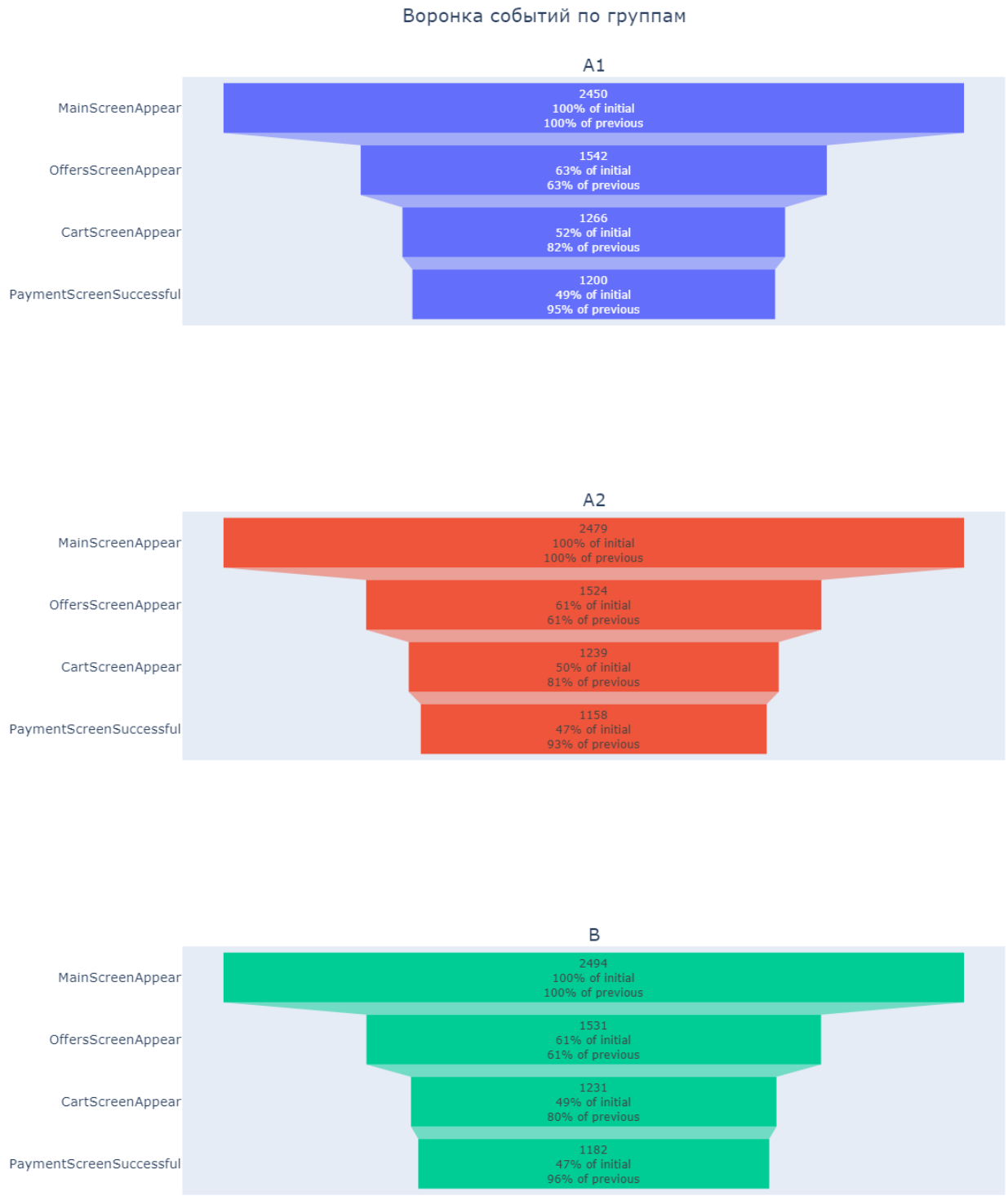
Также стоит отметить, что главную страницу открывают не 100% пользователей. Может быть связано с некорректной выгрузкой данных, возможно, первое событие у этих пользователей произошло в "отрезанный" нами период, но все же рекомендуется проверить наличие технических проблем с запуском главной страницы.

Воронка событий. На каком шаге теряется больше всего клиентов?

In [23]:

```
event = event.query('event_name != "Tutorial"')

fig = go.Figure(go.Funnel(y = event["event_name"],
                          x = event["user_count"],
                          textposition = "inside",
                          textinfo = "value+percent initial+percent previous"))

fig.update_layout(title='Воронка событий',
                  title_x = 0.5,
                  showlegend=False,
                  height=600,
                  width=900)
fig.show()

event_aab = data.query('event_name != "Tutorial"')\
    .pivot_table(index='event_name', columns='group', values='user_id', aggfunc='nunique')\
.reset_index().sort_values(by='A1', ascending=False)

def get_funnel(col_1, col_2, col_3):
    fig = make_subplots(rows=3, cols=1, subplot_titles=['A1', 'A2', 'B'])

    fig.add_trace(go.Funnel(x=col_1, y=event_aab["event_name"], textinfo = "value+percent ini
                  1, 1)
    fig.add_trace(go.Funnel(x=col_2, y=event_aab["event_name"], textinfo = "value+percent ini
                  2, 1)
    fig.add_trace(go.Funnel(x=col_3, y=event_aab["event_name"], textinfo = "value+percent ini
                  3, 1)
    fig.update_traces(hoverinfo="all")
    fig.update_layout(
        title="Воронка событий по группам",
        title_x = 0.5,
        showlegend=False,
        height=1200,
        width=1000)
    fig.show()

get_funnel(event_aab['A1'], event_aab['A2'], event_aab['B'])
```





Проанализируем воронку. В целом, группы мало отличаются между собой, воронки групп практически идентичны.

По всем группам вместе:

1. Открыли приложение на главной странице 7423 пользователя
2. **Только 62% из них перешли в каталог.** Именно на этом этапе **мы теряем больше всего клиентов**, почти 38% клиентов ограничиваются просмотром главной страницы. На это может быть много разных причин, рекомендуется проверить:
   - достаточно ли привлекательна, информативна главная страница?
   - понятно ли интуитивно как перейти к каталогу с товарами?
   - сталкиваются ли пользователи с какими-то техническими неполадками уже на стартовой странице? Возможно, технические проблемы характерны для какой-то конкретной платформы.

3. **81% из тех, что перешли в каталог, добавили товары в корзину.** Высокий показатель, товары, предложенные в каталоге достаточно привлекательны для пользователей.
4. **95% пользователей успешно оплачивают товары, добавленные в корзину.** Тоже хороший показатель, можно попробовать "доработать" оставшиеся 5% - например, отправлять пуш-сообщения с напоминанием о неоплаченных товарах, однако стоит подходить к этому с осторожностью, назойливость подобных сообщений скорее полностью отвернет клиентов.

Конверсия 48%, почти каждый второй клиент совершает заказ.

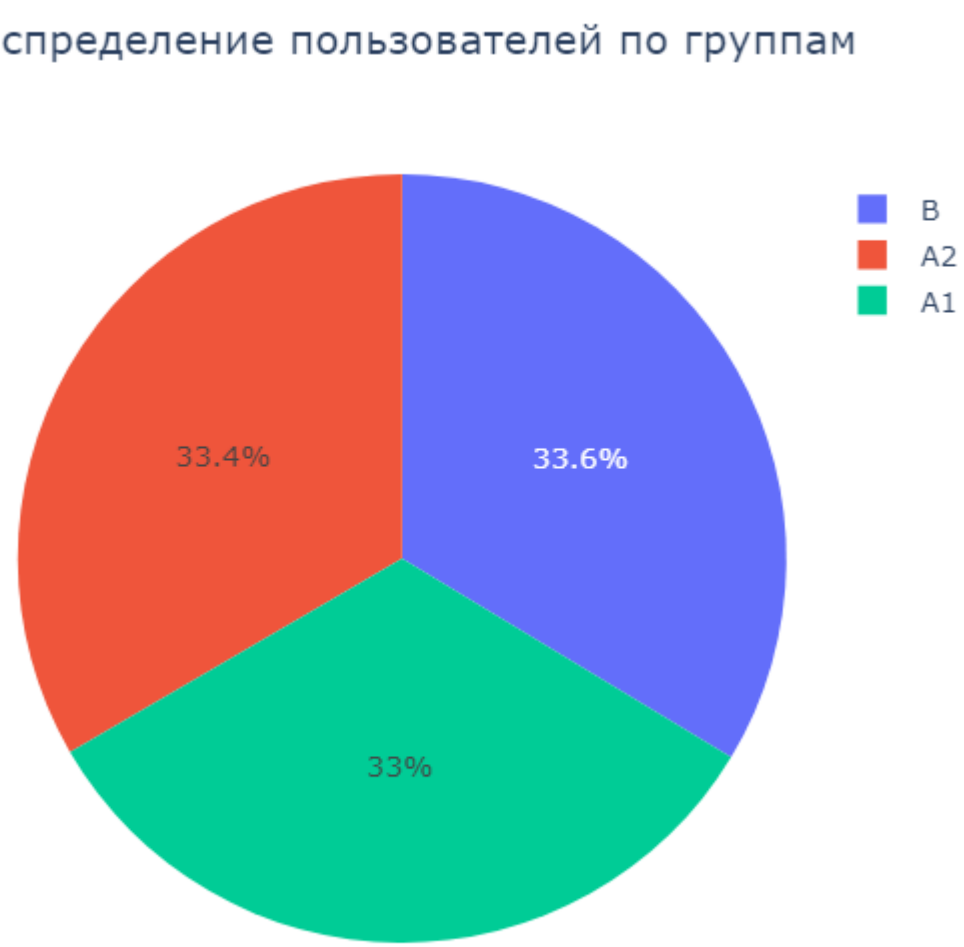
Результаты эксперимента

- Сколько пользователей в каждой экспериментальной группе?
- Есть 2 контрольные группы для A/A-эксперимента, чтобы проверить корректность всех механизмов и расчётов. Проверьте, находят ли статистические критерии разницу между выборками 246 и 247.
- Выберите самое популярное событие. Посчитайте число пользователей, совершивших это событие в каждой из контрольных групп. Посчитайте долю пользователей, совершивших это событие. Проверьте, будет ли отличие между группами статистически достоверным. Проделайте то же самое для всех других событий (удобно обернуть проверку в отдельную функцию). Можно ли сказать, что разбиение на группы работает корректно?
- Аналогично поступите с группой с изменённым шрифтом. Сравните результаты с каждой из контрольных групп в отдельности по каждому событию. Сравните результаты с объединённой контрольной группой. Какие выводы из эксперимента можно сделать?
- Какой уровень значимости вы выбрали при проверке статистических гипотез выше? Посчитайте, сколько проверок статистических гипотез вы сделали. При уровне значимости 0.1 каждый десятый раз можно получать ложный результат. Какой уровень значимости стоит применить? Если вы хотите изменить его, проделайте предыдущие пункты и проверьте свои выводы.

Мы уже проверяли ранее распределение пользователей по группам, посмотрим это распределение после исключения события Tutorial из набора данных. Также дополнительно убедимся, что у `всех трех групп одинаковый исследуемый период` и не встречаются пользователи, `состоящие в нескольких группах одновременно`

In [24]:
```
data = data.query('event_name != "Tutorial"')

group_count = data.groupby('group')['user_id'].nunique().reset_index()
fig = go.Figure(data=[go.Pie(labels=group_count['group'], values=group_count['user_id'])])
fig.update_layout(
        title="Распределение пользователей по группам",
        title_x = 0.5,
        showlegend=True,
        height=500,
        width=500)
fig.show()

#проверка пользователей, состоят ли они в нескольких группах сразу
user_group_check = data.groupby('user_id').agg({'group' : 'nunique'})
if user_group_check['group'].max() == 1:
    print('Каждый пользователь состоит только в одной группе.')
else:
    print('Встречаются пользователи, состоящие в двух группах одновременно.')

print()
```

```
print('Минимальные и максимальные даты в группах:')
data.groupby('group').agg({'date':['min','max']}).reset_index()
```




```
Каждый пользователь состоит только в одной группе.

Минимальные и максимальные даты в группах:
```

Out[24]:

group		date	
		min	max
0	A1	2019-07-31	2019-08-07
1	A2	2019-07-31	2019-08-07
2	B	2019-07-31	2019-08-07

Пользователи распределены по трем группам в примерно равных пропорциях, `различия в количестве между группами менее 1%`. Временной промежуток у всех групп совпадает, `не встречаются пользователи, состоящие одновременно в двух группах`

A1/A2-тест

Есть две контрольные группы A1 и A2, на которых мы и проверим коректность всех механизмов и рассчетов.

Для всех групп будем проводить z-тест.

Т.к. обе группы A - контрольные, подразумеваем, что между ними не должно быть статистической разницы, поэтому `зададим уровень статистической значимости 0.01`.

Зададим функцию 'get_z_test', которая проведет z-тест по каждому событию. Функции передаются две группы и уровень статистической значимости

- **Нулевая гипотеза:** Между группами **нет** статистически значимых различий
- **Нулевая гипотеза:** Между группами **есть** статистически значимые различия

In [25]:
```
#создание таблицы с подсчетом уник.пользователей в каждой группе:
group_users_cnt = data.pivot_table(columns='group', values='user_id', aggfunc='nunique')

#создание функции для проведения z-теста:
def get_z_test(gr_1, gr_2, alpha):
    for event in event_aab.index:
        # пропорция успехов в первой группе:
        p1 = event_aab[gr_1][event] / group_users_cnt[gr_1]
        # пропорция успехов во второй группе:
        p2 = event_aab[gr_2][event] / group_users_cnt[gr_2]
        # пропорция успехов в комбинированном датасете:
        p_combined = ((event_aab[gr_1][event] + event_aab[gr_2][event])/
                      (group_users_cnt[gr_1] + group_users_cnt[gr_2]))
        # разница пропорций в датасетах:
        difference = p1 - p2
        # считаем статистику в ст.отклонениях стандартного нормального распределения
        z_value = difference / mth.sqrt(p_combined * (1 - p_combined) *
                                         (1/group_users_cnt[gr_1] + 1/group_users_cnt[gr_2]))
        # задаем стандартное нормальное распределение (среднее 0, ст.отклонение 1)
        distr = st.norm(0, 1)
        p_value = (1 - distr.cdf(abs(z_value)))*2
        print(event_aab['event_name'][event],)
        print('p-value: {}'.format(p_value))
        if (p_value < alpha):
            print('Отвергаем нулевую гипотезу, между группами есть статистически значимые раз
        else:
            print('Не отвергаем нулевую гипотезу, между группами нет статистически значимых р
        print()

print('Тест A1/A2')
print()
get_z_test('A1', 'A2', 0.01)
```

```
Тест A1/A2

MainScreenAppear
p-value: [0.67020827]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.26673479]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.21811884]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.10288527]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

По результатам A/A теста можем сказать, что контрольные группы `статистически не различаются, разбивка на группы работает корректно.`

A/A/B-тесты

Теперь проведем тесты между контрольными и экспериментальной группой. Группам транслируются разные шрифты, мы хотим увидеть изменения, поэтому установим уровень `статистической значимости в 0.05`.

Проведем следующие тесты:

- A1/B
- A2/B
- A1+A2/B

Проверяемые гипотезы остаются те же:

- **Нулевая гипотеза:** Между группами **нет** статистически значимых различий
- **Нулевая гипотеза:** Между группами **есть** статистически значимые различия

A1/B-тест

In [26]:
```
print('Тест A1/B')
get_z_test('A1', 'B', 0.05)
```

```
Тест A1/B
MainScreenAppear
p-value: [0.39691005]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.21442477]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.08564272]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.22753675]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

По всем четырем событиям **нет статистически значимой разницы** между группами

A2/B-тест

In [27]:
```
print('Тест A2/B')
get_z_test('A2', 'B', 0.05)
```

```
Тест A2/B
MainScreenAppear
p-value: [0.67231677]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.89713765]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.62645998]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.66803679]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

По всем четырем событиям **нет статистически значимой разницы** между группами

A1+A2/B-тест

In [28]:
```
# добавляение в таблицы event_aab и group_users_cnt столбца с суммой A1 и A2
event_aab['AA'] = event_aab['A1']+event_aab['A2']
group_users_cnt['AA'] = group_users_cnt['A1'] + group_users_cnt['A2']
```

In [29]:
```
print('Тест A1+A2/B')
get_z_test('AA', 'B', 0.05)
```

```
Тест A1+A2/B
MainScreenAppear
p-value: [0.45994688]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.43034312]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.20361356]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.65591289]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

По всем четырем событиям **нет статистически значимой разницы** между группами

Провели 12 тестов, ни один из них не подтверждает наличие статистической разницы между группами.

Применение поправок Бонферрони и Шидака

Проведем те же тесты с `поправкой Бонферрони` - поделим принятый уровень значимости на число сравнений

In [30]:
```
alpha = 0.05
hyp_count = 4
bonferroni_alpha = alpha/hyp_count

print('Результаты с применением поправки Бонферрони')
print()
print('Тест A1/A2:')
get_z_test('A1', 'A2',bonferroni_alpha)
print('--------------------------------')

print('Тест A1/B:')
get_z_test('A1', 'B', bonferroni_alpha)
print('--------------------------------')

print('Тест A2/B:')
get_z_test('A2', 'B', bonferroni_alpha)
print('--------------------------------')

print('Тест A1+A2/B:')
get_z_test('AA', 'B', bonferroni_alpha)
print('--------------------------------')
```

```
Результаты с применением поправки Бонферрони

Тест A1/A2:
MainScreenAppear
p-value: [0.67020827]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.26673479]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.21811884]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.10288527]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A1/B:
MainScreenAppear
p-value: [0.39691005]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.21442477]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.08564272]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.22753675]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A2/B:
MainScreenAppear
p-value: [0.67231677]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.89713765]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.62645998]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.66803679]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A1+A2/B:
MainScreenAppear
p-value: [0.45994688]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.43034312]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.20361356]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

```
PaymentScreenSuccessful
p-value: [0.65591289]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
```

Без изменений: по всем четырем событиям **нет статистически значимой разницы** между группами

Теперь применим `поправку Шидака:`

In [31]:
```
sidak_alpha = 1 - (1 - alpha)**(1/hyp_count)

print('Результаты с применением поправки Бонферрони')
print()
print('Тест A1/A2:')
get_z_test('A1', 'A2',sidak_alpha)
print('--------------------------------')

print('Тест A1/B:')
get_z_test('A1', 'B', sidak_alpha)
print('--------------------------------')

print('Тест A2/B:')
get_z_test('A2', 'B', sidak_alpha)
print('--------------------------------')

print('Тест A1+A2/B:')
get_z_test('AA', 'B', sidak_alpha)
print('--------------------------------')
```

```
Результаты с применением поправки Бонферрони

Тест A1/A2:
MainScreenAppear
p-value: [0.67020827]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.26673479]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.21811884]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.10288527]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A1/B:
MainScreenAppear
p-value: [0.39691005]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.21442477]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.08564272]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.22753675]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A2/B:
MainScreenAppear
p-value: [0.67231677]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.89713765]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.62645998]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

PaymentScreenSuccessful
p-value: [0.66803679]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

--------------------------------
Тест A1+A2/B:
MainScreenAppear
p-value: [0.45994688]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

OffersScreenAppear
p-value: [0.43034312]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

CartScreenAppear
p-value: [0.20361356]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

```
PaymentScreenSuccessful
p-value: [0.65591289]
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий

------------------------------
```

По всем четырем событиям **нет статистически значимой разницы** между группами

Вывод

По результатам A/A/B теста нельзя зафиксировать победу ни одной из групп.

Выводы

Проанализирована и предобработана таблица logs, содержащая на момент начала анализа 244126 записей.

Исследуемый период 25.07.19 по 07.08.2019 сокращен до **2019-07-31 20:30:45 - 2019-08-07 21:15:17** (за предыдущую неделю данных очень мало). После сокращения периода и удаления дубликатов исследованию подлежит **241749 событий и 7538 уникальных пользователей** (процент потери небольшой - 0.81% событий и 0.17% уникальных пользователей.)

Пользователи и группы

Пользователи **пропорционально распределены между тремя группами:** контрольные A1 и A2, экспериментальная B, каждая группа содержит ~2500 уникальных пользователей.

Среднее число событий на одного пользователя: 32.3. Среднее сильно зависит от имеющихся выбросов, встречаются пользователи, совершившие аномальное число событий. Необходимо проверить корректность выгрузки данных: встречается пользователь, совершивший более 1000 заказов за сутки. Исходя из этого, лучше ориентироваться на **медианное число событий на пользователя: 20**

Воронка событий

Воронка определена следующим образом:

- **MainScreenAppear** - 7423 пользователя, 98,5% - Главная/стартовая страница
- **OffersScreenAppear** - 4597 пользователей, 61% - Страница предложений/каталог
- **CartScreenAppear** - 3736 пользователей, 49,56% - Экран корзины/ Товары добавлены в корзину
- **PaymentScreenSuccessful** - 3540 пользователей, 46,96% - Успешная оплата товара

Событие **Tutorial** не вписывается в последовательную цепочку и в построении воронки не участвует, но стоит заметить, что к этой странице обращаются 11% клиентов, что достаточно много и может сигнализировать о неинтуитивности интерфейса.

Анализ воронки:

- **Основной отток клиентов происходит уже на первом этапе: только 62% пользователей переходят со стартовой страницы в каталог**. Именно этому этапу следует уделить особое внимание:
  - достаточно ли привлекательна, информативна главная страница?
  - понятно ли интуитивно как перейти к каталогу с товарами?
  - сталкиваются ли пользователи с какими-то техническими неполадками уже на стартовой странице? Возможно, технические проблемы характерны для какой-то конкретной платформы.
- На остальных этапах не выявлено ярко выраженных проблем:
  - **Из каталога 81% клиентов добавляют товары в корзину** - большой процент пользователей находит нужные товары
  - **95% клиентов успешно оплачивают товары из корзины.** Рекомендуется проверить нет ли каких-то технических неполадок на этапе оплаты, возможно стоит отправлять пуш-уведомления с напоминанием о "забытой корзине"

До оплаты доходит 3540 пользователей, что составляет 47% от общего числа. Воронки групп значимо не различаются между собой.

Результаты эксперимента:

Группы численно различаются между собой менее, чем на 1%, исследуемый период у всех групп одинаковый, каждый пользователь состоит только в одной группе.

- Проведен A/A тест между контрольными группами, группы распределены корректно, значимой разницы между контрольными группами не обнаружено.

- Проведены тесты между каждой контрольной и экспериментальной группой, а также между объединенной контрольной группой и экспериментальной. Тесты проведены по каждому событию, кроме Tutorial. **По результатам A/A/B-теста нельзя зафиксировать победу ни одной из групп**.

Между группами нет статистически значимой разницы, **изменения шрифта не принесет качественных изменений.**