Table of Contents

Проект для мобильного приложения "Ненужные вещи": выделение групп пользователей на основе поведения

Команда приложения "Ненужные вещи" хочет проанализировать поведение своих пользователей для дальнейшей адаптации приложения. Интересующие метрики:

1. retention rate,
2. время, проведённое в приложении,
3. частота действий,
4. конверсия в целевое действие — просмотр контактов.

Также необходимо сегментировать пользователей на основе действий, выявить целевую группу, и проверить две статистические гипотезы. По итогам исследования будет подготовлена презентация и дашборд в Tableau.

Декомпозиция/ План исследования

1. Импорт необходимых библиотек и открытие датасетов
2. Знакомство с данными, предобработка (обеих таблиц)
   - 2.1. Переименование столбцов
   - 2.2. Обработка пропусков при наличии
   - 2.3. Проверка на явные и неявные дубликаты
   - 2.4. Проверка столбцов с категориальными значениями на наличие неявных дубликатов
   - 2.5. Изменение типов данных при необходимости
   - 2.6. Беглый взгляд на распределение данных, определение временного промежутка
   - 2.7. Объединение таблиц, предварительная проверка совпадений user_id
   - 2.8. Добавление столбцов с датой и номером дня
3. Исследовательский анализ
   - 3.1 Retention Rate
     - 3.1.1. RR: когорты по неделям

        - 3.1.2. RR: когорты по дням
    - 3.2. Сессии, анализ времени, проведенного в приложении
        - 3.2.1. Объединение событий в сессии
        - 3.2.2. Длительность сессий
        - 3.2.3. Количество сессий по дням
        - 3.2.4. Определение Топа-часов по по количеству сессий
        - 3.2.5. Количество действий за одну сессию
    - 3.3. События, анализ частот действий
        - 3.3.1. Количество событий за весь период
        - 3.3.2. Количество событий по пользователям
        - 3.3.3. Количество действий в день по типу событий
        - 3.3.4. Количество действий за одну сессию
    - 3.4. Конверсия в целевое действие
        - 3.4.1 Рассчет конверсии в целевое действие
        - 3.4.2. Определение воронки событий и пути пользователя
        - 3.4.3. Анализ целевых действий
4. Сегментация пользователей
    - 4.1 Определение признаков для сегментации
    - 4.2 Сегментация пользователей по среднему количеству действий за одну сессию
    - 4.3 Анализ сегментов. Определение ЦА
5. Проверка гипотез
    - 5.1 Гипотеза о разнице конверсий пользователей, пришедших из Yandex и Google
    - 5.2 Гипотеза о разнице конверсий пользователей совершающих 'favorites_add' и не совершающих это действие
6. Общие выводы
7. Презентация
8. Дашборд

Все пункты и подпункты являются "генеральным планом" - по ходу анализа могут быть добавлены другие подпункты. После каждого блока будут даны выводы и рекомендации, где это необходимо.

Описание данных

В нашем распоряжении два датасета, содержащих следующие данные:

- **mobile_dataset**
    - event.time — время совершения
    - event.name — название события
        - advert_open — открытие карточки объявления
        - photos_show — просмотр фотографий в объявлении
        - tips_show — пользователь увидел рекомендованные объявления
        - tips_click — пользователь кликнул по рекомендованному объявлению
        - contacts_show и show_contacts — пользователь нажал на кнопку "посмотреть номер телефона" на карточке объявления
        - contacts_call — пользователь позвонил по номеру телефона на карточке объявления
        - map — пользователь открыл карту размещенных объявлений
        - search_1 — search_7 — разные события, связанные с поиском по сайту
        - favorites_add — добавление объявления в избранное
    - user.id — идентификатор пользователя

- **mobile_sources.csv**mobile_sources.csv
  - userId — идентификатор пользователя
  - source — источник, с которого пользователь установил приложение

Импорт библиотек, открытие датасетов

In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

from scipy import stats as st
import math as mth

import seaborn as sns

import matplotlib.pyplot as plt
plt.style.use('ggplot')
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import plotly
import plotly.express as px
from plotly import graph_objects as go
from plotly.subplots import make_subplots
import plotly.io as pio
pio.templates.default = 'ggplot2'

pio.renderers.default = "png"
svg_renderer = pio.renderers["png"]
svg_renderer.scale = 1.2

from pandas.plotting import register_matplotlib_converters

pd.options.display.max_colwidth = 130
```

In [2]:
```
try:
    sources = pd.read_csv(r'\Users\Hp\Desktop\Практикум\FINAL\mobile_sources.csv')
except:
    sources = pd.read_csv('https://code.s3.yandex.net/datasets/mobile_soures.csv')

sources.head()
```

Out[2]:

	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 [3]:
```
try:
    mobile_dataset = pd.read_csv(r'\Users\Hp\Desktop\Практикум\FINAL\mobile_dataset.csv')
except:
    mobile_dataset = pd.read_csv('https://code.s3.yandex.net/datasets/mobile_dataset.csv')
mobile_dataset.head()
```

Out[3]:

	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

Оба датасета загружены.

Знакомство с данными. Предобработка

Таблица sources

In [4]:
```
sources.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
```

Таблица содержит `4293 наблюдения,` 2 столбца, пропусков нет, данные хранятся в корректном типе.

- приведем названия столбцов к snake_case
- проверим, нет ли дубликатов в столбце с ID
- посмотрим, какие данные хранятся в столбце с источниками, не встречаются ли неявные дубликаты
- бегло посмотрим как распределены данные по источникам

Переименование столбцов

In [5]:
```
sources = sources.rename(columns={'userId':'user_id'})
print("Названия столбцов в таблице 'sources':", sources.columns.tolist())
```

```
Названия столбцов в таблице 'sources': ['user_id', 'source']
```

Проверка столбца user_id

In [6]:
```
if len(sources['user_id'].unique()) == len(sources):
    print("В столбце 'user_id' нет дубликатов, все ID уникальны")
else:
    print('Встречаются дубликаты, необходима обработка')
```

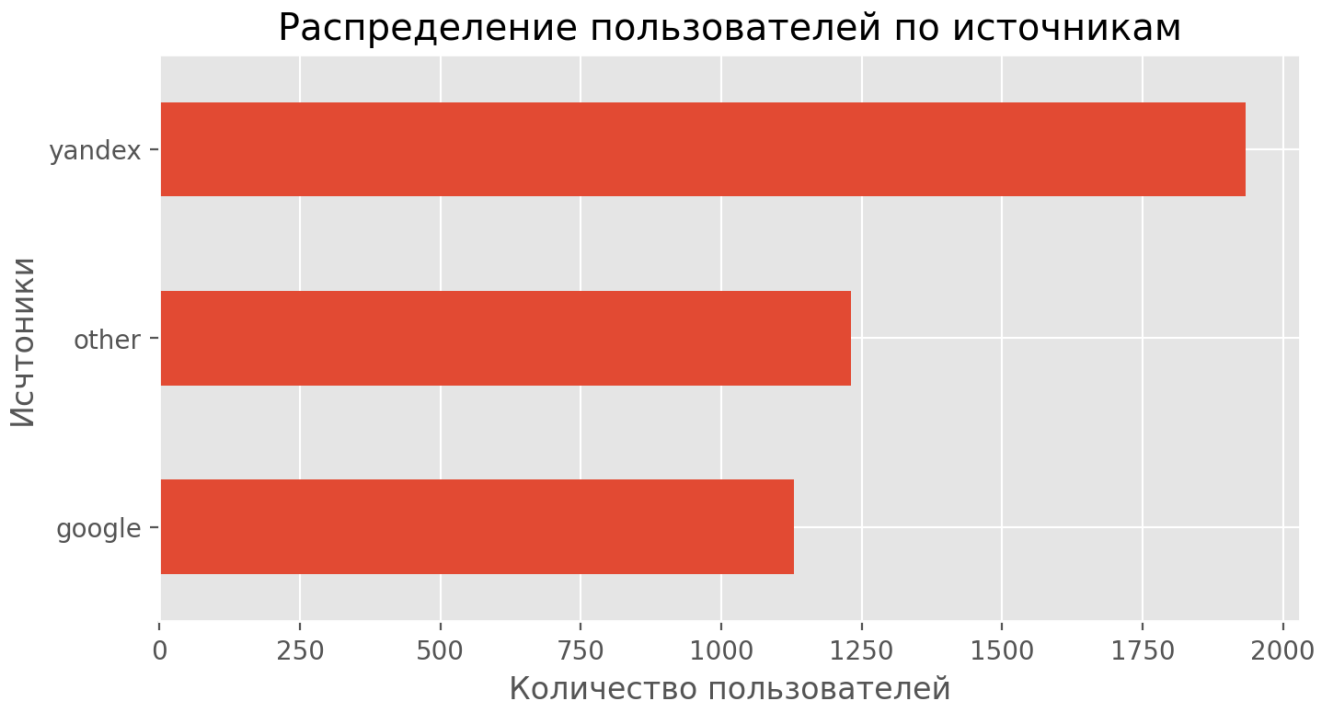
```
В столбце 'user_id' нет дубликатов, все ID уникальны
```

Проверка столбца source

In [7]:
```
print("Источники в столбце 'sources':", sources['source'].unique())
sources.groupby('source')['user_id'].count().plot(kind='barh',
                                                  figsize=(8,4),
                                                            xlabel='Исчтоники',
```

```
                                                    title='Распределение пользователей
plt.xlabel('Количество пользователей');
```

```
Источники в столбце 'sources': ['other' 'yandex' 'google']
```



Встречается `три источника`. Большинство клиентов пришло из Яндекс, данные распределены нормально

Таблица 'sources' готова к дальнейшей работе

Таблица mobile_dataset

In [8]:
```
mobile_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
```

Таблица содержит `74197 наблюдений,` 3 столбца. Пропусков нет.

- приведем столбцы к snake_case
- проверим наличие неявных дубликатов в столбце с названиями событий
- приведем столбец с датой к корректному типу
- выявим временной промежуток
- оценим как распределены данные по типу события

Переименование столбцов

In [9]:
```
mobile_dataset.columns = [x.replace('.', '_') for x in mobile_dataset.columns.values]
print("Названия столбцов в таблице 'mobile_dataset':", mobile_dataset.columns.tolist())
```

```
Названия столбцов в таблице 'mobile_dataset': ['event_time', 'event_name', 'user_id']
```

Проверка дубликатов

In [10]:
```
print('Количество полных дубликатов:', mobile_dataset.duplicated().sum())
```

```
Количество полных дубликатов: 0
```

In [11]:
```
print("Количество дубликатов в связке 'event_time'-'user_id':",
      len(mobile_dataset[mobile_dataset.duplicated(subset=['event_time','user_id'])]))
```

```
Количество дубликатов в связке 'event_time'-'user_id': 0
```

Проверка столбца event_name

In [12]:
```
mobile_dataset['event_name'].sort_values().unique().tolist()
```

Out[12]:
```
['advert_open',
 'contacts_call',
 'contacts_show',
 'favorites_add',
 'map',
 'photos_show',
 'search_1',
 'search_2',
 'search_3',
 'search_4',
 'search_5',
 'search_6',
 'search_7',
 'show_contacts',
 'tips_click',
 'tips_show']
```

Очевидно, события `'contacts_show' и 'show_contacts' относятся к одному и тому же событию`, стоит их объединить.

Отметим, что встречаются `7 видов события search.` Из описания данных нам неизвестно как расшифровывается каждое из семи "поисковых" событий по отдельности, мы не можем утверждать, что эти события происходят в определенном порядке. `Объединим все "поиски" в общее событие 'search'`

In [13]:
```
'''Функция, принимает значение строки,
    если значение строки = 'contacts_show', то значение заменится на 'show_contacts',
    если строка содержит 'search', то значение строки заменится на 'search'
    остальные значения остаются неизменными
'''
def rename_event_name(row):
    if row == 'contacts_show':
        return 'show_contacts'
    elif 'search' in row:
        return 'search'
    else:
        return row
```

In [14]:
```
mobile_dataset['event_name'] = mobile_dataset['event_name'].apply(rename_event_name)
```

In [15]:
```
mobile_dataset.groupby('event_name')['user_id'].count().sort_values().plot(kind='barh',
                                                                           figsize = (10,4),
                                                                           xlabel='События',
                                                                           title='Распределен

plt.xlabel('Количество действий');
```

Чаще всего происходит событие `'tips_show'` - скорее всего, это "неизбежное" событие, при запуске приложения пользователь видит рекомендованные объявления. В целом, распределение выглядит нормальным.

Столбец event_time, определение временного промежутка

Необходимо `изменить тип данных только в столбце с датой`. Отметим, что в столбце 'event_time' указаны миллисекунды, отсечем их, оставим только дату и время до секунд включительно

In [16]:
```
mobile_dataset['event_time'] = pd.to_datetime(mobile_dataset['event_time'], format='%Y-%m-%d
mobile_dataset['event_time'] = mobile_dataset['event_time'].dt.strftime("%Y-%m-%d %H:%M:%S")
mobile_dataset['event_time'] = pd.to_datetime(mobile_dataset['event_time'], format='%Y-%m-%d
display(mobile_dataset.head(2))
mobile_dataset.info()
```

	event_time	event_name	user_id
0	2019-10-07 00:00:00	advert_open	020292ab-89bc-4156-9acf-68bc2783f894
1	2019-10-07 00:00:01	tips_show	020292ab-89bc-4156-9acf-68bc2783f894

```
<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  datetime64[ns]
 1   event_name  74197 non-null  object        
 2   user_id     74197 non-null  object        
dtypes: datetime64[ns](1), object(2)
memory usage: 1.7+ MB
```

Дата и время приведены к корректному типу. Посмотрим, какой временной промежуток дан для исследования

In [17]:
```
print('Минимальная дата:', mobile_dataset['event_time'].min())
print('Максиимальная дата:', mobile_dataset['event_time'].max())
print('Временной промежуток:', mobile_dataset['event_time'].max() - mobile_dataset['event_tim
```

```
Минимальная дата: 2019-10-07 00:00:00
Максиимальная дата: 2019-11-03 23:58:12
Временной промежуток: 27 days 23:58:12
```

В нашем распоряжении почти 28 дней (4 недели)

Таблица 'mobile_dataset' готова к дальнейшей работе

Объединение таблиц, добавление необходимых столбцов

Проверка значений в общем столбце, объединение таблиц

Для дальнейшей работы `необходимо объединить таблицы в одну по общему столбцу user_id.` Проверим соответствие user_id: в обеих таблицах должны быть одни и те же user_id.

In [18]:
```
if mobile_dataset['user_id'].sort_values().unique().tolist() == sources['user_id'].sort_value
    print('Уникальные id в обеих таблицах совпадают')
else:
    print('В таблицах встречаются несовпадающие уникальные id')
```

```
Уникальные id в обеих таблицах совпадают
```

Никаких сюрпризов, id в таблицах совпадают, `объединим таблицы в 'data'`, с которой и будем работать далее

In [19]:
```
data = mobile_dataset.merge(sources, on='user_id', how='left')
display(data.sample(5))
data.info()
```

	event_time	event_name	user_id	source
45072	2019-10-24 17:17:13	advert_open	ac87fdbf-7f28-4f1a-a35d-66ce09eb3dee	other
51123	2019-10-26 20:23:40	tips_show	23a000d8-36aa-4a4d-91ae-a25597fe09b2	yandex
15478	2019-10-13 22:07:38	tips_show	1a454cb5-c741-4685-89e1-fae58db18118	yandex
34296	2019-10-21 08:03:58	photos_show	e7090338-0c13-488b-aaa0-828ffa55cdbd	yandex
63550	2019-10-30 20:42:06	tips_show	aed9d717-9562-4ac9-8629-d7ef7fd7029e	yandex

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 74197 entries, 0 to 74196
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   event_time  74197 non-null  datetime64[ns]
 1   event_name  74197 non-null  object        
 2   user_id     74197 non-null  object        
 3   source      74197 non-null  object        
dtypes: datetime64[ns](1), object(3)
memory usage: 2.8+ MB
```

Таблицы объединины корректно, пропусков нет, все столбцы на месте

Добавление столбцов

Для дальнейшего анализа добавим столбцы:

- 'date' - с датой (без времени)
- 'day' - с днем недели

In [20]:
```
data['date'] = data['event_time'].dt.strftime("%Y-%m-%d")
data['date'] = pd.to_datetime(data['date'], format='%Y-%m-%d')

data['weekday'] = data['event_time'].dt.dayofweek
```

In [21]:
```
display(data.sample(4))
data.info()
```

	event_time	event_name	user_id	source	date	weekday
16341	2019-10-14 10:31:08	tips_show	c68bdd6d-bad7-4f51-8113-302f2364387c	yandex	2019-10-14	0
37849	2019-10-22 12:46:11	tips_show	cdea846c-f331-4e00-beba-4837aa078f50	yandex	2019-10-22	1
39183	2019-10-22 20:19:45	show_contacts	e549f8ef-653b-4c5c-a6bd-8970e6bd860b	google	2019-10-22	1
12579	2019-10-12 19:58:49	advert_open	1580911b-65db-4f1a-be7e-1ca39becac30	google	2019-10-12	5

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 74197 entries, 0 to 74196
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   event_time  74197 non-null  datetime64[ns]
 1   event_name  74197 non-null  object        
 2   user_id     74197 non-null  object        
 3   source      74197 non-null  object        
 4   date        74197 non-null  datetime64[ns]
 5   weekday     74197 non-null  int64         
dtypes: datetime64[ns](2), int64(1), object(3)
memory usage: 4.0+ MB
```

Необходимые столбцы успешно добавлены. **Таблица 'data' готова к дальнейшему анализу**

Выводы

Загружены и предобработаны оба датасета: 'sources' и 'mobile_dataset'.

- все столбцы приведены к snake_case
- оба датасета не содержат пропусков
- проанализированы дубликаты, проработаны неявные дубликаты в категориальных значениях
- данные приведены к корректным типам
- просмотрены основные распределения данных (по источнику, по типу событий), данные распределены нормально
- оценен временной промежуток: располагаем данными за период 7.10.2019-3.11.2019, 4 недели
- датасеты объедены в 'data' по общему столбцы user_id (уникальные id в обеих таблицах совпадают), далее работаем с этой таблицой
- созданы столбцы с датой и днем недели

Обратим внимание, `что мы не располагаем данными о самих сессиях,` только о дате начала каждого события. Иными словами, мы не знаем даты начала каждой сессии и ее длительности, что усложняет работу и вынуждает для дальнейшего анализа самостоятельно объединить события в одну сессию на основании тайм-аута, `что может несколько исказить результаты исследования.`

Таблица 'data' готова к дальнейшему анализу

Исследовательский анализ

Retention Rate

Retention Rate - один из важных показателей "здоровья" бизнеса, построим графики у выясним, `насколько успешно мы удерживаем клиентов.`

Нам известно, что в датасете содержатся данные пользователей, впервые совершивших действия в приложении после 7 октября 2019 года, `т.е. в нашем наборе данных все пользователи новые` .

Для построения графика Retention `необходимо разбить пользователей на когорты.` Мы можем сформировать когорты по дню первого события, получить 28 когорт рассчитать процент удержания на каждый лайфтайм, но в этом случае получим достаточно громоздкий и малоинформативный график. `Сформируем когорты по неделям` и пранализируем, какой процент пользователей удерживается каждую следующую неделю.

Retention Rate: когорты по неделям

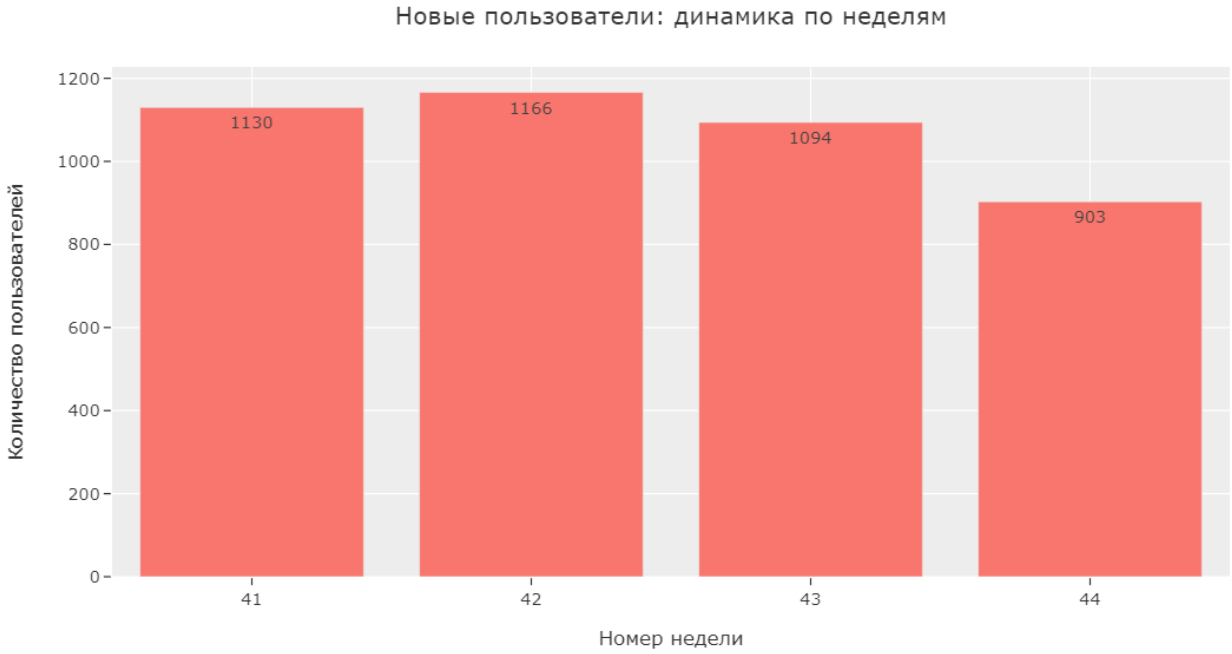
In [22]:
```
# Добавим в 'data' столбец с порядковым номером недели для каждого события:
data['week'] = data['date'].dt.isocalendar().week
# Сгруппируем таблицу по пользователям и найдем для каждого минимальную неделю:
first_week = data.groupby('user_id')['week'].min().reset_index()
first_week = first_week.rename(columns={'week':'first_week'})
first_week.head()
```

Out[22]:

	user_id	first_week
0	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	41
1	00157779-810c-4498-9e05-a1e9e3cedf93	42
2	00463033-5717-4bf1-91b4-09183923b9df	44
3	004690c3-5a84-4bb7-a8af-e0c8f8fca64e	42
4	00551e79-152e-4441-9cf7-565d7eb04090	43

Получили таблицу 'first_week', которая содержит данные о номере первой недели активности для каждого пользователя. Можем заодно посмотреть **сколько пользователей пришло к нам в каждую из исследуемых недель**

In [23]:
```
week_new_users = first_week.groupby('first_week')['user_id'].count().reset_index()

fig = px.bar(week_new_users, x='first_week', y='user_id', text = 'user_id')
fig.update_layout(title='Новые пользователи: динамика по неделям',
                  xaxis = dict(tickvals = week_new_users['first_week']),
                  height=500, width=950)
fig.update_xaxes(title='Номер недели')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```



Больше всего первых сессий совершено в 42-ю неделю.

Видим спад числа новых пользователей на 43 и 44 неделе. Команде, отвечающей за привлечение клиентов, необходимо обратить внимание, что в 44-ю неделю свои первые сессии начали только 903 пользователя - ожидаем ли такой отток (например, связан с сезонностью) или стоит более внимательно изучить этот вопрос?

Продолжим изучать изучать удерживание пользователей: `рассчитаем лайфтаймы в неделях по каждому событию`

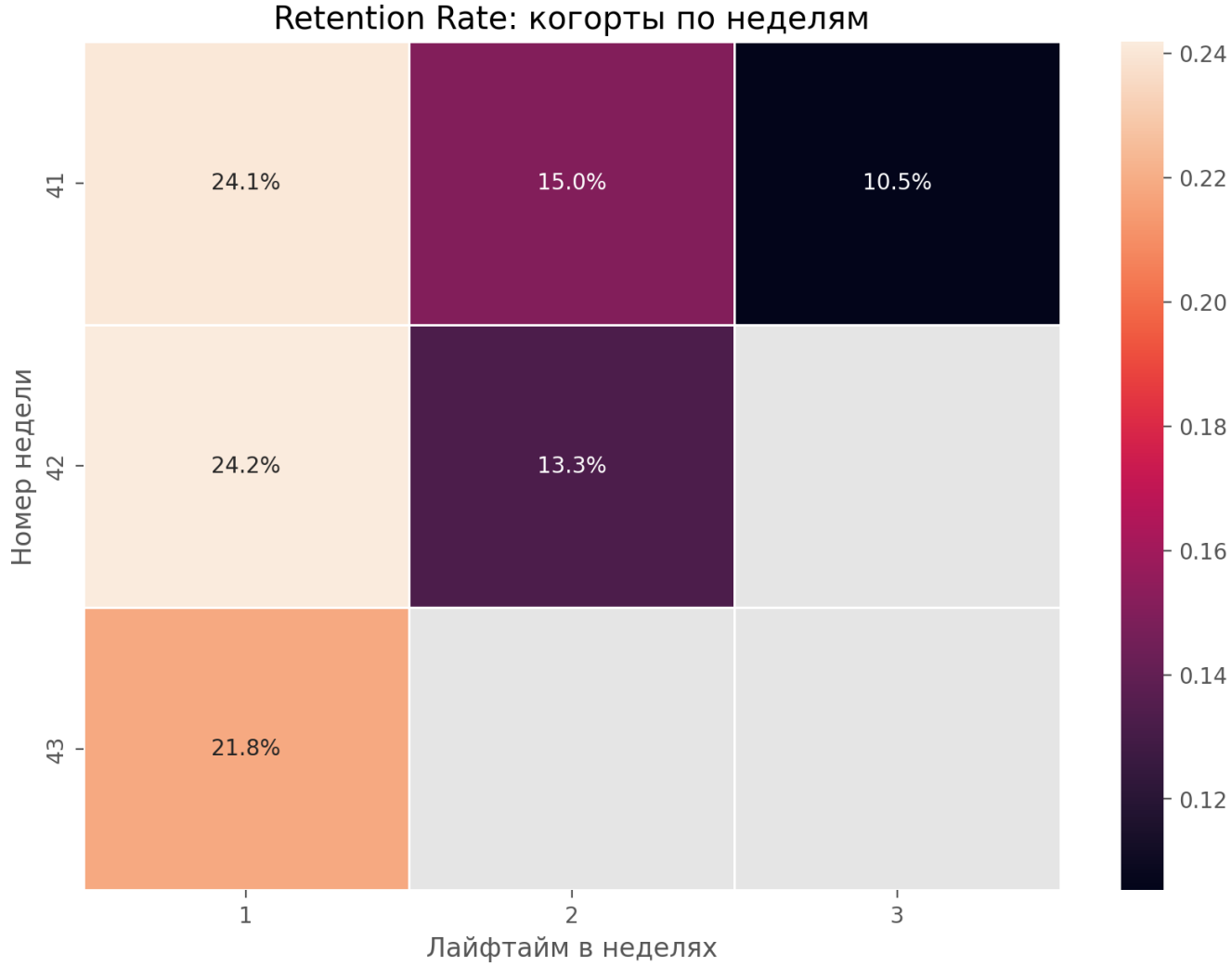
In [24]:
```
# В 'week_result_raw' сохраним все данные из 'data', добавим столбец с первой неделей:
week_result_raw = data.merge(first_week[['user_id', 'first_week']], on='user_id', how='left')
# Лайфтайм в неделях для каждого события:
week_result_raw['lifetime'] = week_result_raw['week'] - week_result_raw['first_week']
```

In [25]:
```
# Сводная таблица: группируем по первой неделе, для каждого лайфтайма подсчитываем кол-во уни
week_result_grouped = week_result_raw.pivot_table(index='first_week', columns='lifetime', val
```

In [26]:
```
# применяем div к столбцу с нулевой неделей, удаляем этот столбец
week_result_grouped = week_result_grouped.div(
    week_result_grouped[0], axis=0
).drop(columns=[0])
```

In [27]:
```
#Удалим строку с 44-й неделей - по ней данные еще "не созрели":
week_result_grouped = week_result_grouped.drop([44])
```

In [28]:
```
plt.figure(figsize=(10,7))
plt.title('Retention Rate: когорты по неделям')
sns.heatmap(week_result_grouped, annot=True, fmt='.1%', linewidths=1)
plt.xlabel('Лайфтайм в неделях')
plt.ylabel('Номер недели');
```



Для построения графика исключили когорту 44-й недели - пользователи "прожили" только одну неделю, для них еще не собралось достаточно данных.

Основные наблюдения:

- Когорты отличаются по проценту удержания друг у друга
- У первых двух когорт удержание на вторую неделю жизни практически одинаковое: ~**24%**. **Третья когорта удерживается чуть хуже: 21.8%** клиентов продолжает использовать приложение на второй "неделе жизни"
- **На третьей неделе** прододжают использовать приложение **15%** пользователей первой и **13.3%** второй когорт
- Данные на 4-ю неделю сформированы только для когорты 41-й недели. **К 4-ой неделе мы теряем 89.5% клиентов** - только 10.5% продолжают использовать приложение

Отметим, что у нас небольшой временной диапазон, `для многих пользователей данные не успели накопиться в нужном объеме` : по неделям более-менее реально оценить мы можем только пользователей из первой когорты. **Все же посмотрим на удержание по дням**

Retention Rate: когорты по дням

Как уже отмечалось ранее, визуализация по когортам каждого дня будет сильно перегружена, в связи с этим малоифнормативна. К тому же получим много пустых значений, что тоже не упростит визуальное восприятие. **Построим визцализацию Retention Rate по дням по следующему принципу:**

- отберем пользователей, успевших "прожить" 3 недели
- зададим горизонт анализа в 14 дней

In [29]:
```
# Оставляем только пользователей, совершивших первые сессии в 41 и 42 недели (именно они успе
long_lived = week_result_raw.query('first_week == 41 or first_week == 42', engine='python')
long_lived_users = long_lived['user_id'].unique().tolist()
long_lived_data = data.query('user_id in @long_lived_users')
```

Собрали пользователей, для которых есть данные за три недели. Найдем для каждого из них даты первой сессии

In [30]:
```
long_lived_first_session = long_lived_data.groupby('user_id')['event_time'].min().reset_index
long_lived_first_session = long_lived_first_session.rename(columns={'event_time':'first_sessi
```

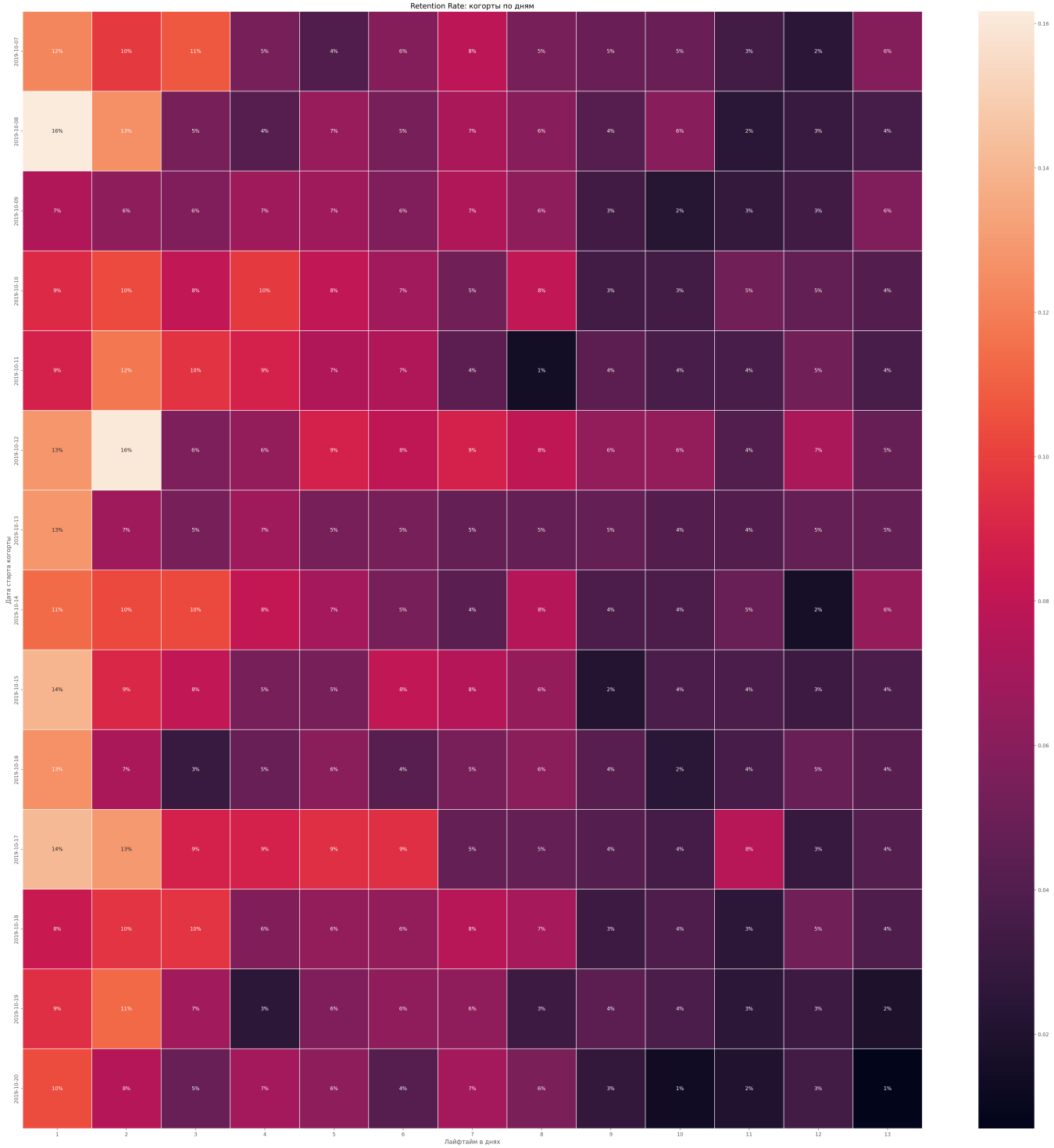
Даты первых сессий выделены, объединим таблицы в 'long_lived_result_raw' и высчитаем для каждой строки лайфтайм в днях

In [31]:
```
# добавляем столбец с первой сессией, выделяем из нее дату:
long_lived_result_raw = long_lived_data.merge(long_lived_first_session[['user_id', 'first_ses
long_lived_result_raw['first_date'] = long_lived_result_raw['first_session'].dt.strftime("%Y-

# Рассчитываем лайфтайм в днях для каждого события
long_lived_result_raw['lifetime'] = (long_lived_result_raw['event_time'] - long_lived_result_
# Оставляем события до 14-го лайфтайма включительно:
long_lived_result_raw = long_lived_result_raw.query('lifetime < 14')
```

In [32]:
```
# Группируем таблицу по дате, для каждого лайфтайма подсчитываем количество уникальных пользо
result_grouped = long_lived_result_raw.pivot_table(index='first_date', columns='lifetime', va
```

In [33]:
```
# Расчитываем процент удержания, удаляем столбец с нулевым лайфтаймом
result_grouped = result_grouped.div(
    result_grouped[0], axis=0
).drop(columns=[0])
```

In [34]:
```
plt.figure(figsize=(40, 40))
plt.title('Retention Rate: когорты по дням')
sns.heatmap(result_grouped, annot=True, fmt='.0%', linewidths=1)
plt.xlabel('Лайфтайм в днях')
plt.ylabel('Дата старта когорты');
```



- **Какая-либо общая тенденция удержания у когорт не выражена.** На второй день может вернутся как 16%, так и 7% пользователей
- **Когорты не удерживаются равномерно:** например, на 8-й лайфтайм может вернуться только 1% когорты, а на 12-й 5% из этой же когорты

В целом, вряд ли ожидается, что пользователи будут пользоваться приложением ежедневно. Для подобных сервисов логичнее собрать данные за бОльший срок и посмотреть, как пользователи ведут себя на протяжении, например, полугода.

По имеющимся данным можем выделить следующее:

- **уже на 4-й лайфтайм удержание не достигает 10% по всем когортам**
- удержание не убывает постепенно: на более поздний лайфтайм коэффициент может быть выше, чем на более ранний

Рекоммендуется определить желаемое поведение пользователей в соответствии с бизнес планом: какая частота действий ожидается от пользователей в неделю или месяц? На основании этих

ожиданий можно определить горизонт для анализа Retention Rate и рассмотреть удерживания для бОльшего временного промежутка.

Выводы: Retention Rate

- наблюдается отток новых пользователей: в 44 неделю впервые вопсользовалиси приложением 903 пользователя, в то в ремя как в 41-ю - 1130
- **Retention Rate по неделям:**
  - когорты удерживаются не равномерно, клэффициент удерживания на вторую неделю у трех когорт разный (24.1%, 24.2%, 21.8%)
  - по данным когорты 1-й недели: **на 4 неделю после первой сессии почти 90% клиентов перестает использовать приложение**
- **Retetion RAte по дням:**
  - удержание не спадает равномерно по когортам: в одной когорте на второй день может вернутся 16% новых клиентов, в другой только 7%
  - внутри когорт удержание тоже колеблется изо дня в день: внутри одной когорты на 9-й "день жизни" может вернутся 1% пользователей, а на 12-й - 5%
  - на 4-й лайфтайм ни у одной из когорт не наблюдается удержание >10%

Предполагаю, что от пользователей не ожидается ежедневное использование приложения, более информативно для бизнеса в этом случае рассмотреть удержание на бОльшем временном промежутке и оценить Retention Rate по месяцам и/или неделям.

Сессии. Анализ времени, проведенного в приложении

Объеденение действий пользователей в сессии

На этапе предобработки данных отметили, что `мы не располагаем данными о начале и окончании каждой сессии,` соответственно не знаем и не можем рассчитать длительность сессий.

Зная дату каждого события, можем самостоятельно объединить события в сессии и в дальнейшем рассчитать ее длительность. Поступим следующим образом:

- **отсортируем таблицу по id и времени события:** так мы для каждого пользователя выстроим порядковую последовательность по совершенным событиям
- **определим тайм-аут:** предположим, что если между двумя последовательным действиями пользователя прошло более **30 минут** - события относятся к разным сессиям, если меньше - соответственно, события относятся к одной сессии
- **каждой сесии присвоим уникальный id**
- **для каждой сессии выделим минимальную и максимальную даты:** разница между ними и будет длительностью сессии

In [35]:
```
data = data.sort_values(['user_id', 'event_time'])
g = (data.groupby('user_id')['event_time'].diff() > pd.Timedelta('30Min')).cumsum()
data['session_id'] = data.groupby(['user_id', g], sort=False).ngroup() + 1
data.head(20)
```

Out[35]:

	event_time	event_name	user_id	source	date	weekday	week	session_id
805	2019-10-07 13:39:45	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
806	2019-10-07 13:40:31	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
809	2019-10-07 13:41:05	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
820	2019-10-07 13:43:20	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
830	2019-10-07 13:45:30	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
831	2019-10-07 13:45:43	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
832	2019-10-07 13:46:31	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
836	2019-10-07 13:47:32	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
839	2019-10-07 13:49:41	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-07	0	41	1
6541	2019-10-09 18:33:55	map	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-09	2	41	2
6546	2019-10-09 18:35:28	map	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-09	2	41	2
6565	2019-10-09 18:40:28	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-09	2	41	2
6566	2019-10-09 18:42:22	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-09	2	41	2
36412	2019-10-21 19:52:30	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36416	2019-10-21 19:53:17	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36419	2019-10-21 19:53:38	map	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36421	2019-10-21 19:54:45	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36423	2019-10-21 19:54:56	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36430	2019-10-21 19:56:49	map	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3
36435	2019-10-21 19:57:21	tips_show	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	other	2019-10-21	0	43	3

Получили новый столбец `'session_id'`: выше можем оценить, что действия одного пользователя разделены на три разные сессии, разбивка работает корректно.

Создадим таблицу 'session', в которой соберем необходимые нам данные из 'data' и расчитаем длительность каждой сессии

In [36]:
```
'''Сначала создадим "сырую" таблицу. Индексами будут значения из 'session_id',
т.к. data отфильтрована по пользователю и времени события, можем сразу добавить для каждой се
даты события, user_id, дата и день недели внутри одной сессии будут одинаковыми, поэтому прос
```

```
Дополнительно посчитаем количество событий внутри каждой сессии
'''
session_raw = data.pivot_table(index='session_id', aggfunc={'event_time':['min', 'max'],
                                                            'user_id':'first',
                                                            'event_name':'count',
                                                            'date':'first',
                                                            'weekday':'first'}).reset_index()


session_raw = session_raw.set_index('session_id')
session_raw.columns = session_raw.columns.droplevel()
session_raw.columns = ['date', 'event_count', 'end', 'start', 'user_id', 'weekday']
display(session_raw.head())
print('Количество сессий:', len(session_raw))
```

session_id	date	event_count	end	start	user_id	weekday
1	2019-10-07	9	2019-10-07 13:49:41	2019-10-07 13:39:45	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	0
2	2019-10-09	4	2019-10-09 18:42:22	2019-10-09 18:33:55	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	2
3	2019-10-21	14	2019-10-21 20:07:30	2019-10-21 19:52:30	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	0
4	2019-10-22	8	2019-10-22 11:30:52	2019-10-22 11:18:14	0001b1d5-b74a-4cbf-aeb0-7df5947bf349	1
5	2019-10-19	9	2019-10-19 21:59:54	2019-10-19 21:34:33	00157779-810c-4498-9e05-a1e9e3cedf93	5

```
Количество сессий: 10368
```

`Собрали в 'session_raw' следующие данные:`

- id сессии
- дату сессии
- количество событий в сессии
- время старта и окончания
- id пользователя
- день недели

Всего наши события разбились на 10368 сессий

Теперь можем рассчитать длительность каждой сессии. Также `извлечем час` из столбца 'start' - проанализируем, в какие часы клиенты чаще всего пользуются приложением

In [37]:
```
# рассчитваем длительность в секундах
session_raw['duration_sec'] = (session_raw['end'] - session_raw['start']).dt.total_seconds().
# рассчитваем длительность в минутах
session_raw['duration_min'] = round(session_raw['duration_sec'] /60).astype('int')
# извлекаем час из даты старта
session_raw['hour'] = session_raw['start'].dt.hour
```

Все необходимые столбцы добавили, перезапишем данные в итоговую таблицу **'sessions'** и передадим ей более удобный порядок столбцов

In [38]:
```
sessions = session_raw[['date', 'weekday', 'hour', 'start', 'end', 'event_count', 'duration_s
```

`Таблица готова, приступим к анализу сессий.`

Предлагаю проанализировать выделенные сессие в следующем ключе:

1. Длительность сессий

- посмотрим на размах данных, проанализируем выбросы
- выявим наиболее типичную длительность сессии

2. Сессии в разрезе даты и времени

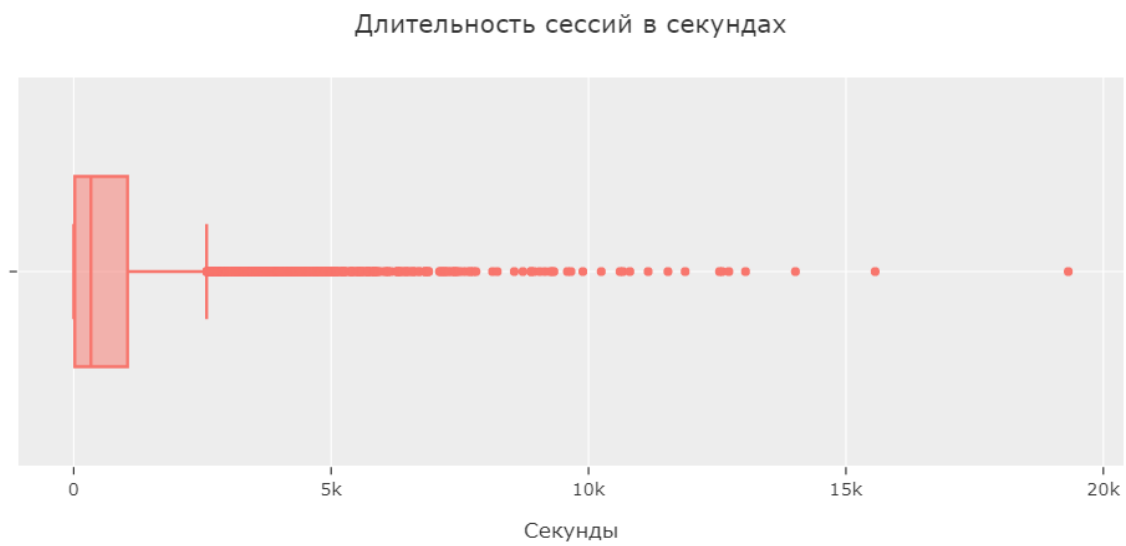
- посмотрим, как количественно распределены сессии по дням
- выявим, есть ли разница между количеством сессий в будни и выходные дни, наблюдается ли разница в длительности сессий
- определим "топ-часы" по количеству сессий: в какие часы пользователи чаще всего используют приложение

3. Сессии по пользователям

- определим, сколько сессий обычно соврешает один пользователь за весь период

Длительность сессий

In [39]:
```
fig = px.box(sessions, x='duration_sec')
fig.update_layout(title='Длительность сессий в секундах',
                  height=400, width=900)
fig.update_xaxes(title='Секунды')
fig.update_yaxes(title='')
fig.show()
```



Медианное значение 335 секунд. Нижняя граница ящика упирается в 0 - в нашем наборе данных достаточно много "нулевых" сессий. Видим большое количество выбросов - экстремально долгие сессии.

Почему мы видим сессии с нулевой длительностью?

Как уже несколько раз отмечалось ранее, в предоставленных нам датасетах `нет длительности, даты старта и окончания сессий.` Эти метрики мы рассчитывали самостоятельно исходя из начала `события`. Если за сессию произошло `только одно событие,` то дата начала этого события является одновременно и минимальной, и максимальной датой, другими словами, нам не из чего высчитать разницу и мы не можем рассчитать длительность.

Посмотрим на сессии длительностью более 90 минут: нет ли каких-то в них аномалий? Как много действий совершается в такие длительные сессии?

In [40]:
```
sessions.query('duration_sec > 5400')['event_count'].describe()
```

Out[40]:
```
count    112.000000
mean      45.339286
std       26.613873
min        8.000000
25%       29.000000
50%       38.500000
75%       53.250000
max      149.000000
Name: event_count, dtype: float64
```

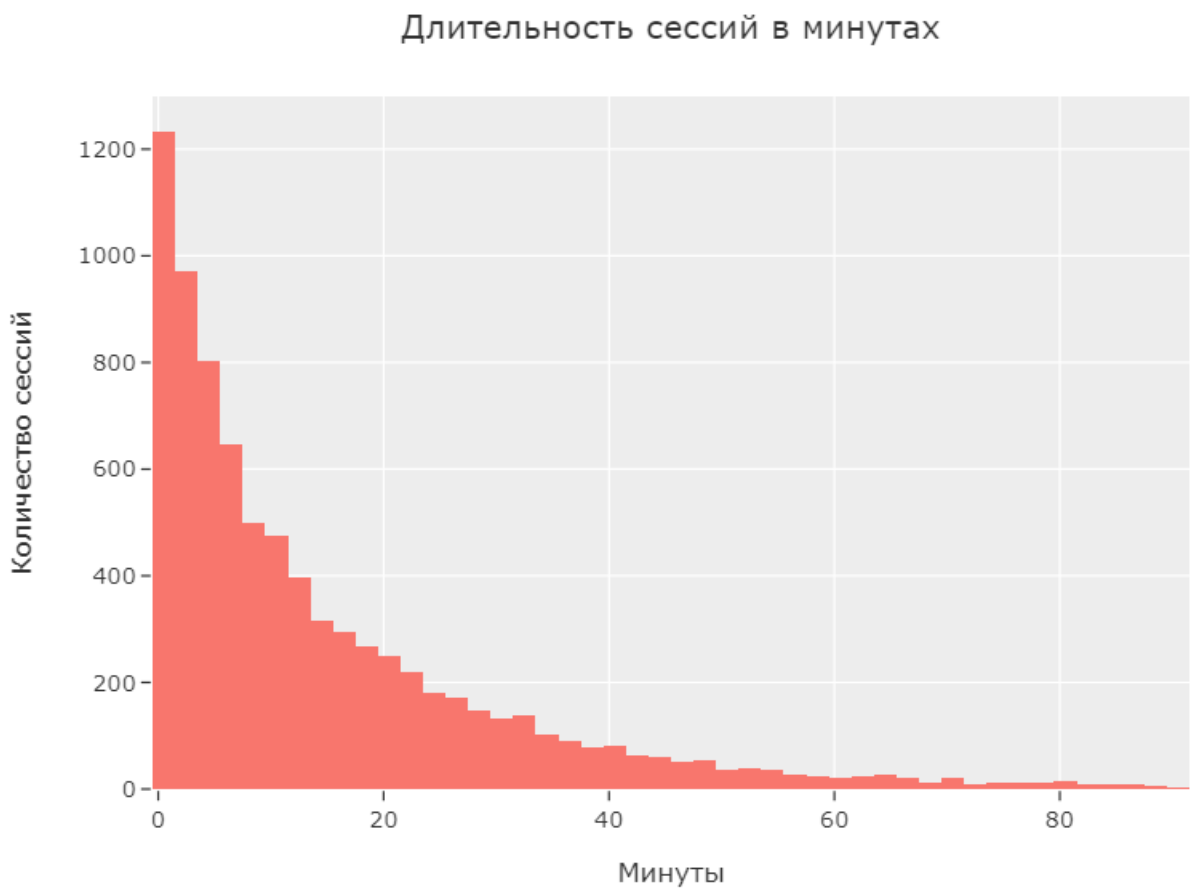
Всего 112 экстремально долгих сессий. Минимальное количество действий: 8, медианное: 38.5, среднее: 45.4, максимальное: 149. `Похоже, что это действительно "сверхактивные пользователи".`

Определили, что аномально длинные сесии имеют право на существование, но все же не типичны для нашего набора данных, Для построения графика срежем их, "нулевые" тоже исключим - для этих сессий невозможно вычислить длительность.

Определим сколько **чаще всего длится одна сессия в минутах** (в минутах воспринимать инфорацию проще, чем в секундах)

In [41]:
```
print('Количество сессий после среза аномалий:', len(sessions.query('5400 > duration_sec > 0'
print('Медиана по срезанным данным:', sessions.query('5400 > duration_sec > 0')['duration_min

fig = px.histogram(sessions.query('5400 > duration_sec > 0'), x='duration_min', nbins=90)
fig.update_layout(title='Длительность сессий в минутах')
fig.update_xaxes(title='Минуты')
fig.update_yaxes(title='Количество сессий')
fig.show()
```

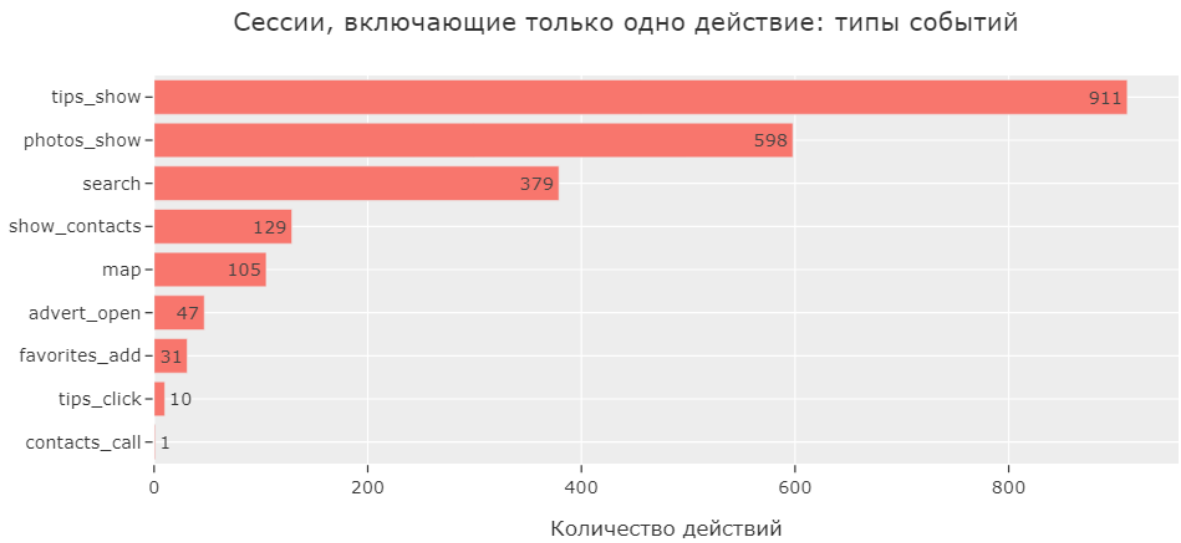
```
Количество сессий после среза аномалий: 8111
Медиана по срезанным данным: 9.0
```



- **Чаще всего одна сессия длится до 1 минуты**
- Медианная длительность сессии: 9 минут
- Отбросив сессии без длительности и экстремально длинные сессии:
  - 37% сессий длятся до 5 минут
  - 51% сессий длятся до 9 минут
  - 65% сессий длятся до 15 минут
- По всем сессиям:
  - сверхдолгие сессии (более 90 минут) составляют 1.1%

Дополнительно посмотрим, какие события совершены в "нулевые" сессии (для которых мы не можем рассчитать длительность)

In [42]:
```
null_duration_sessions_list = sessions.query('duration_sec == 0').index.tolist()
null_duration_sessions= data.query('session_id in @null_duration_sessions_list')['event_name'
                .value_counts().to_frame().sort_values(by='event_name')

fig = px.bar(null_duration_sessions, x='event_name', text='event_name')
fig.update_layout(title='Сессии, включающие только одно действие: типы событий',
                  height=400, width=900)
fig.update_xaxes(title='Количество действий')
fig.update_yaxes(title='')
fig.show()
```



Большое количество действий 'tip_show' ожидаемо - пользователь мог открыть приложение, увидеть рекомендацию и закончить сессию. **Удивляет наличие всех остальных событий, включая целевое.** Судя по данным, пользователь может в одно действие открыть карту, или посмотреть фото только открыв приложение.

Это сигнализирует нам о следующем:

- либо нам предоставили не все события пользователей
- либо у нас пропущены какие-то определенные типы событий: например, 'login', после которого можно открыть поиск или карту. Так же заметим, что у нас нет события "зашел в избранное" - вероятно, посмотреть фото, контакты, открыть карточку можно из него.

Сессии в разрезе даты и времени

В этом блоке ответим на следующие вопросы:

- Сколько происходит сессий каждый день?

- Отличаются ли будни и выходные дни по количеству и длительности сессий?
- В какие часы суток происхоит больше всего сессий?

Количество и длительность сессий по дням

Сгруппируем сессии по дате и посчитаем их количество в каждый из дней

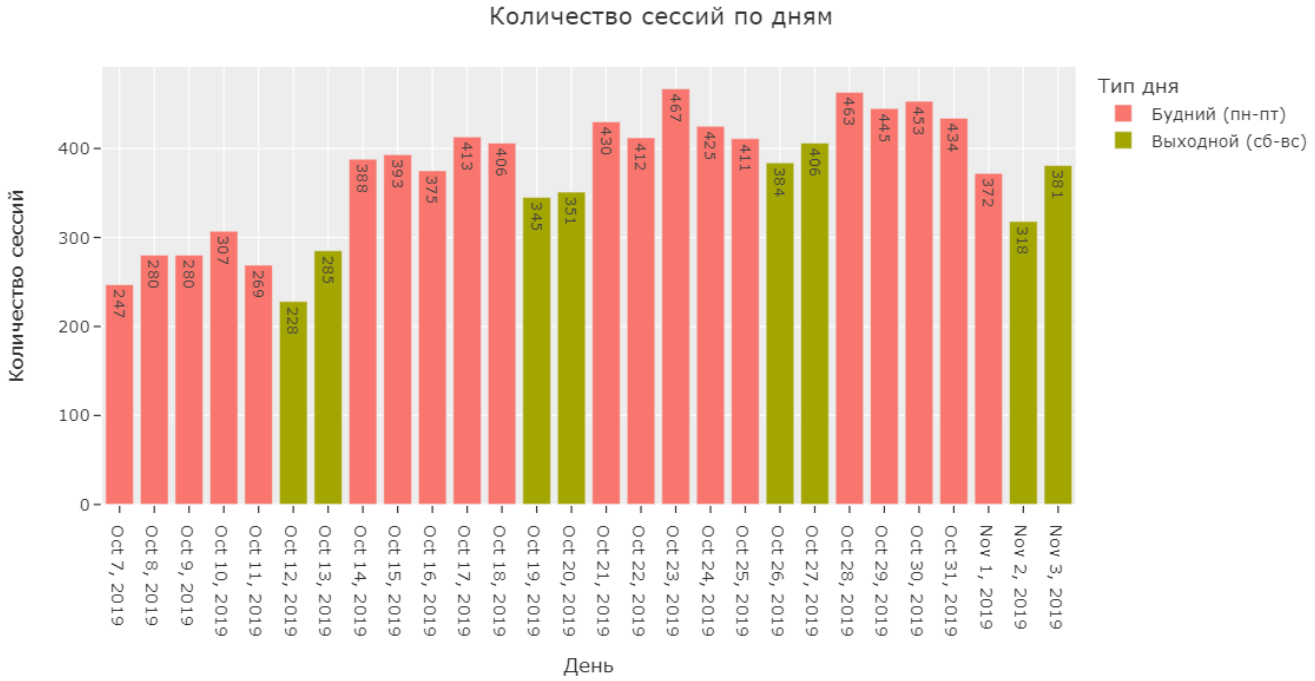
Здесь мы не будем исключать сессии с нулевой длительностью, т.к. нас интересует сам факт начала сессии, а не ее длительность

In [43]:
```
'''Функция, принимает значение строки,
возвращает True - если день относится к выходному, False - к буднему'''
def weekend(row):
    if row in [5,6]:
        return True
    else:
        return False

sessions['weekend'] = sessions['weekday'].apply(weekend)
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_7184\1821922071.py:9: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
```

In [44]:
```
sessions_date_cnt = sessions.groupby('date').agg({'user_id':'count', 'weekend':'first'})
```

In [45]:
```
fig = px.bar(sessions_date_cnt,
             y='user_id',
            color = 'weekend',
            text='user_id')
fig.update_layout(title='Количество сессий по дням',
                  yaxis_title='Количество сессий',
                  xaxis_title='День',
                 height=500, width=950,
                xaxis = dict(tickvals =sessions_date_cnt.index),
                legend=dict(title= 'Тип дня'))
newnames = {'True':'Выходной (сб-вс)', 'False': 'Будний (пн-пт)'}
fig.for_each_trace(lambda t: t.update(name = newnames[t.name]))
fig.show()
```

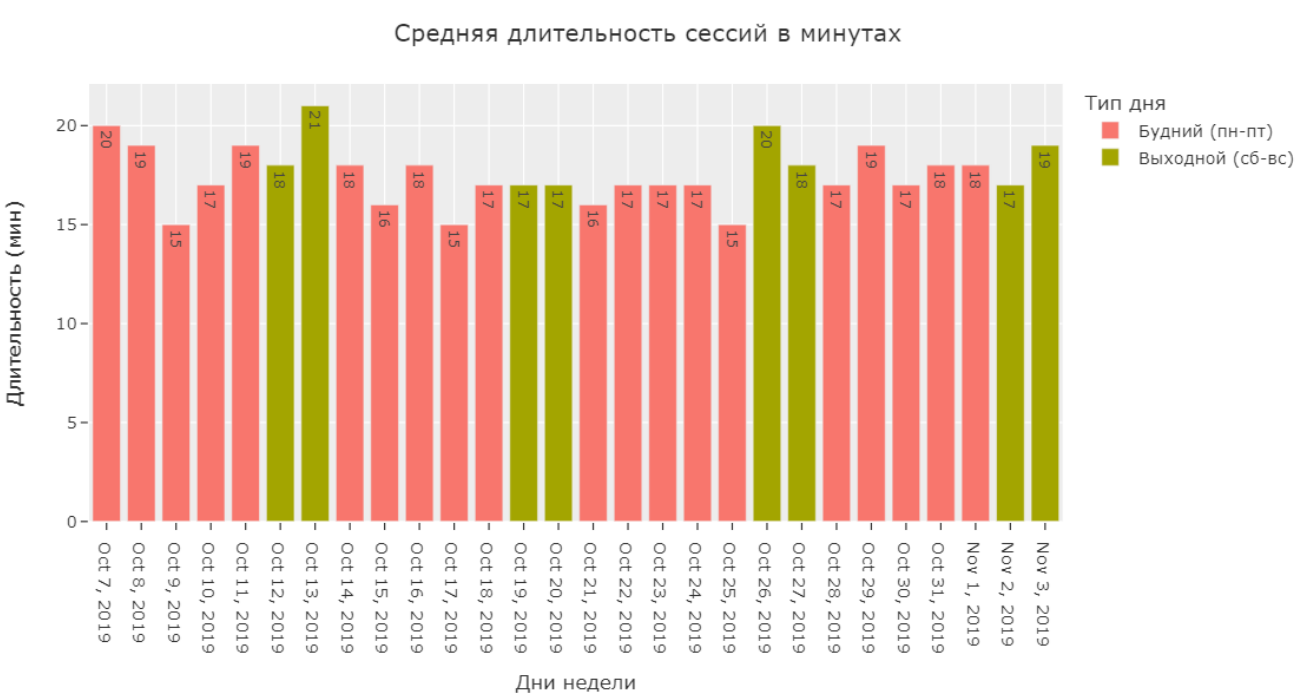


- **начиная с 14 октября количество сессий не опускается ниже 350 сессий в день** (за исключаением одного дня), в то время как в период 7-13 октября количество сессий в день не превышало 300 (кроме 10 октября с 307 сессиями). **Пользователи ведут себя активнее, чем в начале изучаемого периода**
- В выходные количество сессий обычно меньше, чем в будние дни (внутри одной недели)

Посмотрим, отличаются ли как-то дни по длительности сессий.

In [46]:
```
sessions_day_duration = sessions.query('duration_min > 0').groupby('date')\
    .agg({'duration_min':'mean', 'weekend':'first'}).reset_index()
sessions_day_duration['duration_min'] = round(sessions_day_duration['duration_min'])
```

In [47]:
```
fig = px.bar(sessions_day_duration,
             x ='date',
             y='duration_min',
            text='duration_min',
            color = 'weekend')

fig.update_layout(title='Средняя длительность сессий в минутах',
                  yaxis_title='Длительность (мин)',
                  xaxis_title='Дни недели',
                 legend=dict(title= 'Тип дня'),
                 xaxis = dict(tickvals =sessions_date_cnt.index),
                 height=500, width=950)
newnames = {'True':'Выходной (сб-вс)', 'False': 'Будний (пн-пт)'}
fig.for_each_trace(lambda t: t.update(name = newnames[t.name]))
fig.show()
```



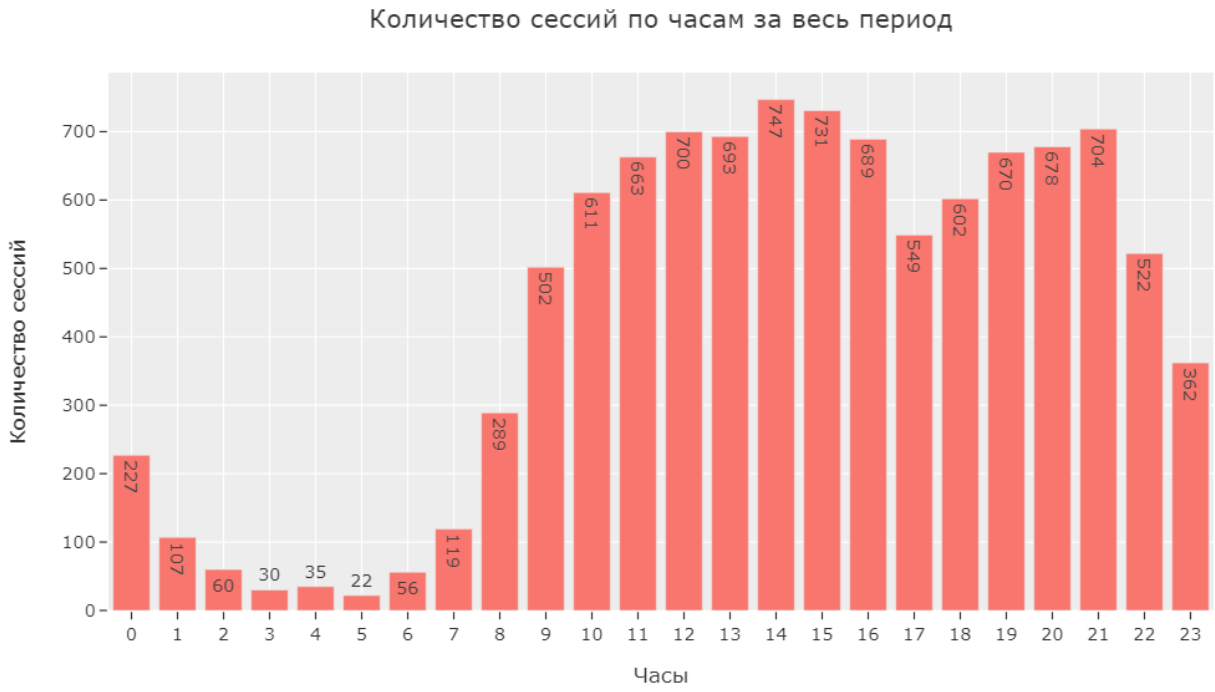
- успредненные данные по дням **не демоснтрируют какой-либо "общей логики"** - например, в разные недели понедельник может показывать как самый высокий, так один из самых низких результатов
- все средние знаечения лежат **в диапазоне 15-21 минута**
- **в выходные средняя длительность сессии выше,** чем в будние дни (либо на том же уровне)

В какое время чаще всего пользуются приложением?

Определим в какие часы пользователи чаще начинают сессии, `в какое время они чаще всего активны в приложении`. Сгруппируем сессии по часам и посчитаем количество сессий.

In [48]:
```
sessions_hours_cnt = sessions.groupby('hour')['user_id'].count().reset_index()
```

In [49]:
```
fig = px.bar(sessions_hours_cnt,
             x='hour',
             y='user_id',
            text='user_id')

fig.update_layout(title='Количество сессий по часам за весь период',
                  yaxis_title='Количество сессий',
                  xaxis_title='Часы',
                 xaxis = dict(tickvals = sessions_hours_cnt['hour']),
                 height=500, width=900)

fig.show()
```



Наблюдается небольшая активность в период 00:00-01:00, в ночнее время сессий, очевидно, ничтожно мало. "Просыпаются" пользователи в период 09:00-10:00 - за 4 недели в этот период совершено 502 сессии.

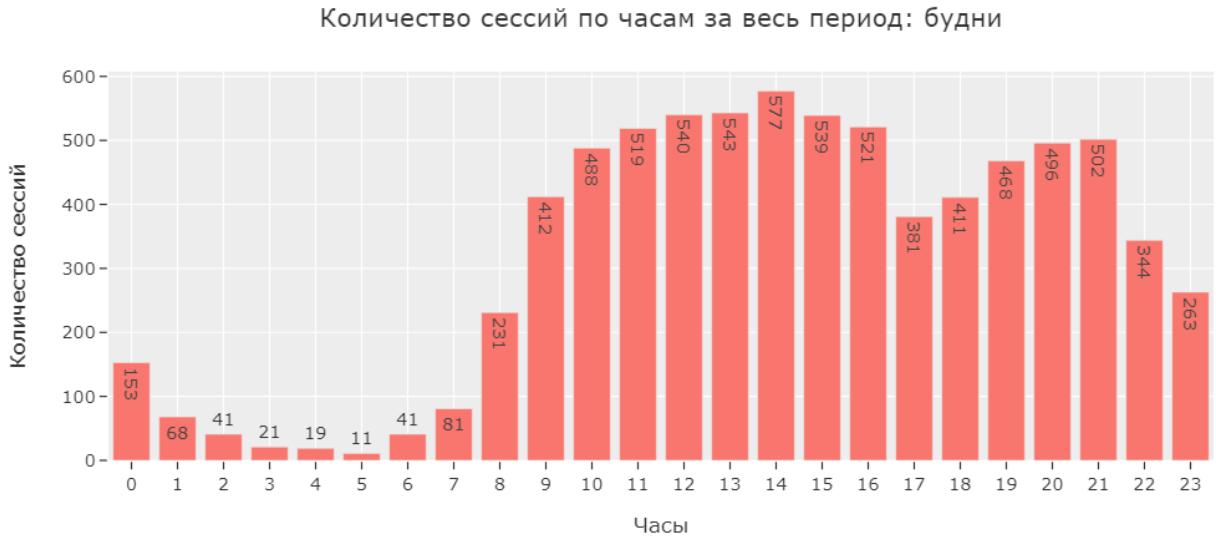
Чаще всего наши пользователи заходят в приложение:

- днем, **14:00-16:00** - самое популуряное дневное время. В целом можно выделить промежуток 12:00-17:00 - одни из самых популярных часов по количеству сессий
- вечером, **21:00-22:00** - самое популярное вечернее время.

Мы уже выявили, что количество и длительность сессий отличаются в зависимости от типа дня, логично проверить, `отличаются ли и  популярные часы в рабочие и выходные дни`

In [50]:
```
# сохраняем сессии будних и выходных дней в две отдельные таблицы
sessions_workday_hours = sessions.query('weekend == False').groupby('hour')['user_id'].count(
sessions_weekend_hours = sessions.query('weekend == True').groupby('hour')['user_id'].count()
```

In [51]:
```
fig = px.bar(sessions_workday_hours,
             x='hour',
             y='user_id',
            text='user_id')
fig.update_layout(title='Количество сессий по часам за весь период: будни',
                  yaxis_title='Количество сессий',
                  xaxis_title='Часы',
                 xaxis = dict(tickvals = sessions_workday_hours['hour']),
                 height=400, width=900)
fig.show()

fig = px.bar(sessions_weekend_hours,
             x='hour',
             y='user_id',
            text='user_id')
fig.update_layout(title='Количество сессий по часам за весь период: выходные',
                  yaxis_title='Количество сессий',
                  xaxis_title='Часы',
                 xaxis = dict(tickvals = sessions_weekend_hours['hour']),
                 height=400, width=900)
fig.show()
```



Будни:

- заметная активность начинается в **9:00**
- самый популярный период: **13:00-15:00**, в целом популярен период 12:00-16:00
- вечером основная активность происходит в период **20:00-22:00**

Выходные:

- заметная активность начинается в **10:00**
- самые популярный период: вечер, **19:00-20:00, 21:00-22:00**
- дневная активность выражена слабее вечерней, днем самый популярный час **15:00-16:00**

Эту информацию можно использовать для определения времени рассылки пуш-уведомлений и прочих "напоминалок" либо для каких-то иных маркетинговых целей

Количество сессий на одного пользователя

Напоследок выясним, сколько сессий обычно совершает один пользователь. Сгруппируем сессии по пользователям и посчитаем их количество по каждому.

In [52]:
```
user_sessions = sessions.groupby('user_id')['date'].count().reset_index()
user_sessions = user_sessions.rename(columns={'date':'sessions_total'})
user_sessions['sessions_total'].describe()
```

Out[52]:
```
count    4293.000000
mean        2.415094
std         3.536466
min         1.000000
25%         1.000000
50%         1.000000
75%         3.000000
max        99.000000
Name: sessions_total, dtype: float64
```

При построении графика исключим аномально активного пользователя с 99 сессиями

In [53]:
```
fig = px.histogram(user_sessions.query('sessions_total!=99'), x='sessions_total', nbins=60)
fig.update_layout(title='Количество сессий на одного пользователя')
fig.update_xaxes(title='Количество сессий')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```

К сожалению, чаще всего **пользователи совершают одну сессию** (53.6% от всех пользователей). 20% пользователей совершило 2 сессии, 10.7% - 3 сессии, большее число сессий - редкость.

Встречаются крайне активные пользователи с числом сессий >20

Выводы: сессии

События разбиты на 10368 сессий на основании таймаута 30 минут

- **Длительность сессии:**
  - медианная длительность: 9 минут
  - 15% сессий длятся до 1 минуты
  - 7% сессий длятся до 5 минут
  - 66% сессий длятся до 15 минут
  - встречаются аномально долгие сессии, более 90 минут (всего 1.1% от всех сессий)
- **Сессии по дате и времени**:

    - выросло количество сессий по сравнению с первой неделей: **в день пользователями совершается больше 350 сессий, в пиковые дни - больше 460**
    - средняя длительность сессии колеблется изо дня в день от 15 до 21 минут, общей тенденции по неделям не наблдюдается
    - в выходные происходит меньше сессий. чем в будни, при этом длительность сессий в сб-вс либо находится на том же уровне, либо выше, чем в будние дни
    - **пиковые часы:**
        - Будни:
            - первая заметная активность: 9:00
            - день: 13:00-15:00 (самый поплярный период
            - вечер: 20:00-22:00
        - Выходные:
            - первая заметная активность: 10:00
            - день: 15:00-16:00
            - вечер: 19:00-20:00, 21:00-22:00 (вечер популрянее дня)
- **Чаще всего пользователи совершают одну сессию (53.6% всех пользователей)**
    - 31% совершают 2-3 сессии

События. Анализ частоты действий

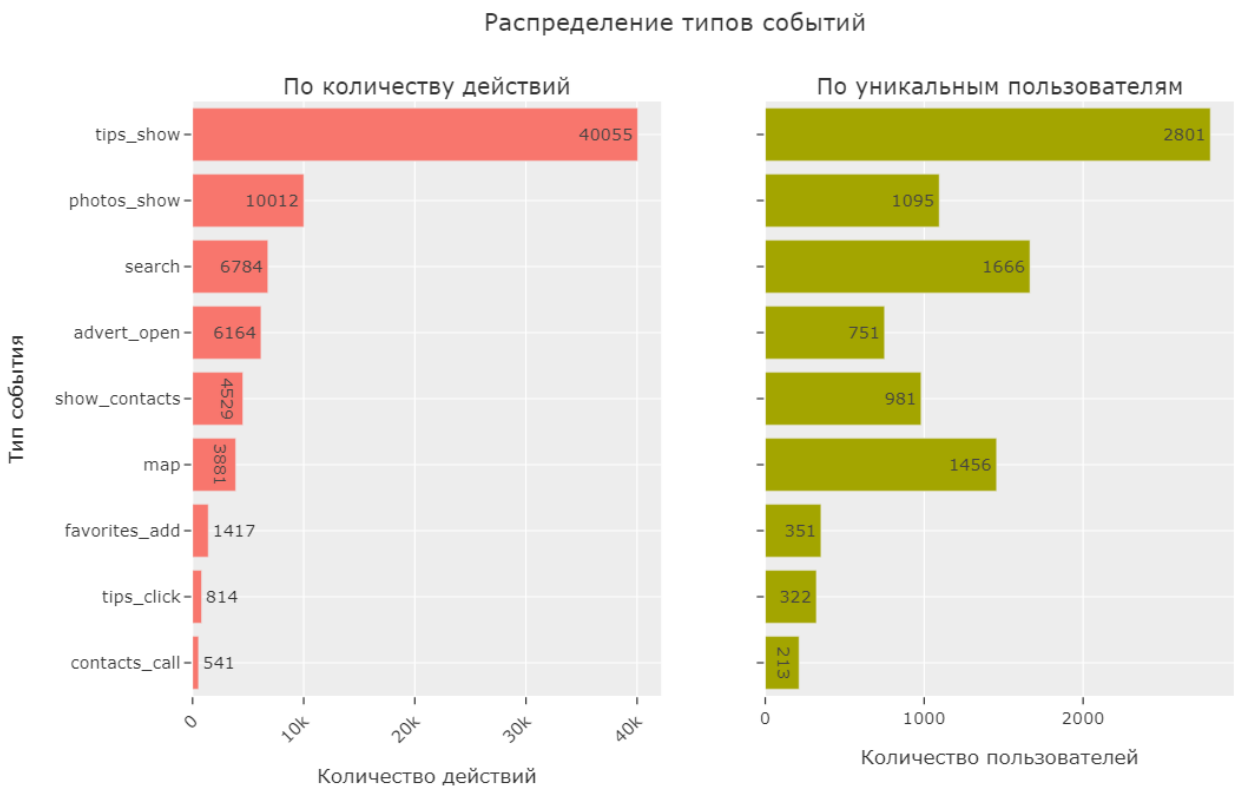
В этом блоке детальнее изучим действия/события:

- Сколько раз произошло каждое из событий? Сколько пользователй совершило каждое из событий?
- Сколько действий обычно совершает пользователь за исследуемый период?
- Как распределены события по дням? Сколько из них целевых?
- Сколько действий обычно происходит за сессию?

Количество событий за весь период

In [54]:
```
event_cnt = data.groupby('event_name').agg({'session_id':'count', 'user_id':'nunique'}).reset
event_cnt = event_cnt.rename(columns={'session_id':'total'}).sort_values(by='total')
```

In [55]:
```
fig = make_subplots(rows=1, cols=2, shared_yaxes=True,
                    subplot_titles=['По количеству действий', 'По уникальным пользователям'])

fig.add_trace(go.Bar(y=event_cnt['event_name'], x=event_cnt['total'], orientation='h',
                     text=event_cnt['total'],
                    ),
              1, 1)

fig.add_trace(go.Bar(y=event_cnt['event_name'], x=event_cnt['user_id'],  orientation='h',
                       text=event_cnt['user_id'], ),
              1, 2)

fig.update_layout(xaxis_tickangle=-45, title='Распределение типов событий',
                  title_x = 0.5,showlegend=False, height=600, width=950)
fig.update_yaxes(title='Тип события', col=1, row=1)
fig.update_yaxes(title='', col=2, row=1)
fig.update_xaxes(title='Количество действий', col=1, row=1)
fig.update_xaxes(title='Количество пользователей', col=2, row=1)
fig.show()
```



Чаще всего происход событие **tips_show** - пользователь видит рекоммендованное объявление. Это событие можем охарактеризовать как "автоматическое" - оно не требует от пользователя совершения какого-либо действия. **Целевое действие совершено 4529 раз - его совершил 981 уникальный пользователь**

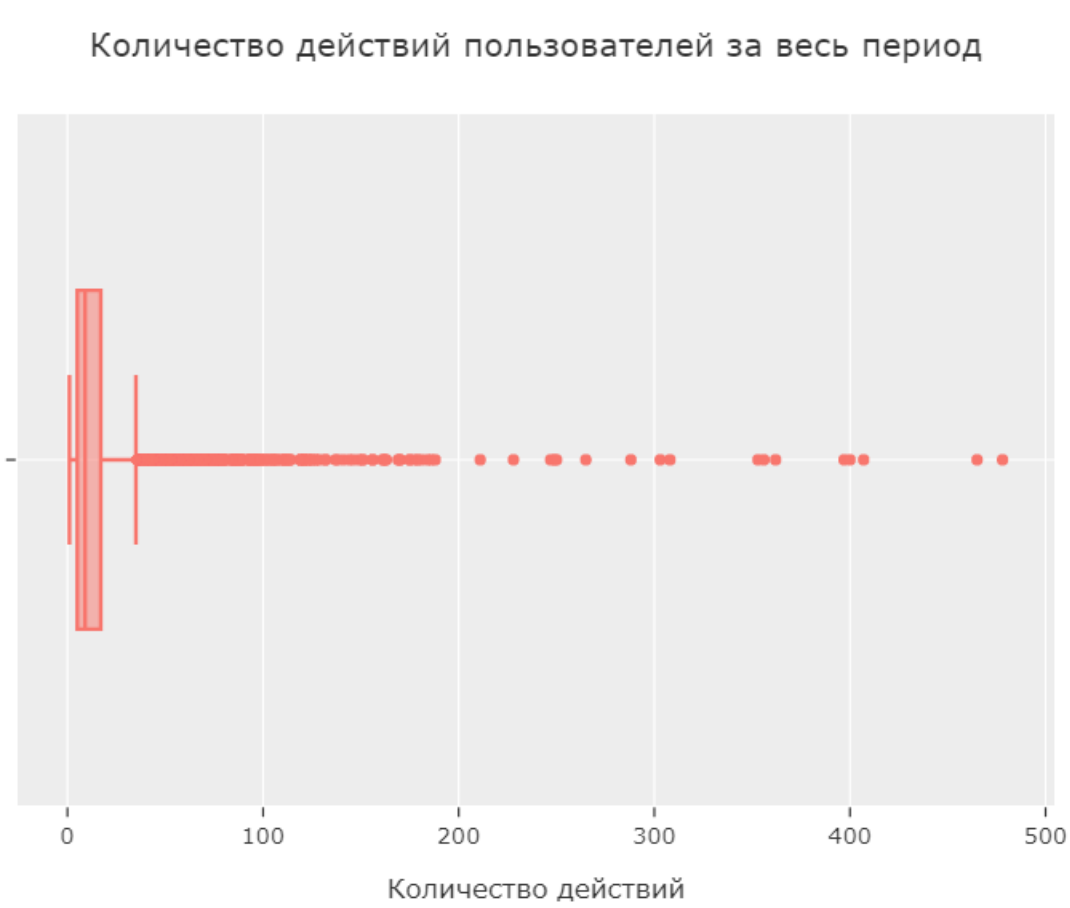
Количество событий по пользователям

Сгруппируем данные по пользователям и сосчитаем количество своершенных ими действий за всесь период.

In [56]:
```
user_event_cnt = data.groupby('user_id')['event_name'].count().reset_index()
```

In [57]:
```
print('Всего за исследуемый период совершено', user_event_cnt['event_name'].sum(), 'действий'

fig = px.box(user_event_cnt, x='event_name')
fig.update_layout(title='Количество действий пользователей за весь период')
fig.update_xaxes(title='Количество действий')
fig.update_yaxes(title='')
fig.show()
```

```
Всего за исследуемый период совершено 74197 действий
```



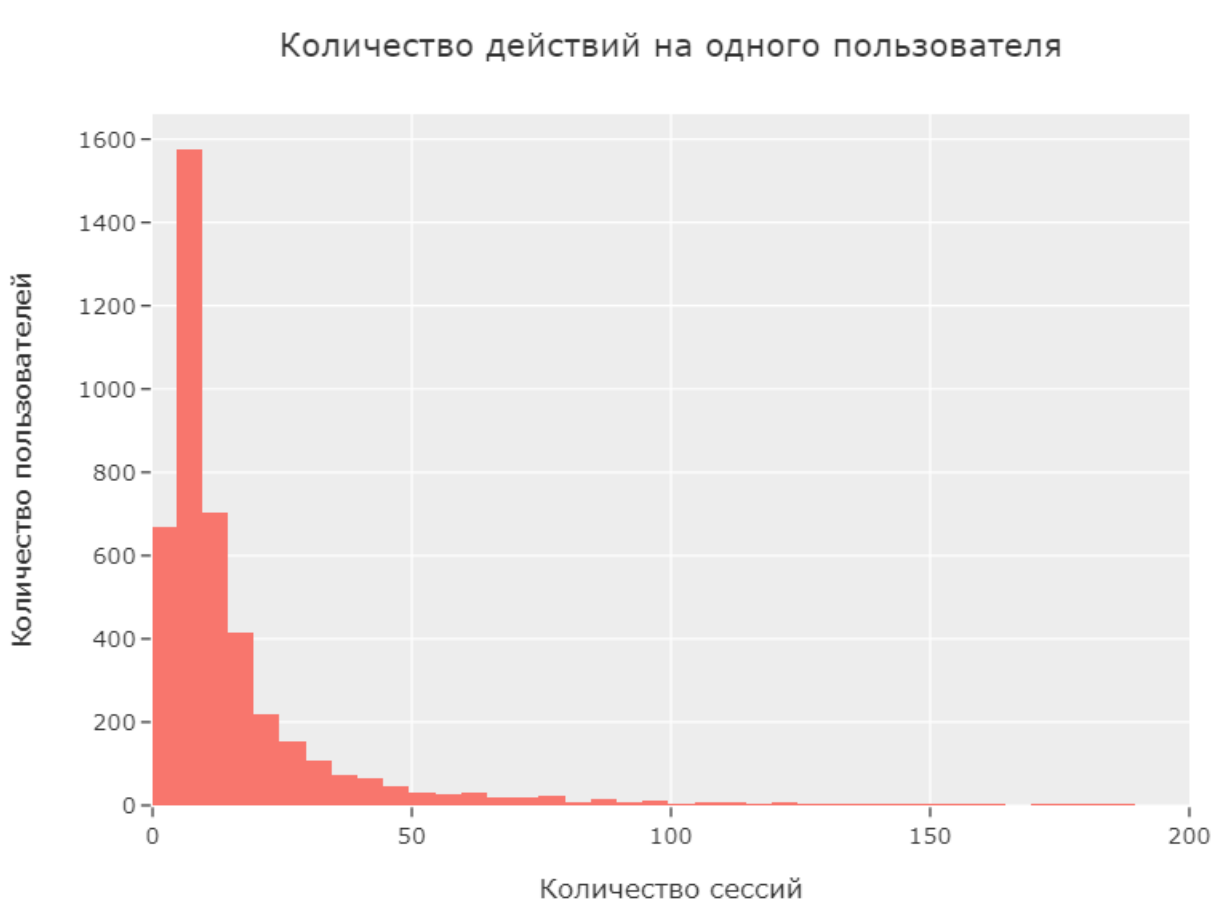
- Минимум пользователи за весь период совершают одно действие
- **Медианное значение: 9**
- Большинство значений лежит в диапазоне **1-35 действий** за весь период

Мы уже и раньше наблюдали "сверхактивных" пользователей - тех, кто совершил очень много сессий, тех, чьи сессии длятся необыкновенно долго. Теперь ожидаемо видимо, что есть пользователи, которые успели совершить более 200 событий за исследуемый период.

Построим гистрограмму, ограничив ось с количество действий до 200, чтобы лучше рассмотреть поведение обычных, не настолько активных пользователей

In [58]:
```
print('Пользователей, совершивших больше 100 действий:', len(user_event_cnt.query('event_name

fig = px.histogram(user_event_cnt, x='event_name', nbins=200, range_x=[0,200])
fig.update_layout(title='Количество действий на одного пользователя')
fig.update_xaxes(title='Количество сессий')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```

```
Пользователей, совершивших больше 100 действий: 76
```



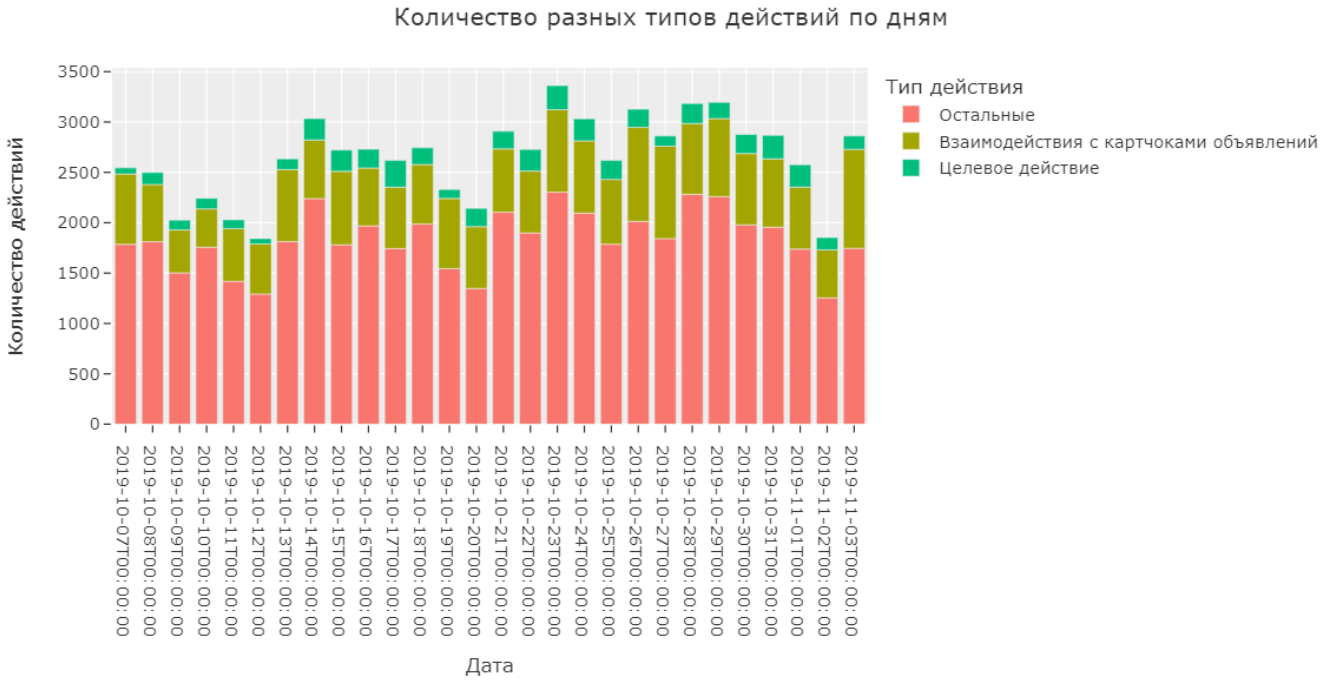
- **Чаще всего на одного пользователя приходится 5-9 действий** (36% пользователей совершают именно столько действий)
- **15.6%** пользователей совершают **меньше 5 действий**
- **75%** пользователей за период совершили **до 19 действий**
- только **1.8%** пользователей успели совершить **более 100 действий**

Количество действий в день по типу событий

In [59]:
```
event_per_day = data.groupby(['date', 'event_name'])['user_id'].count().reset_index()
def rename_event(row):
    if row in ['advert_open', 'favorites_add', 'photos_show', 'tips_click']:
        return 'B'
    elif row in ['show_contacts']:
        return 'C'
    else:
        return 'A'
event_per_day['event_name'] = event_per_day['event_name'].apply(rename_event)
event_per_day = event_per_day.groupby(['date', 'event_name'])['user_id'].sum().reset_index()
event_per_day.index = event_per_day['date']
```

In [60]:
```
fig = px.bar(event_per_day,
             y='user_id',
             color='event_name')
fig.update_layout(title = 'Количество разных типов действий по дням',
                  xaxis_title = 'Дата',
                  yaxis_title = 'Количество действий',
                  legend=dict(title= 'Тип действия'),
                  xaxis = dict({'categoryorder':'total ascending'},
                               tickvals = event_per_day.index, ticktext = event_per_day.index
                  height=500, width=950)

newnames = {'A':'Остальные', 'B': 'Взаимодействия с картчоками объявлений', 'C':'Целевое дейс
fig.for_each_trace(lambda t: t.update(name = newnames[t.name]))
```

```
fig.show()
```



Количество действий по дням распределено неравномерно, график "скачет", не наблюдается ни явного прироста количества, ни спада. Обратим внимание, что **доля действий, связанных с открытием карточек объявлений, мала, доля целевых и вовсе теряется на фоне всех остальных действий** - даже в сумме эти два типа действия не достигают половины от всех событий.

Количество действий за одну сессию

Снова вернемся к таблице с сессиями - посмотрим, сколько действий совершают пользователи за ону сессию

In [61]:
```
print('Всего сессий:', len(sessions))
fig = px.histogram(sessions, x='event_count', nbins=100)
fig.update_layout(title='Количество действий за одну сессию')
fig.update_xaxes(title='Количество действий')
fig.update_yaxes(title='Количество сессий')
fig.show()

print('Сессий, с количеством действий >19:', len(sessions.query('event_count > 19')))
fig = px.box(sessions, x='event_count')
fig.update_layout(title='Количество действий за одну сессию: размах данных')
fig.update_xaxes(title='Количество действий')
fig.update_yaxes(title='')
fig.show()
```

```
Всего сессий: 10368
```


```
Сессий, с количеством действий >19: 743
```

- Большинство сессий, **23%**, включают в себя **2-3 действия**
- **20% сессий заканчиваются на первом и единственном действии**
- Медианное количество действий за сессию: **4**
- Верхняя граница "ящика с усами" упирается в число **19 - большинство сессий ограничиваются этим количеством действий**

- **7.2%** сессий включают в себя больше 19 событий за сессию
- Встречаются аномально "активные" сессии, включающие себя > 100 действий

Выводы: действия и их частота

- топ-3 события по количеству совершенных действий:
  - tips_show - совершено 40055 раз
  - photo_show - 10012 раз
  - search - 6784 раз
- топ-3 по уникальным пользователям иной, более связан с поиском объявлений:
  - tips_show - 2801 уникальный пользователь
  - search - 1666
  - map - 1456

Скорее всего, пользователи сначала интересуются способами поиска объявлений - видят ленту рекоммендаций (tips_show), открывают поиск, открывают карту

- Действия по дням:
  - не наблюдается опредленной тенденции, дейсвия по дням распределены неравномерно
  - **доля целевых действий в каждый из дней крайне мала**, действия в связке "целевое"+"взаимодействие с карточкой" ни в один из дней не достигает даже половины от всех действий
- Действия и пользователи:
  - Медианное число событий на пользователя: 9, чаще всего пользователи совершают **5-9 действий за период**
  - **75%** пользователей в среднем совершают **до 19 событий**
  - встречаются пользователи, совершившие более 100 действий за период, но их доля крайне мала (1.8%)
- Действия за сессию:
  - **20% сессий включают в себя только одно действие**
  - медианное количество действий за сессию: 4, **большинство сессий ограничиваются 19 действиями,** большее количество действий - редкость (7.2%)

✔ __Комментарий от тимлида №1__ Всё верно. Согласен с расчетом

Конверсия в целевое действие

В этом пункте ответим на следующие вопросы:

- Конверсия в целевое действие
- Как может выглядеть путь пользователя в приложении? Возможно ли определить воронку?
- Проанализируем особенности целевых действий

Рассчет конверсии в целевое действие

Отметим, что `'show_contacts'` - не является последним, конечным действием. После просмотра контактов пользователь может, например, позвонить прямо из приложения, в таком случае 'contact_call' будет заключительным этапом при успешном взаимодействии с карточкой объявления.

В рамках данного исследования мы **считаем целевым действием именно просмотр контактов** - на этом этапе пользователь уже достаточно заинтересован в покупке и проявляет желание связаться с автором объявления. Он может, например, посмотреть контакт и позвонить с другого телефона, или сделать скриншот и позвонить позже.

Перейдем к рассчету конверсий.

In [62]:
```
'''Функция, принимает число успехов и общее число,
рассчитывает конверсию, выводит на экран в %
'''
def get_conversion(success, total):
    cr = success/total*100
    print('Конверсия:', "{0:.2f}%".format(cr))
```

Конверсия в уникальных пользователях

In [63]:
```
uniq_payer = data.query('event_name == "show_contacts"')['user_id'].nunique()
uniq_users = data['user_id'].nunique()


print('Всего уникальных пользователей за период:', uniq_users)
print('Из них хотя бы раз посмотрели контакты:', uniq_payer)

get_conversion(uniq_payer, uniq_users)
```

```
Всего уникальных пользователей за период: 4293
Из них хотя бы раз посмотрели контакты: 981
Конверсия: 22.85%
```

На первый взгляд конверсия выглядит достаточно внушительной, но если посмотреть с другой стороны: **77% пользователей (3312 человек) за весь период не соврешили целевое действие**. Снова немного отстранимся и вспомним, что `просмотр контактов не равен факту покупки` - в таком случае, **77% пользователей не заинтересовались ни одним из объявлений** настолько, чтобы связаться с их авторами.

Конверсия по ссесиями: доля сессий, включающих целевое действие

In [64]:
```
good_sessions = data.query('event_name == "show_contacts"')['session_id'].nunique()
total_sessions = len(sessions)


print('Всего сессий:', total_sessions)
print('Сессий, во время которых хотя бы раз пользователь просмотрел контакт:', good_sessions)
get_conversion(good_sessions, total_sessions)
```

```
Всего сессий: 10368
Сессий, во время которых хотя бы раз пользователь просмотрел контакт: 1703
Конверсия: 16.43%
```

В сессиях показатель хуже. **83% сессий не включают в себя целевое действие**, т.е. более 8600 сессий включают в себя любые действия, кроме целевого.

Конверсия по действиям: доля целевых событий событий среди всех остальных

In [65]:
```
cr_event = data.query('event_name == "show_contacts"')['user_id'].count()
total_event = len(data)

print('Всего действий:', total_event)
print('Целевых действий:', cr_event)
get_conversion(cr_event, total_event)
```

```
Всего действий: 74197
Целевых действий: 4529
Конверсия: 6.10%
```

На этот показатель стоит ориентироваться в меньшей степени, он скорее носит ознакомительный характер. Если мы исключаем возможные проблемы с интерфейсом и интуитивностью использования приложения, то пользователь на такого рода платформе может делать сколько угодно действий - крутить ленту и смотреть рекоммендации, настраивать поиск и т.п.. Главное, чтобы пришедший клиент чаще становился "покупателем", а сессии чаще заканчивались "успехом" - целевым действием. **Поэтому важно ориентироваться на две предыдущее конверсии: по пользователям и сессиям.**

Однако рекоммендуется потестировать приложение и проверить "путь пользователя" - хорошо ли настроены рекоммендации, все ли действия интуитивно понятны пользователю, чтобы не оказалось, что клиент совершает слишком много действий, потому что не может найти нужный товар по фильтрам, видит нерелевантные предложения в ленте рекоммендаций и т.п.

О невозможности определения воронки событий и пути пользователя в приложении

В общем мы можем разделить действия пользователя на 4 типа:

1. Пользователь изучает объявления через три источника:

- tips_show - пользователь видит рекоммендованные объявления (лента рекоммендаций)
- search - открывает поиск по объявлениям
- map - открывает карту с объявлениями

2. Ознакамливается с заинтересовавшим объявлением:

- tips_click - кликает на **рекоммендованное объявление**
- advert_open - открытие карточки объявления **из поиска или из карты**

3. Совершает целевое действие

- show_contacts - просмотр контактов

4. Необязательные события:

- photos_show - просмотр фото в объявлении
- favorites_add - добавление объявления в избранное
- contacts_call - заключительное событие, пользователь звонит по указанному в объявлении номеру

Однако у всех этих действий нет определенного порядка.

Почему нельзя определить воронку?

1. **Нельзя сказать, что пользователь действует по определенному паттерну:** например, открывает карту - открывает объявление - смотрит контакты. За одну сессию пользователь может искать объявления через поиск, возвращаться в ленту, искать объявления на карте и т.д. Порядок его действий хаотичен, мы не можем проследить "типичный" путь пользователя

2. **У нас не полные данные** Проанализировав десяток сессий, прихожу к выводу, что в данных нам датасетах указаны не все возможные события, либо не сказано о возможности **смотреть контакты не открывая при этом само объявление.**

Первый пример:

In [66]:
```
data.query('session_id == 242').head(10)
```

Out[66]:

	event_time	event_name	user_id	source	date	weekday	week	session_id
12279	2019-10-12 17:36:57	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12286	2019-10-12 17:38:06	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12294	2019-10-12 17:40:09	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12307	2019-10-12 17:48:44	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12311	2019-10-12 17:52:29	show_contacts	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12313	2019-10-12 17:59:17	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12316	2019-10-12 18:01:27	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12321	2019-10-12 18:06:03	show_contacts	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12325	2019-10-12 18:11:30	tips_show	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242
12338	2019-10-12 18:16:02	tips_click	06bdb96e-2712-47b3-a0af-d19f297abd6c	yandex	2019-10-12	5	41	242

Пользователь 4 раза увидел объявления, не совершал ни 'advert_open', ни 'tip_click', но при этом посмотрел контакты. Потом снова два раза видит объявления и смотрит контакты. **Либо есть возможность посмотреть контакты, не открывая объявление, либо у нас есть пропуски в событиях**. Иными словами, **мы не можем быть хоть сколько уверены, что перед просмотром контактов есть некое обязательное событие.**

Второй пример:

In [67]:
```
data.query('session_id == 10116')
```

Out[67]:

	event_time	event_name	user_id	source	date	weekday	week	session_id
42691	2019-10-23 21:02:36	search	f9e48f0a-b7b3-452b-9018-0d1e1574b9f5	yandex	2019-10-23	2	43	10116
42696	2019-10-23 21:04:35	favorites_add	f9e48f0a-b7b3-452b-9018-0d1e1574b9f5	yandex	2019-10-23	2	43	10116
42700	2019-10-23 21:05:59	favorites_add	f9e48f0a-b7b3-452b-9018-0d1e1574b9f5	yandex	2019-10-23	2	43	10116
42709	2019-10-23 21:09:12	photos_show	f9e48f0a-b7b3-452b-9018-0d1e1574b9f5	yandex	2019-10-23	2	43	10116
42716	2019-10-23 21:10:23	photos_show	f9e48f0a-b7b3-452b-9018-0d1e1574b9f5	yandex	2019-10-23	2	43	10116

Пользователь осуществляет поиск, добавляет объявление в избранное (снова не открывая карточку объявления, видимо, есть возможность "лайкать" на превью), потом просматривает фото. **Либо есть возможность "скролить" фото объявления не открывая его, либо**

пользователь смотрит фото из своего "избранного". Опять же, либо у нас пропущены события, связанные с просмотром объявлений, либо нам просто не даны некоторые типы событий (например, пользователь зашел в избранное)

К сожалению, не удастся построить корректные воронки событий. **Но имея даже такие данные мы можем провести некоторый анализ целевых действий.**

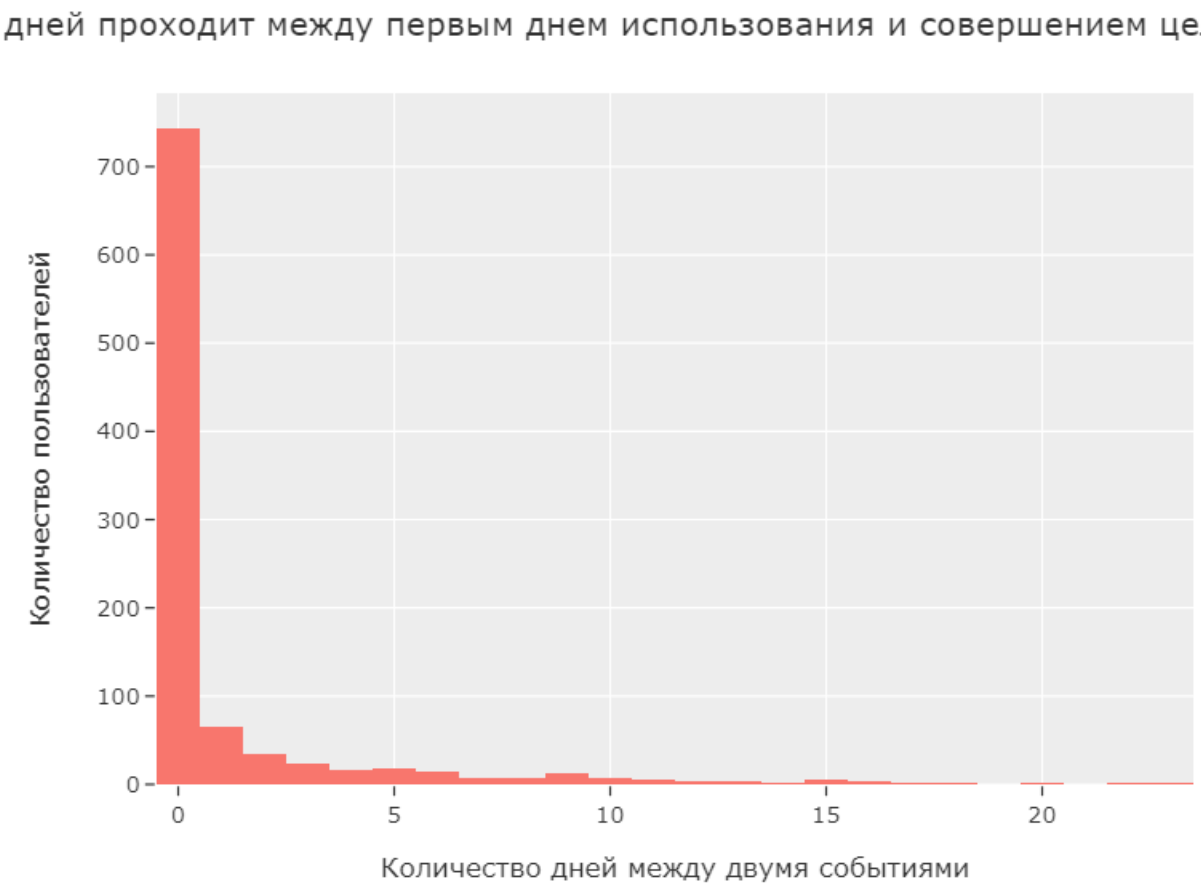
Анализ целевых действий

Через сколько дней после установки приложения пользователь совершает целеое действие?

In [68]:
```
# отбираем пользователей, хоть раз совершивших целевое действие, фильтруем по ним 'data'
good_user = data.query('event_name == "show_contacts"')['user_id'].unique().tolist()
good_data = data.query('user_id in @good_user')
# создаем таблицу с первым днем, таблицу с днем первого целевого события:
good_data_first_day = good_data.groupby('user_id')['date'].first().reset_index()
good_data_first_cr = good_data.query('event_name == "show_contacts"').groupby('user_id')['dat
# объединям таблицы, рассчитываем разницу в днях:
cr_timedelta = good_data_first_cr.merge(good_data_first_day, on='user_id', how='left')
cr_timedelta['timedelta'] = cr_timedelta['date_x'] - cr_timedelta['date_y']
cr_timedelta['timedelta'] = cr_timedelta['timedelta'].dt.days
```

In [69]:
```
print('Всего пользователей, совершавших целевое событие:', len(cr_timedelta))
fig = px.histogram(cr_timedelta, x='timedelta', nbins=25)
fig.update_layout(title='Сколько дней проходит между первым днем использования и совершением
fig.update_xaxes(title='Количество дней между двумя событиями')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```

```
Всего пользователей, совершавших целевое событие: 981
```



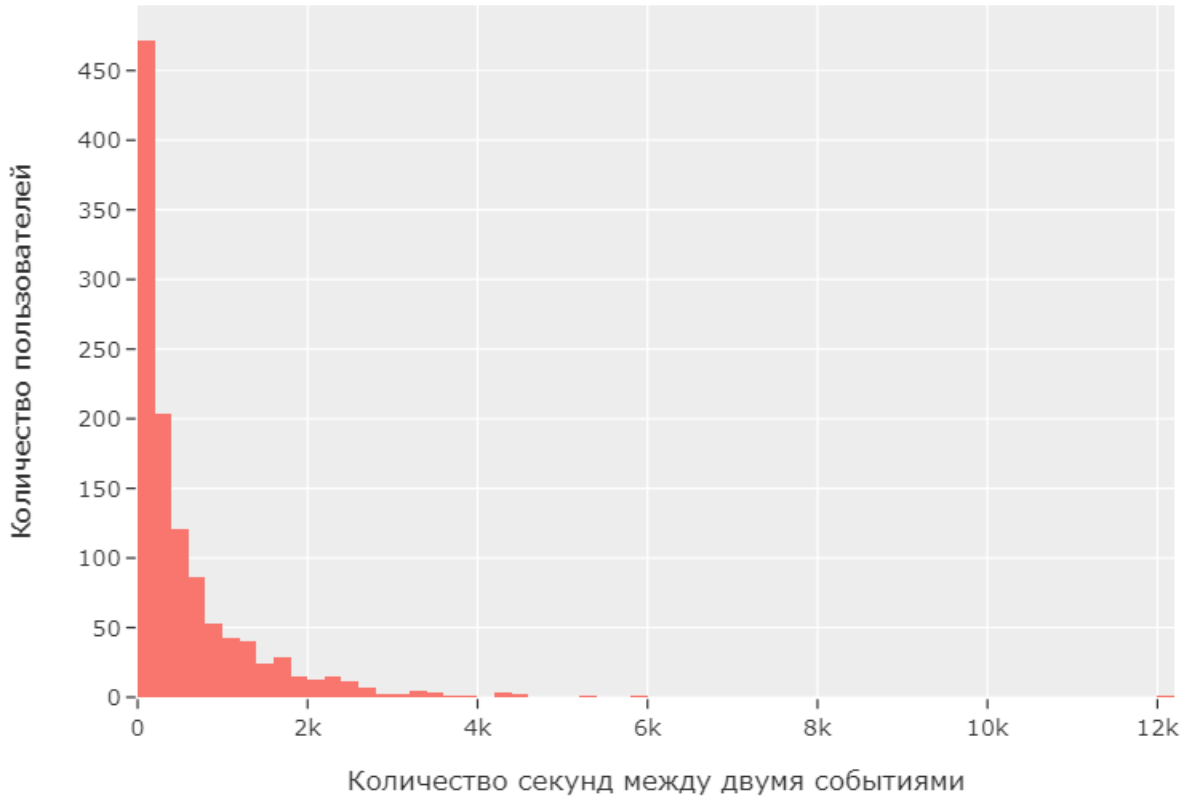
Отличный результат: среди пользователей, совершавших целевое событие, большинство **(75%)**, **совершают его в первый же день использования приложения**.

Сколько времени проходит с начала сессии до целевого действия?

In [70]:
```
# по аналогии с п.3.4.4.1 - только рассчитываем по сессиям, ищем разницу между времени начала
good_data_first_event = good_data.groupby('session_id')['event_time'].first().reset_index()
good_data_first_event_cr = good_data.query('event_name == "show_contacts"')\
            .groupby('session_id')['event_time'].first().reset_index()

first_event_cr = good_data_first_event_cr.merge(good_data_first_event, on='session_id', how='
first_event_cr['timedelta'] = (first_event_cr['event_time_x'] - first_event_cr['event_time_y'
            .dt.total_seconds().astype('int')
# в данных встречаются пользователи, сразу совершивших целевоедействие, очередной недостаток
first_event_cr = first_event_cr.query('timedelta !=0')
```

In [71]:
```
fig = px.histogram(first_event_cr, x='timedelta', nbins=100)
fig.update_layout(title='Сколько секунд проходит между началом сессии и совершением целевого
fig.update_xaxes(title='Количество секунд между двумя событиями')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```

олько секунд проходит между началом сессии и совершением целевого событ



Большинству пользователей нужно не более 3 минут для совершения целевого действия - это тоже хороший результат.

Однако обратим внимание на наличие значительного количетсва сессий, в которых до целевого действия проходит более 30 минут: очередной сигнал проверить удобство поиска и прочих инструментов взаимодействия пользователя с карточками

Сколько "успешных" сессий совершает пользователь?

In [72]:
```
good_session_list = good_data.query('event_name == "show_contacts"')['session_id'].unique().t
good_session_data = data.query('session_id in @good_session_list')
good_session_data_cnt = good_session_data.groupby('user_id')['session_id'].nunique().reset_in
```

In [73]:
```
fig = px.histogram(good_session_data_cnt, x='session_id', nbins=50)
fig.update_layout(title='Количество сессий, включающих целевое событие, на одного пользовател
```

```
fig.update_xaxes(title='Количество успешных сессий')
fig.update_yaxes(title='Количество пользователей')
fig.show()
```

А вот этот показатель сигнализирует нам о не желательном поведении клиентов: **чаще всего пользователи ограничивается одной сессией с целевым событием.** Процент таких пользователь внушительный - **68% из всех, совершавших целеове действие.** Есть вероятность того, что пользователь заходит в приложение, находит нужное объявление, совершает целевое действие и перестает использовать приложение. Стоит обратить внимание на эту метрику, ведь в наших инетресах, чтобы пользователь, совершивший одно целевое действие, продолжал совершать его в дальнейшем, **т.е. оставался с нами**

Количество целевых действий за одну сессию

In [74]:
```
cr_event_by_session = data.query('event_name == "show_contacts"').groupby('session_id')['even
```

In [75]:
```
print('Сессий, содержащих 2-5 целевых действий', len(cr_event_by_session.query('5 >event_name
print('Сессий, содержащих более 5 действий', len(cr_event_by_session.query('event_name >5')))
fig = px.histogram(cr_event_by_session, x='event_name', nbins=100)
fig.update_layout(title='"Успешные" сессии: количество целевых действий')
fig.update_xaxes(title='Количество целевых действий')
fig.update_yaxes(title='Количество сессий')
fig.show()
```

```
Сессий, содержащих 2-5 целевых действий 256
Сессий, содержащих более 5 действий 162
```

Большинство сессий ожидаемо содержат в себе только одно целевое действие. Однако обратим внимание, **что 15% "успешных" сессий включают в себя 2-5 целевых действий, 5.7% - больше пяти целевых действий,** встречаются сессии, включающие в себя 10, 30 и даже 60 целевых действий. Это хороший знак - есть пользователи "затягивающиеся" в приложение, находящие за одну сессию несколько интересных им объявлений. **Метрику стоит повышать доступными способами:** нам оценю важны настолько заинтересованные клиенты.

Выводы: конверсия

Рассчитаны следующие конверсии:

- **по уникальным пользователям: 22.85%**
  - большинство пользователей (73%) не совершают целевое действие
- **по сессиям: 16.43%**
  - большинство сессий заканчиваются "ничем" - пользователи не находят нужное объявление
- **по действиям: 6.10%**
  - конверсия носит ознакомительный характер, большинство действий - нецелевые, что ожидаемо

Рекомендуется:

```
* тестирование пути пользователя, не сталкивается ли пользователь с
техническими проблемами на любом из этапов
* проверить настройки рекомендаций, таргетирования: релевантные ли
рекомендованные объявления видит каждый из пользователей
* корректно ли работает поиск через 'search' и 'map' - внедрены ли нужные
фильтры, сортировки и т.п.
* есть ли какие-то рекомендации по оформлениюо объявлений? Контролируется
ли как-то этот момент? Среди всех действий только 13.5% относятся к
```

просмотру фото - нет ли проблемы в том, что авторы объявлений загружают неинформативное или не соответсвующие действительности фото?

Путь пользователя и воронка событий

Обращаю внимание, **что либо нам даны не все события, либо имеются пропуски в действиях**. В любом случае, поведение пользователя хаотично, определить воронку невозможно, перед целевым действием нет обязательного события - пользователь может просмотреть контакты или фото, не открывая при этом карточку.

Анализ целевых действий (среди пользователей, хоть раз совершивших "просмотр контактов":

- **Положительные наблюдения:**
  - чаще всего пользователь **совершает целевое действие в первый день в приложении**
  - для совершения целевого действия обычно **нужно не более 3-минут с начала сессии**
  - встречаются пользователи, совершающие 2 и более целевых действий за одну сессию

Первые две метрики необходимо поддерживать на том же уровне, третью повыашть всеми способами.

- **Отрицательные наблюдения:**
  - чаще всего **после первой успешной сессии пользователь не совершает вторую такую же.**

Это очень важная метрика, необходимо изучить ее более глубоко: не получил ли пользователь какой-то неудачный опыт взаимодействия с платформой после совершения целевого действия? Почему чаще всего он не возвращается?

Сегментация пользователей

Определение признаков для сегментации

По какому принципу лучше сегментировать пользователей?

Имеющиеся у нас данные позвояют сегментировать пользователей по следующим признакам:

- **По источнику привлечения.**

В целом, мы действительно легко разделим пользователей в зависимости от исчтоника, но в рамках текущего исследования от такой сегментации мы не получим много полезных инсайтов. Сегментировать по источникам логичнее при проведении маркетингового анализа, в рамках которого мы бы рассчитали ROI, CAC и прочие характеристики. По имеющимся у нас данным мы можем рассчитать конверсии и выявить какие-то поведеенческие особенности клиентов из каждой группы - но так ли это полезно?

- **На основании времени: по дате первой сесии**

Куда более бесполезный вариант: невозможно определить ЦА. Нелогично адаптировать приложение под пользователей, которые приходят по понедельникам, еще нелогичнее адаптировать под пользователей, пришедщих к нам в какой-то определенный период

- **На основании удержания**

Вариант вполне рабочий, можно сегментировать на основании "продолжительности жизни" в приложении. Но вспоминаем недостаток наших данных - слишком маленький временной диапазон. Пользователи, пришедшие к нам в последнюю неделю, очевидно покажут результаты хуже - они еще не успели достаточно "прожить", успели совершить мало сессий и т.д.

- **На основании поведенческих особенностей**

Единственный логичный вариант. Даже те пользователи, что пришли к нам совсем недавно успели соврешить какие-либо действия - на основании этих действий мы и можем сегментировать пользователей

Остановимся на последном варианте. Можем разделить пользователей по количеству сессий или по количеству совершенных действий, но опять столкнемся с "проблемой новых пользователей" - очевидно, они совершили меньше сессий и действий, чем остальные. Но мы точно знаем, что **каждый пользователь совершил хотя бы одну сессию**, в таком случае мы можем рассмотреть **количество действий за одну сессию.**

На основании всего выше сказанного, **разделим пользователей по среднему количеству действий за одну сессию**

Сегментация пользователей по среднему количеству действий за одну сессию

Обратимся к раннее созданной таблице 'sessions' - посчитаем для каждого пользователя среднее количество действий за сессию, сохраним в отдельную таблицу 'event_per_session' и посмотрим как распределены данные.

In [76]:
```
event_per_session = sessions.groupby('user_id')['event_count'].mean().reset_index()
event_per_session['event_count'] = round(event_per_session['event_count'],2)
```

In [77]:
```
print(event_per_session['event_count'].describe())
fig = px.histogram(event_per_session, x='event_count', nbins=60)
fig.update_layout(title='Среднее количество действий за одну сессию: распределение по пользов
                   yaxis_title='Количество пользователей',
                   xaxis_title='Количество действий',
                  height=500, width=900)
fig.show()
```

```
count    4293.000000
mean        8.291216
std         8.393237
min         1.000000
25%         3.670000
50%         6.000000
75%        10.000000
max       104.000000
Name: event_count, dtype: float64
```

Разделим пользователей на 4 сегмента на основании среднего количества действий за сессию

- **A** - пользователи с "активными" сессиями: > **10 действий за сессию**
- **B** - пользователи с менее активными сессиями: **от 6 до 10 действий включительно**
- **C** - пользователи с количество сессий меньше медианного, но больше Q1: **от 4 включительно до 6 действий**
- **D** - пользователи с малоактивными сессиями: **< 4 действий за сессию**

In [78]:
```
''' Функция, принимает строку, возвращает название сегмента в зависимости
от значения строки
'''
def make_segment_ssss(row):
    if row > 10:
        return 'A'
    if 10 >= row >=6:
        return 'B'
    if 6 > row >= 4:
        return 'C'
    else:
        return 'D'

event_per_session['segment'] = event_per_session['event_count'].apply(make_segment_ssss)
print('Число уникальных пользователей в каждом сегменте:')
print(event_per_session['segment'].value_counts())
print()
print('Общее число уникальных пользователей, разбитых на сегменты:', event_per_session['segme
```

```
Число уникальных пользователей в каждом сегменте:
B    1210
D    1109
A     993
C     981
Name: segment, dtype: int64

Общее число уникальных пользователей, разбитых на сегменты: 4293
```

В каждой группе достаточное количество уникальных пользователей, общее количество совпадает, никого не потеряли. Для дальнешей работы создадим для каждого сегменты отдельные data и sessions

In [79]:
```
'''Функция, принимает название сегмента,
создает список с уникальными user_id заданного сегмента,
фильтрует 'data' и 'sessions' по полученному списку с user_id
возвращает две таблицы: отфильтрованные 'data' и 'session'
'''

def get_df_for_group(segment):
    group_users = event_per_session.query(f'segment == "{segment}"')['user_id'].tolist()
    group_data = data.query('user_id in @group_users')
    group_sessions = sessions.query('user_id in @group_users')
    return group_data, group_sessions

a_data, a_sessions = get_df_for_group("A")
b_data, b_sessions = get_df_for_group("B")
c_data, c_sessions = get_df_for_group("C")
d_data, d_sessions = get_df_for_group("D")
```

Необходимые таблицы получены, приступим к анализу полученных сегментов

Анализ сегментов. Определение ЦА

Конверсия в уникальных пользователях

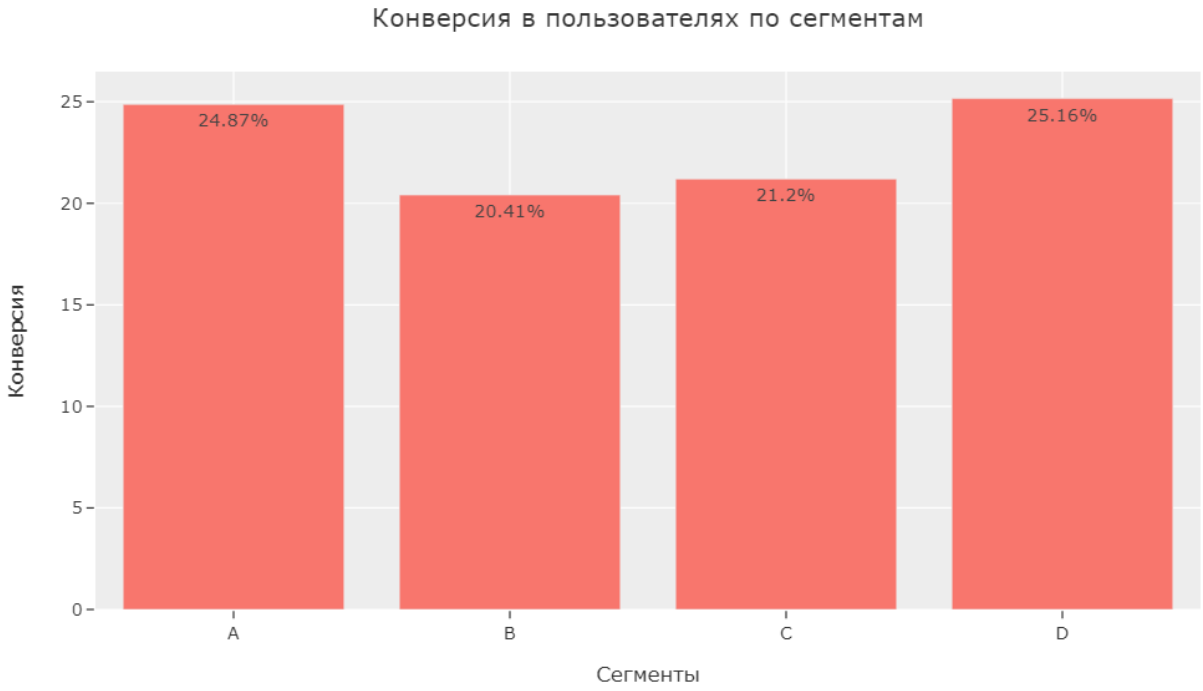
Посмотрим, какой из сегментов показывает лучшую конверсию в уникальных пользователях

In [80]:
```
''' Функция, принимает таблицу,
считает кол-во уник. пользователей, совершивших целевое действие и общее кол-во уник.пользова
расчитывает конверсию
считает кол-во сессий, включающих целевое действие, и общее кол-во сессий
рассчитывает конверсию
добавляет обе конверсии в список, возвращает список
'''
def get_cr_for_group(df):
    cr_group = []
    user_success = df.query('event_name == "show_contacts"')['user_id'].nunique()
    user_total = df['user_id'].nunique()
    user_cr = user_success/user_total
    session_success = df.query('event_name == "show_contacts"')['session_id'].nunique()
    session_total = df['session_id'].nunique()
    session_cr = session_success/session_total
    cr_group.extend([user_cr, session_cr])
    return cr_group
a_cr = get_cr_for_group(a_data)
b_cr = get_cr_for_group(b_data)
c_cr = get_cr_for_group(c_data)
d_cr = get_cr_for_group(d_data)
```

In [81]:
```
# Создаем таблицу с конверсиями каждого сегмента
cr_df_percent = pd.DataFrame([a_cr, b_cr, c_cr, d_cr], columns=['user_cr', 'sessions_cr'], in
cr_df_percent = round(cr_df_percent*100,2)
```

In [82]:
```
fig = px.bar(y = cr_df_percent['user_cr'], x= cr_df_percent.index,
                     text =cr_df_percent['user_cr'])
fig.update_traces(text = ["{}%".format(val) for val in cr_df_percent['user_cr'] ])
fig.update_layout(title='Конверсия в пользователях по сегментам',
                    yaxis_title='Конверсия',
                    xaxis_title='Сегменты',
                   height=500, width=900)
fig.show()
```

```
C:\Users\Hp\anaconda4\lib\site-packages\numpy\core\numeric.py:2446: FutureWarning:

elementwise comparison failed; returning scalar instead, but in the future will perform eleme
ntwise comparison
```



Лучшую конверсию показывают группы D и A: чаще всего совершают целевое действие пользователи совершающие много действий за сессию, и, удивительно, наоборот - пользователи с меньшим числом действий.

`Высокая конверсия первой группы интуитивна понятна:` пользователи совершают много действий за одну сессию (соответственно, больше времени проводят в приложении) - видят больше объявлений (больше выбора, больше шансов увидеть интересное предложение) - чаще совершают целевое действие.

Высокая конверсия группы D наталкивает на мысль, что пользователь действует по схеме `пришел-увидел-купил-ушел` , т.е. пользователь приходит с конкретным запросом - за короткое количество действий находит нужный товар - совершает целевое действие. Есть вероятность того, что если конкретный запрос удовлетворен, то пользователю больше нет необходимости использовать приложеним.

Проверим, так ли это. **Посмотрим на удержание каждой из групп**

Retention Rate

Работаем с ранее созданной таблицей 'long_lived_result_raw' - в ней собраны пользователи, успевшие "прожить" 3 недели, горизонт анализа - 14 дней

In [83]:
```
'''Функция, принимает таблицу,
собирает уникальные user_id из нее в список,
фильтрует по этому списку  таблицу 'long_lived_result_raw',
создает таблицу с удержанием по аналогии с пунктом 3.1
возвращает таблицу с удержанием
'''

def get_retention_for_group(df):
    group_users = df['user_id'].unique().tolist()
    group_data = long_lived_result_raw.query('user_id in @group_users')
    retention_frame_raw = group_data.pivot_table(columns='lifetime', values='user_id', aggfun
```

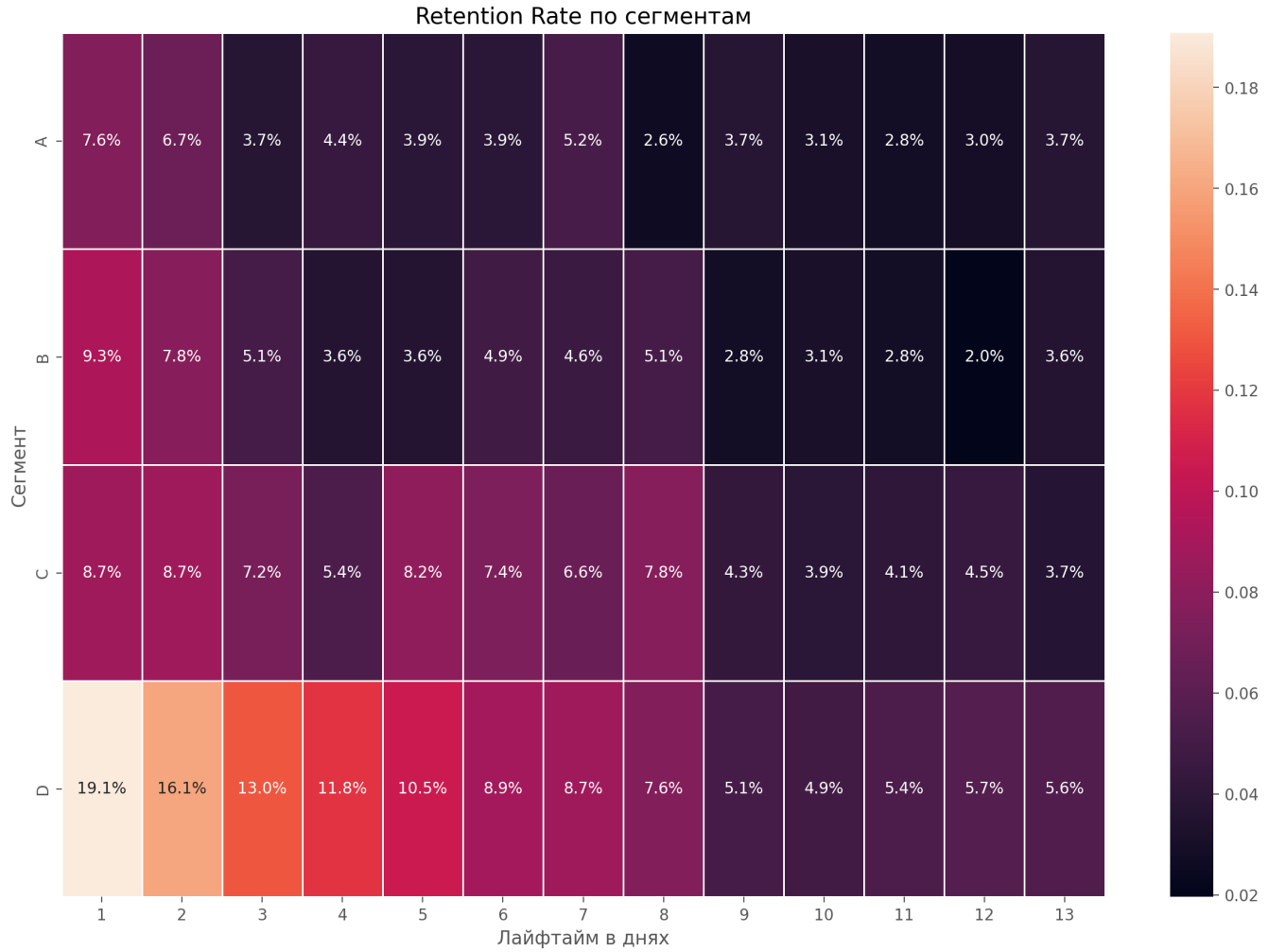
```
    retention_frame= retention_frame_raw.div(
    retention_frame_raw[0], axis=0
).drop(columns=[0])
    return retention_frame

a_retention = get_retention_for_group(a_data)
b_retention = get_retention_for_group(b_data)
c_retention = get_retention_for_group(c_data)
d_retention = get_retention_for_group(d_data)
# собираем все группы в одну таблицу:
all_group_retention = pd.concat([a_retention, b_retention, c_retention, d_retention])
all_group_retention.index = ['A','B','C','D']
all_group_retention
```

Out[83]:

lifetime	1	2	3	4	5	6	7	8	9	10	
A	0.075926	0.066667	0.037037	0.044444	0.038889	0.038889	0.051852	0.025926	0.037037	0.031481	0.0
B	0.093137	0.078431	0.050654	0.035948	0.035948	0.049020	0.045752	0.050654	0.027778	0.031046	0.0
C	0.087379	0.087379	0.071845	0.054369	0.081553	0.073786	0.066019	0.077670	0.042718	0.038835	0.0
D	0.190779	0.160572	0.130366	0.117647	0.104928	0.089030	0.087440	0.076312	0.050874	0.049285	0.0

In [84]:

```
plt.figure(figsize=(15,10))
plt.title('Retention Rate по сегментам')
sns.heatmap(all_group_retention, annot=True, fmt='.1%', linewidths=1)
plt.xlabel('Лайфтайм в днях')
plt.ylabel('Сегмент');
```



Наша гипотеза не подтвердилась: группа D показывает результат по удержанию заметно лучше, чем другие группы.

Выходит, мы не можем сказать, что пользователи из **группы D** прекращают использовать приложение после нескольких сессий, наоборот - **на второй день возвращается 19.1% пользователей,** такой результат более чем в 2 раза превышает результат любой другой группы. **Удержание на 14-й лайфтайм у группы D тоже выше, чем у остальных групп**

Группа A наоборот показывает самые низкие коэффициенты удержания. Хотя пользователи из групп и проводят достаточно долгие и насыщенные действиями сессии, возвращаются в приложение они куда менее охотно.

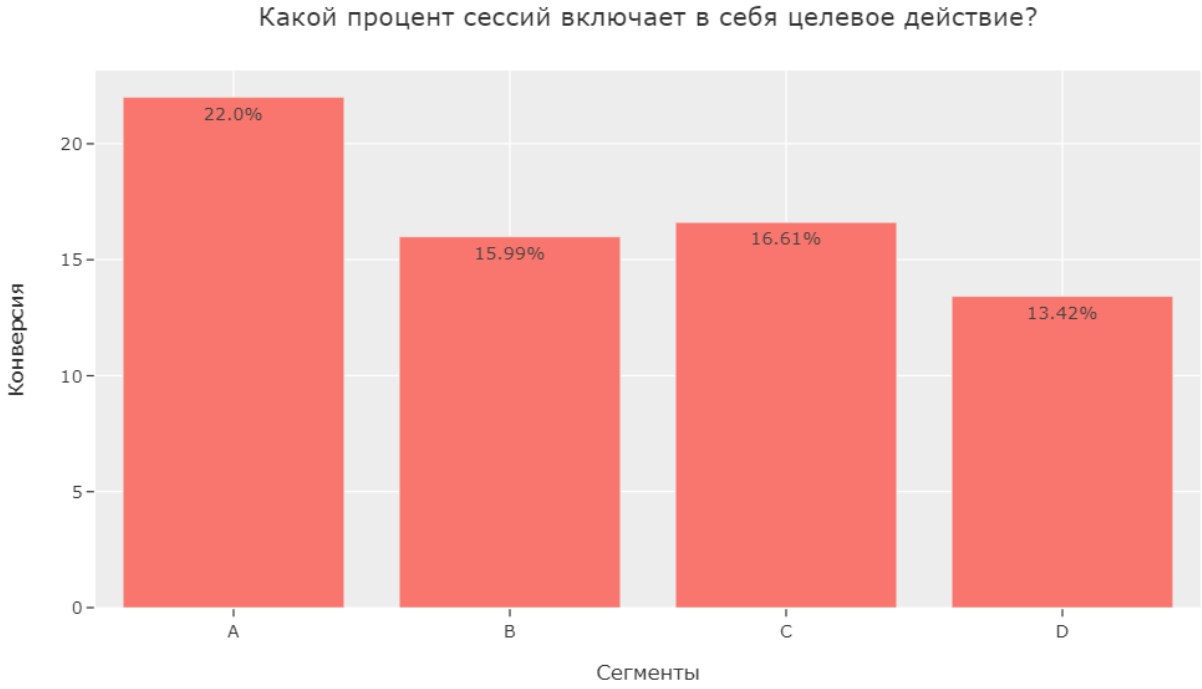
Группы **B и C** не демонстрирует каких-то примечательных результатов: **на второй день отсекаются ~90% новых пользователей,** к концу второй недели "доживают" только 3.1%.

Конверсия в сессиях

Посмотрим насколько "успешны" сами сессии у каждой из групп - какой процент сессий включает в себя целевое действие?

In [85]:
```
fig = px.bar(y = cr_df_percent['sessions_cr'], x= cr_df_percent.index,
                      text =cr_df_percent['sessions_cr'])
fig.update_traces(text = ["{}%".format(val) for val in cr_df_percent['sessions_cr'] ])
fig.update_layout(title='Какой процент сессий включает в себя целевое действие?',
                  yaxis_title='Конверсия',
                  xaxis_title='Сегменты',
                 height=500, width=900)
fig.show()
```

```
C:\Users\Hp\anaconda4\lib\site-packages\numpy\core\numeric.py:2446: FutureWarning:

elementwise comparison failed; returning scalar instead, but in the future will perform eleme
ntwise comparison
```



А вот **сессии у сегмента A намного успешнее, чем у других сегментов - 22% сессий включают в себя целевое действие**.

Другие группы заметно отстают от A. **Ниже всех конверсия у группы D - 86.6% сессий не включают в себя просмотр контактов.**

"Сверхуспешные" сессии

Ранее мы выяснили, что пользователь за одну сессию может несколько раз совершить целевое действие. Назовем такие сессии "сверхуспешными" и **посмотрим, сколько сессий с двумя и более целевыми действиями совершили пользователи из разных групп.**

In [86]:
```
''' Функция, принимает таблицу,
оставляет только строки с целевым действием,
группирует по сессиям, подсчитывает количество событий в каждой сессии,
оставляет только сессии, в которых >1 события, подсчитывает их количество.
Сохраняет  топ-10 сессий по кол-ву целевых действий в список
Возвращает кол-во сессий, список топ-10
'''

def count_success_per_session(df):
    total_sessions = df['session_id'].nunique()
    success_data = df.query('event_name == "show_contacts"')
    success_sessions = success_data['session_id'].nunique()

    success_cnt_per_session = success_data.groupby('session_id')['event_name'].count()\
        .sort_values(ascending=False).reset_index()
    success_cnt_per_session = success_cnt_per_session.query('event_name >1')
    super_success_sessions = len(success_cnt_per_session)
    head_success = success_cnt_per_session['event_name'].head(10).tolist()

    return head_success, super_success_sessions, success_sessions, total_sessions

a_head, a_super_sessions, a_success_sessions, a_total_sessions = count_success_per_session(a_
b_head, b_super_sessions, b_success_sessions, b_total_sessions = count_success_per_session(b_
c_head, c_super_sessions, c_success_sessions, c_total_sessions = count_success_per_session(c_
d_head, d_super_sessions, d_success_sessions, d_total_sessions = count_success_per_session(d_
```

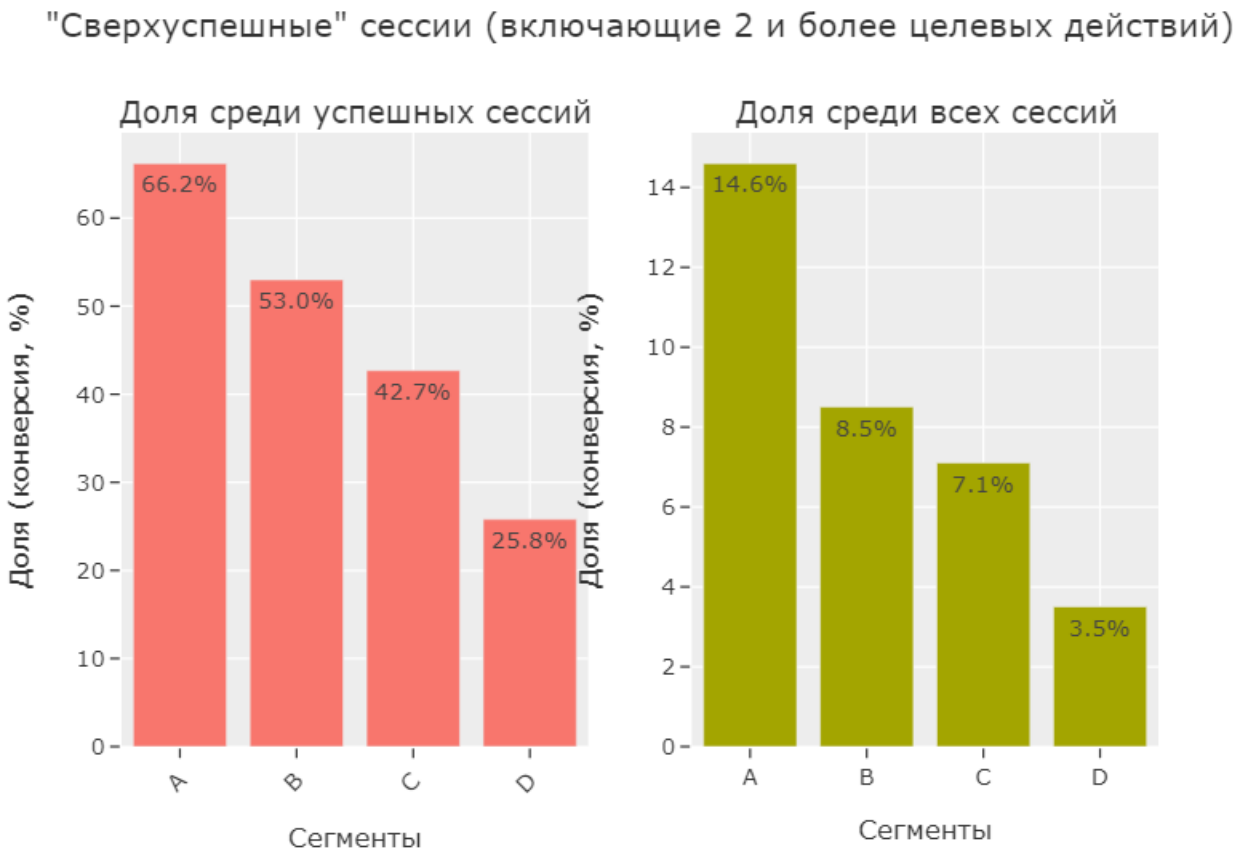
In [87]:
```
# собираем результаты функций ф отдельные таблицы
success_session_cr = pd.DataFrame({'segment': ['A', 'B', 'C', 'D'],
                        'super': [a_super_sessions, b_super_sessions, c_super_sessions, d_su
                       'success':[a_success_sessions, b_success_sessions, c_success_sessions
                       'total': [a_total_sessions, b_total_sessions, c_total_sessions, d_tot

success_session_cr['super_succ_cr'] = round(success_session_cr['super']/success_session_cr['s
success_session_cr['super_total_cr'] = round(success_session_cr['super']/success_session_cr['
success_session_cr
```

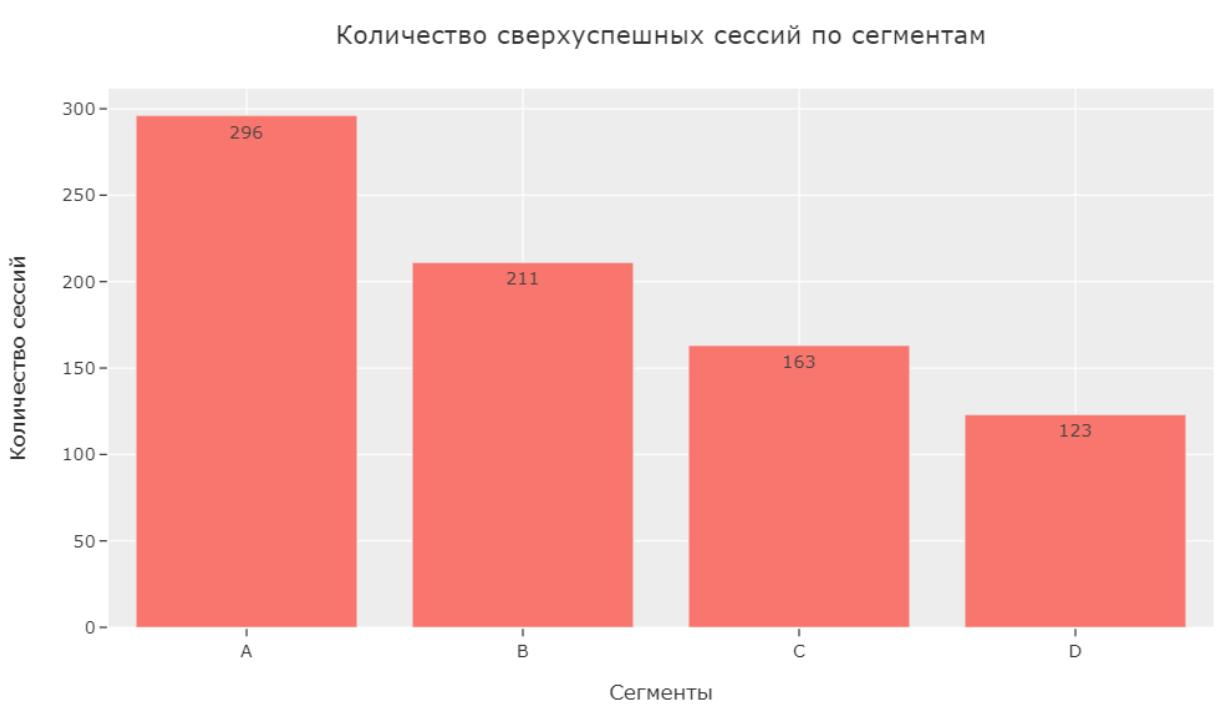
Out[87]:

	segment	super	success	total	super_succ_cr	super_total_cr
0	A	296	447	2032	66.2	14.6
1	B	211	398	2489	53.0	8.5
2	C	163	382	2300	42.7	7.1
3	D	123	476	3547	25.8	3.5

In [88]:
```
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'Bar'}, {'type':'Bar'}]],
                    subplot_titles=['Доля среди успешных сессий', 'Доля среди всех сессий'])

fig.add_trace(go.Bar(x=success_session_cr['segment'], y=success_session_cr['super_succ_cr'],
                     text=["{}%".format(val) for val in success_session_cr['super_succ_cr'] ]
                    ),
              1, 1)

fig.add_trace(go.Bar(x=success_session_cr['segment'], y=success_session_cr['super_total_cr'],
                      text=["{}%".format(val) for val in success_session_cr['super_total_cr']
              1, 2)

fig.update_layout(xaxis_tickangle=-45, title='"Сверхуспешные" сессии (включающие 2 и более це
                  title_x = 0.5,showlegend=False, )
fig.update_yaxes(title='Доля (конверсия, %)', col=1, row=1)
```

```
fig.update_yaxes(title='Доля (конверсия, %)', col=2, row=1)
fig.update_xaxes(title='Сегменты', col=1, row=1)
fig.update_xaxes(title='Сегменты', col=2, row=1)
fig.show()

fig = px.bar(success_session_cr,
             x='segment',
             y='super',
            text='super')

fig.update_layout(title='Количество сверхуспешных сессий по сегментам',
                  xaxis_title='Сегменты',
                  yaxis_title='Количество сессий',
                 height=500, width=900)

fig.show()
```





Здесь наблюдаем **явное преимущество сегмета 'A'**: среди сессий, содержащих в себе целевое действие, **66% включают в себя два и более целевых действий**. Доля таких сессий среди всех сессий тоже досаточно внушительная: **14.6%**. Ни одна из групп не может похвастаться даже близким к этому значению.

Также посмотрим на самые насыщенные на целевое действие сессии:

In [89]:
```
group_head =  pd.DataFrame({'A':a_head, 'B':b_head, 'C':c_head, 'D':d_head})
group_head
```

Out[89]:

	A	B	C	D
0	67	37	13	7
1	62	28	11	7
2	53	19	9	6
3	49	16	9	6
4	33	14	8	5
5	29	13	8	5
6	29	11	7	5
7	28	10	7	5
8	27	10	6	5
9	23	10	5	4

У группы A есть сессии, включающие себя 50+ целевых действий, на 10 месте среди сверхуспешных сессий - 23 просмотра контактов. Снова ни одна группа не демострирует такой результат. В целом, логично, что чем больше действий совершено, тем, вероятно, среди них будет больше и целевых действий, но такой результат все равно выглядит значительным.

Выводы. Определение ЦА

Группы B и C не демонстрируют каких-то примечательных результатов, а вот **группы A и D мы можем считать лидерами**. Однако не рекомендуется рассматривать группы как ЦА по модели

"ориентируемся на две лидирующие группы, а остальные пытаемся "дотянуть" до показателей лидеров".

Группы A и D - разные. Одна группа показывает лучший результат по доле успешных сессий, вторая по удержанию, обе имеют высокую конверсию в уникальных пользователях. Примем типичное поведение двух этих групп как **желаемое поведение пользователя** и будем ориетироваться сразу на два направения:

- **Поведенческий паттерн группы A:**

Активные, долгие сессии: пользователи любят долго "сидеть" в приложении. Сессии насыщены событиями - клиенты используют весь функционал приложения. Рекомендуется:

- **проработка многовариативности пользовательского пути:** пользователь совершает много действий разного характера, всегда дожна быть возможность "перескакивать" с одного блока на другой - с карты в поиск, из поиска в избранное и т.д, желательно, чтобы это совершалось одним-двумя кликами
- **проверить (усовершенствовать при необходимости) корректность рекоммендаций:** пользователя легко заинтересовать объявлением, в том числе и из рекоммендаций
- **пользователь любит проводить время в приложении, но редко возвращается:** для этой группы могут сработать пуши-напоминалки, пуши-подборки по типу "объявления, похожие на те, что вы искали", напоминания об отложенном. Повторюсь, пользователь любит приложение - главное напоминать ему возвращаться) Но не переусердствовать.

- **Поведенческий паттерн группы D:**

Короткие, малонасыщенные действиями сессии. Пользователи ценят возможность быстро найти нужное объявление. Рекомендуется:

- **максимально понятный интерфейс:** пользователь ценит свое время и хочет ьыстро найти нужное
- **корректность работы поиска и всех его фильтров и сортировок:** есть вероятность, что пользователь свернет приложение, не найдя нужного сразу
- **информативность карточек на превью:** пользователь сразу должен видеть цену, территориальные расположение, описание, рейтинг продавца - не открывая карточку
- **возможность короткого пути до целевого действия:** пользователь не должен долго искать контакты, нашел объявление - открыл - позвонил
- **пуши скорее спугнут:** пользователи и так достаточно активно возвращаются в приложение, главное - поддерживать удобство его использования для них

Проработав рекомендации для обеих групп мы одновременно покроем потребности двух самых "конверсионных" категорий пользователей: тех, что любят "посидеть" в приложении, и тех, что любят быстро закрыть свою потребность

Проверка гипотез

Гипотеза о разнице конверсий пользователей, пришедших из Yandex и Google

Некоторые пользователи установили приложение по ссылке из yandex, другие — из google.

Проведем z-тест и выясним, есть ли разница между конверсиями в целевое действие у этих двух групп. Установим уровень статистической значимости в 0.01.

Сформулируем гипотезы:

- **Нулевая гипотеза (H0):** Между конверсиями двух групп **нет** статистически значимой разницы (группы равны)
- **Альтернативная гипотеза (H1):** Между конверсиями двух груп **есть** статистически значимая разница (группы не равны)

In [90]:
```
# фильтруем 'data' по источнику привлечения:
google_users_data = data.query('source == "google"')
yandex_users_data = data.query('source == "yandex"')

# В 'users_total' сохраняем всех уникальных пользователей по каждому источнику,
# в 'users_success' только тех, кто совершил целевое действие:
google_users_total = google_users_data['user_id'].nunique()
yandex_users_total = yandex_users_data['user_id'].nunique()

google_users_success = google_users_data.query('event_name == "show_contacts"')['user_id'].nu
yandex_users_success = yandex_users_data.query('event_name == "show_contacts"')['user_id'].nu
```

In [91]:
```
'''Функция, принимает количество успехов и количество попыток двух групп, уровень стат.значим
рассчитывает пропорции успехов в обеих групах, пропорцию успехов в комбинированном датасете,
проводит z-тест,
возвращает средние конверсии для каждой группы, p_value, результат z-теста
'''

def get_z_test(success_1, success_2, total_1, total_2, alpha):
        # пропорция успехов в первой группе:
    p1 = success_1 / total_1
        # пропорция успехов во второй группе:
    p2 = success_2 / total_2
        # пропорция успехов в комбинированном датасете:
    p_combined = ((success_1 + success_2)/
                      (total_1 + total_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/total_1 + 1/total_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('Конверсия первой группы:', "{0:.2f}%".format(success_1/total_1*100))
    print('Конверсия первой группы:', "{0:.2f}%".format(success_2/total_2*100))
    print('p-value: {}'.format(p_value))
    if (p_value < alpha):
        print('Отвергаем нулевую гипотезу, между группами есть статистически значимые различи
    else:
        print('Не отвергаем нулевую гипотезу, между группами нет статистически значимых разли
    print()

get_z_test(google_users_success, yandex_users_success, google_users_total, yandex_users_total
```

```
Конверсия первой группы: 24.36%
Конверсия первой группы: 24.72%
p-value: 0.8244316027993777
Не отвергаем нулевую гипотезу, между группами нет статистически значимых различий
```

Вывод

Не удалось опровергнуть нудевую гипотезу, нельзя зафиксировать победу ни одной из групп.
Различие конверсий пользователей пришедших из yandex и google статистически не

значимо

Гипотеза о разнице конверсий пользователей совершающих 'favorites_add' и не совершающих это действие

В приложении доступен функционал "добавить в избранное", т.е. пользователь может вернуться к заинтересовавшему объявлению. Предположим, что пользователи, добавляющие объявление в избранное, чаще совершают целевое действие, чем те, кто ни разу не совершал 'favorites_add'.

Снова проведем z-тест между двумя группами с уровнем статистической значимости 0.01

Сформулируем гипотезы:

- **Нулевая гипотеза (H0):** Между конверсиями двух групп **нет** статистически значимой разницы (группы равны)
- **Альтернативная гипотеза (H1):** Между конверсиями двух груп **есть** статистически значимая разница (группы не равны)

In [92]:
```
# собираем в список 'fav_users_list' пользователей, совершавших 'favorites_add':
fav_users_data_raw = data.query('event_name == "favorites_add"')
fav_users_list = fav_users_data_raw['user_id'].unique().tolist()

fav_users_data = data.query('user_id in @fav_users_list')
not_fav_users_data = data.query('user_id not in @fav_users_list')

# В 'users_total' сохраняем всех уникальных пользователей по каждому источнику,
# в 'users_success' только тех, кто совершил целевое действие:
fav_users_total = fav_users_data['user_id'].nunique()
not_fav_users_total = not_fav_users_data['user_id'].nunique()

fav_users_success = fav_users_data.query('event_name == "show_contacts"')['user_id'].nunique(
not_fav_users_success= not_fav_users_data.query('event_name == "show_contacts"')['user_id'].n
```

In [93]:
```
get_z_test(fav_users_success, not_fav_users_success, fav_users_total, not_fav_users_total, 0.
```

```
Конверсия первой группы: 38.75%
Конверсия первой группы: 21.44%
p-value: 1.3455903058456897e-13
Отвергаем нулевую гипотезу, между группами есть статистически значимые различия
```

Вывод

Фиксируем победу первой группы: **пользователи, добавляющие объявления в избранное имеют конверсию выше, чем пользователи, не совершающие это действие.**

Возможность добавлять карточки в избранное - важный элемент использования приложения. Клиент может вернуться к объявлению позже, добавить несколько похожих объявений для сравнения, так или иначе, вероятность совершения целевого действия будет выше. Команде приложения стоит учитывать значимость этой функции: **добавление в избранное и возвращение к отложенным объявлениям всегда должно быть интуитивно понятным для пользователя**

Выводы

Обработаны и проанализированы оба датасета, объединенный в один: 74197 наблюдений по 4293 уникальным пользователям. Предложенный для исследования период: 28 дней, 07.10.2019-03.11.2019

Комментарии по предоставленным данным

- данные целостные, таблицы не содержат пропусков и некорретных значений
- **нет информации о длительности сессий:** ни о дате страта сессии, ни о дате окончания, события самостоятельно разбиты на сессии на основании 30-минутного таймаута. Возможно некоторое искажение данных
- **возможны пропуски в событиях, не описан конкретный порядок действий пользователя:** не выявляется обязательное событие перед совершением целевого, нет информации о различиях в событиях search_1-7, некоторые события (целевое, добавление в избранное и др.) случаются без открытия карточки объявления: информация в описании об этом не предоставлена. Встречаются случаи совершения целевого события без любого другого предыдущего.

Описанные выше особенности данных могут несколько влиять на результаты исследования.

Исследовательский анализ

Проведен исследовательский анализ данных, проанализированные интересующие заказчика метрики.

1. Retention Rate

Предоставлен короткий временной диапазон для изучения. Рекомендуется провести анализ на бОльшем временном промежутке и оценить метрику по месяцам и/или неделям, т.к. вряд ли от пользователя ожидается ежедневное использование приложения

- **по неделям:**
  - наблюдается отток новых пользователей: 1130 - 1116 - 1094 -903 по неделям
  - когорты удерживаются неравномерно, **коээфицианты удерживания на вторую неделю:** 24.1%, 24.2%, 21.8% (когорты 41, 42, 43 недель соответственно)
  - по данным когорты 41 недели: **на 4-ю недели жизни удерживается только 10.5%**

- **по дням** (*проанализрованы пользователи, "прожившие" 3 недели, горизонт анализа: 14 дней*):
  - **удержание не спадает равномерно по когортам:** в одной когорте на второй день может вернутся 16% новых клиентов, в другой только 7%
  - **удержание внури одной когорты неравномерно:** на 9-й день может вернутся 1%, а на 12-й 5%
  - на 4-й лайфтайм ни у одной из когорт **не наблюдается удержание >10%**

2. Сессии и время, проведенное в приложении

События разбиты на 10368 сессий на основании таймаута. У каждой сессии только один пользователь.

- **Длительность сессии:**

  - Чаще всего одна сессия длится до 1 минуты
  - Медианная длительность сессии: 9 минут
  - 37% сессий длятся до 5 минут, 51% сессий длятся до 9 минут, 65% сессий длятся до 15 минут
  - встречаются сверхдолгие сессии, их доля незначительна (например, сессии более 90 минут: 1.1% от всех сессий)

- **Количество и длительность сессий по дням:**
  - **с 14.10 пользователи ведут себя активнее:** количество сессий не менее 350 в день.
  - внутри одной недели **количество сессий в выходные ниже,** чем в будние дни
  - усредненная длительность сессий по дням колеблется в диапазоне 15-21 минута, показатель по дням неравномерен
  - длительность в выходные не уступает, а иногда и превосходит длиельность в будние дни
- **Топ-часы по посещаемости:**
  - **Будни**
    - заметная активность начинается в 9:00
    - самый популярный период: 13:00-15:00, в целом популярен период 12:00-16:00
    - вечером основная активность происходит в период 20:00-22:00
  - **Выходные**
    - заметная активность начинается в 10:00
    - самые популярный период: вечер, 19:00-20:00, 21:00-22:00
    - дневная активность выражена слабее вечерней, днем самый популярный час 15:00-16:00
- **Количество сессий на одного пользователя**
  - чаще всего пользователь совершает **одну сессию** (53.6% от всех пользователей)
  - 20% пользователей совершило 2 сессии, 10.7% - 3 сессии, большее число сессий - редкость
  - встречаются крайне активные пользователи с числом сессий >20

3. События. Анализ частоты действий

- **Топ-3 событий**
  - **по количеству совершенных действий:**
    - tips_show - совершено 40055 раз
    - photo_show - 10012 раз
    - search - 6784 раз
  - **по уникальным пользователям** (*более связан с поиском объявлений*)
    - tips_show - 2801 уникальный пользователь
    - search - 1666
    - map - 1456

- **События по пользователям**
  - Чаще всего на одного пользователя приходится 5-9 действий (36% пользователей)
  - 15.6% пользователей совершают меньше 5 действий
  - 75% пользователей за период совершили до 19 действий, большее количество действий - редкость

- **Действия по дням:**
  - не наблюдается определенной тенденции, действия по дням распределены неравномерно
  - доля целевых действий в каждый из дней крайне мала, **действия в связке "целевое"+"взаимодействие с карточкой" ни в один из дней не достигает даже половины от всех действий**

- **Действия за одну сессию**
  - **20% сессий включают в себя только одно действие**
  - медианное количество действий за сессию: 4

    - большинство сессий ограничиваются 19 действиями, большее количество действий - редкость (7.2%)

4. Конверсия в целевое действие. Анализ целевых действий

- **по уникальным пользователям: 22.85%**
    - большинство пользователей (73%) не совершают целевое действие
- **по сессиям: 16.43%**
    - большинство сессий заканчиваются "ничем" - пользователи не находят нужное объявление
- **По действиям: 6.10%**
    - конверсия носит ознакомительный характер, большинство действий - нецелевые, что ожидаемо

Рекоммендации по повышению конверсий:

- тестирование пути пользователя, не сталкивается ли пользователь с техническими проблемами на любом из этапов
- проверить настройки рекоммендаций, таргетирования
- корректно ли работает поиск через 'search' и 'map'
- есть ли какие-то рекоммендации по оформлениюо объявлений? Контролируется ли как-то этот момент? Среди всех действий только 13.5% относятся к просмотру фото - нет ли проблемы в том, что авторы объявлений загружают неинформативное или не соответсвующие действительности фото?

Невозможно определить воронку - нет обязательного события перед целевым, даны не все события *см.пункт 3.4.2*

- **Положительные наблюдения:**
    - чаще всего пользователь совершает целевое действие в первый день в приложении
    - для совершения целевого действия обычно нужно не более 3-минут с начала сессии
    - встречаются пользователи, совершающие 2 и более целевых действий за одну сессию

Первые две метрики необходимо поддерживать на том же уровне, третью повыашть всеми способами.

- **Отрицательные наблюдения:**
    - чаще всего после первой успешной сессии пользователь не совершает вторую такую же.

Это очень важная метрика, необходимо изучить ее более глубоко: не получил ли пользователь какой-то неудачный опыт взаимодействия с платформой после совершения целевого действия? Почему чаще всего он не возвращается?

Сегментация пользователей

Пользователи разделены на 4 сегмента на основании среднего количества действий за сессию

```
* A - пользователи с "активными" сессиями: > 10 действий за сессию
* B - пользователи с менее активными сессиями: от 6 до 10 действий
включительно
* C - пользователи с количество сессий меньше медианного, но больше Q1: от 4
включительно до 6 действий
```

```
* D - пользователи с малоактивными сессиями: < 4 действий за сессию
```

Определены группы-лидеры: сегменты A (высокая конверсия в уникальных пользователях, больше успешных сессий, очень много сессий с двумя и более целевыми действия) и D (лучшая конверсия в уникальных пользователях, лучшее удержание)

Рекомендуется адаптировать приложение на основе поведенческих паттернов обеих групп-лидеров. Ориентируемся на долгие, активные сессии и на быстрые сессии, отвечающие на нужный запрос.

- **Поведенческий паттерн группы A:**

Активные, долгие сессии: пользователи любят долго "сидеть" в приложении. Сессии насыщены событиями - клиенты используют весь функционал приложения. Рекомендуется:

- проработка многовариативности пользовательского пути
- проверить (усовершенствовать при необходимости) корректность рекоммендаций
- пользователь любит проводить время в приложении, но редко возвращается, проработать Retention Rate

- **Поведенческий паттерн группы D:**

Короткие, малонасыщенные действиями сессии. Пользователи ценят возможность быстро найти нужное объявление. Рекомендуется:

- максимально понятный интерфейс:
- проверить/усовершенствовать корректность работы поиска и всех его фильтров и сортировок
- проработать информативность карточек на превью
- адаптировать возможность короткого пути до целевого действия

Проверка гипотез

Проверены две гипотезы с помощью z-теста, для обеих установлен уровень статистической значимости в 0.01

1. Гипотеза о разнице конверсий пользователей, пришедших из Yandex и Google

Нет оснований считать разницу конверсий статистически значимой, **нельзя зафиксировать победу ни одной из групп**

2. Гипотеза о разнице конверсий пользователей совершающих 'favorites_add' и не совершающих это действие

Зафиксирована победу первой группы: **пользователи, добавляющие объявления в избранное имеют конверсию выше, чем пользователи, не совершающие это действие.**

Ниже дана ссылка на презентацию с более развернутыми выводами и рекомендациями, и ссылка на дашборд.

Презентация

Ссылка на презентацию: https://disk.yandex.ru/i/8LzF9RbRZxIz6Q

Дашборд

Ссылка на дашборд в Tableau:
https://public.tableau.com/app/profile/.10794569/viz/useless_thing_dash/Dashboard3?publish=yes