Исследование результатов A/B-теста и поиск инсайтов инвестиционного приложения

- Автор: Егорова Ольга
- 2025 г.

Введение

AlphaTrade – это инвестиционное приложение (доступное на смартфонах и через веб-платформу), где пользователи могут вкладывать деньги в:

- Акции
- Валюты
- Криптовалюты
- Биржевые фонды и другие активы.

Целевая аудитория: начинающие инвесторы.

Приложение ориентировано на рынки с разной экономикой: Мексику, Бразилию, Колумбию и Аргентину. Здесь растёт интерес к инвестициям.

Появилась гипотеза: клиенты теряют деньги из-за недостаточной финансовой грамотности и непонимания рисков. Они частов вкладываются в высокорисковые активы (например, криптовалюту) без должного понимания, теряют деньги, разочаровываются, что снижает вовлечённость.

Решение: улучшенный онбординг с объяснением какие бывают типы активов, какие риски с ними связаны, как диверсифицировать портфель.

Опасения: слишком сложный онбординг может оттолкнуть новых пользователей, уменьшить число пополнений и рискованных сделок (которые приносят комиссии).

Что сделано? **A/B-эксперимент**

Период: в эксперименте участвовали новые пользователи, которые зарегистрировались в приложении со 2 по 15 июня 2025 года. Пользователей случайным образом разделили на две равные группы:

- Контрольная группа: стандартный онбординг без обязательного обучения.
- Тестовая группа: улучшенный онбординг с подробным обучением.

После разделения активность пользователей анализировали в течение недели.

Цель эксперимента — оценить, как обновлённый онбординг влияет на поведение пользователей и их дальнейшую инвестиционную активность.

При внедрении новой фичи команда продукта выдвинула такие гипотезы:

1. Гипотеза пользы (рост)
Новый обучающий онбординг повысит лояльность: пользователи, лучше понимая риски, будут чаще пополнять счёт второй раз.

Почему?

- Старый онбординг не объяснял правила инвестиций → новички теряли деньги с первого раза и уходили.
- Новый формат даст реалистичные ожидания → меньше разочарований → больше повторных депозитов.

2. Гипотеза риска (падение конверсии)
Подробные предупреждения о рисках снизят конверсию в первый депозит, особенно у осторожных пользователей.

Почему?

- Часть новичков, увидев предупреждения, испугается и передумает пополнять счёт.
- Особенно критично для Латинской Америки: низкая финансовая грамотность + высокий страх потерь.

3. Гипотеза "осознанного риска"
После нового онбординга те, кто сознательно выберет высокорисковые активы, станут более лояльными:

- Реже будут терять все деньги (так как поймут риски).
- Чаще вернутся за вторым депозитом.

Контраст со старым подходом: Раньше пользователи вкладывались в рисковые активы наугад → быстро теряли деньги → уходили.

Чтобы комплексно оценить эффект от нового обучающего онбординга, команда AlphaTrade **отслеживает 4 ключевые метрики**, которые помогут увидеть не только краткосрочное влияние на конверсию, но и долгосрочное поведение пользователей:

1. Ключевая метрика
Средняя сумма всех депозитов на одного пользователя (включая тех, кто не пополнил счёт)

Зачем?

- Показывает общий экономический эффект от изменений.
- Если он растёт – значит, либо больше людей пополняют счёт, либо увеличиваются суммы вложений.
- Если падает – новые пользователи стали реже или меньше инвестировать.

Ожидание: Не должно снизиться (иначе бизнес-модель пострадает)

2. Барьерная метрика
Конверсия из регистрации в первый депозит

Зачем?

- Показывает, не отпугнул ли новый онбординг новичков.
- Если упадёт – значит, обучение слишком сложное или пугающее.

Ожидание: Допустимо небольшое снижение (если компенсируется ростом других метрик), но сильное падение – стоп-сигнал.

3. Вспомогательная метрика 1 (Качество вовлечения)
Конверсия из первого депозита во второй

Зачем?

- Показывает, стали ли пользователи более лояльными после обучения.
- Если растёт – значит, онбординг помог снизить потери и повысить доверие.

Ожидание: Значимый рост (главный индикатор успеха гипотезы).

4. Вспомогательная метрика 2 (Глубина инвестиций)
Средняя сумма депозитов на пользователя, который пополнил счёт хотя бы раз

Зачем?

- Показывает, стали ли активные инвесторы вкладывать больше.
- Если растёт – значит, обучение помогло им увереннее управлять капиталом.

Ожидание: Рост (особенно если пользователи стали осознаннее диверсифицировать риски).

Идеальный сценарий после эксперимента:

- Ключевая метрика (средний депозит на всех пользователей) – не упала.
- Барьерная метрика (конверсия в первый депозит) – незначительное снижение или стабильность.
- Вспомогательные метрики (повторные депозиты и средний чек активных) – выросли.

Такой набор метрик позволяет проверить:

1. Не навредили ли мы притоку новых пользователей? (барьерная метрика).
2. Улучшили ли мы удержание и глубину инвестиций? (вспомогательные метрики).
3. Получили ли общий рост монетизации? (ключевая метрика)

Задачи

1. Анализ исторических данных

Для понимания текущих проблем в продукте необходимо изучить исторические данные:

- Поведение новых пользователей, в том числе динамику привлечения, сегментацию и ключевые этапы воронки действий.
- Метрики, связанные с внесением депозитов, в том числе средние суммы депозитов.

2. Анализ данных A/B-теста

- Сравнение поведения пользователей в контрольной и тестовой группах, оценка статистической значимости изменений.

3. Глубокий анализ платящих пользователей
Исследование влияния нового онбординга на поведение платящих пользователей. Бутстрап-анализ для сравнения распределений депозитов:

- Как изменились нижние перцентили (осторожные пользователи)?
- Как изменились верхние перцентили (крупные инвесторы)?

4. Выводы и рекомендации
Понять, как обновлённый онбординг повлиял на ключевые бизнес-метрики, найти точки роста и сформулировать рекомендации по улучшению пользовательского опыта и монетизации.

Данные

1. Датасет `hist.csv` содержит исторические данные о ключевых действиях новых пользователей, привлечённых в период с 1 апреля по 1 июня 2025 года включительно. В датасете собраны действия пользователей до оформления второго депозита.
2. Датасет `abt.csv` содержит данные A/B-эксперимента — все действия новых пользователей, которые зарегистрировались со 2 по 15 июня 2025 года включительно. Данные собраны в рамках проверки гипотезы о влиянии нового онбординга на поведение и активность пользователей. Пользователи уже распределены по группам A/B-эксперимента.

Общие поля датасетов:

- `user_id` — уникальный идентификатор пользователя;
- `country_code` — код страны пользователя в формате ISO;
- `platform` — устройство, с которого пользователь взаимодействует с продуктом;
- `first_ts` — время первого появления пользователя в системе;
- `first_dt` — дата первого появления пользователя;
- `event_ts` — время события;
- `event_name` — название события;
- `amount` — сумма пополнения депозита;
- `asset` — тип приобретённого актива;
- `risk_level` — уровень риска актива.

В датасете `abt.csv` содержатся два дополнительных поля:

- `ab_test` — название A/B-эксперимента;
- `group` — пользовательская группа A/B-эксперимента.

План проекта

1. Загрузка исторических данных и их предобработка
2. Исследовательский анализ исторических данных
3. Исследование результатов A/B эксперимента

4. Анализ изменений суммы депозитов на платящего пользователя
5. Выводы

Часть 1

Загрузка исторических данных и их предобработка

In [1]:
```
# Импортируем необходимые библиотеки
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# для математических расчетов
from math import  fabs

# χ²-тест на гомогенность
from scipy.stats import chisquare
# χ²-тест на независимость двух категориальных переменных
from scipy.stats import chi2_contingency

# для z-теста пропорций, расчета эффекта и мощности
from statsmodels.stats.proportion import proportion_effectsize
from statsmodels.stats.proportion import proportions_ztest
from statsmodels.stats.power import zt_ind_solve_power


from scipy.stats import ttest_ind
```

Загрузка и знакомство с данными

In [2]:
```
# Загружаем исторические данные из CSV-файла
url = "https://drive.google.com/uc?export=download&id=1Nds8p2ZzL1jR6T5SUvSgOOKaRYqcG4Bq"
df = pd.read_csv(url, parse_dates=['first_dt', 'event_ts', 'first_ts'])
```

In [3]:
```
# Получаем основную информацию
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238059 entries, 0 to 238058
Data columns (total 10 columns):
 #   Column        Non-Null Count   Dtype         
---  ------        --------------   -----         
 0   user_id       238059 non-null  object        
 1   country_code  238059 non-null  object        
 2   platform      238059 non-null  object        
 3   first_ts      238059 non-null  datetime64[ns]
 4   first_dt      238059 non-null  datetime64[ns]
 5   event_ts      238059 non-null  datetime64[ns]
 6   event_name    238059 non-null  object        
 7   amount        33093 non-null   float64       
 8   asset         15392 non-null   object        
 9   risk_level    15392 non-null   object        
dtypes: datetime64[ns](3), float64(1), object(6)
memory usage: 18.2+ MB
```

In [4]:
```
# Выводим статистики для всех полей
df.describe(include='all')
```

Out[4]:

	user_id	country_code	platform	first_ts	first_dt	event_ts	event_name	amount	asset	risk_level
count	238059	238059	238059	238059	238059	238059	238059	33093.000000	15392	15392
unique	41032	4	2	NaN	NaN	NaN	8	NaN	3	3
top	7be8792d-186b-44fa-a1e1-4b6f0aa28124	BR	mobile	NaN	NaN	NaN	install / open_web	NaN	option	high
freq	8	71120	166698	NaN	NaN	NaN	41032	NaN	5189	7740
mean	NaN	NaN	NaN	2025-05-02 01:15:55.141006848	2025-05-01 09:56:10.897130752	2025-05-02 14:07:17.970582784	NaN	113.527967	NaN	NaN
min	NaN	NaN	NaN	2025-04-01 00:06:34	2025-04-01 00:00:00	2025-04-01 00:06:34	NaN	27.000000	NaN	NaN
25%	NaN	NaN	NaN	2025-04-16 13:58:16	2025-04-16 00:00:00	2025-04-17 00:09:02	NaN	90.000000	NaN	NaN
50%	NaN	NaN	NaN	2025-05-01 21:35:50	2025-05-01 00:00:00	2025-05-02 12:55:17	NaN	109.000000	NaN	NaN
75%	NaN	NaN	NaN	2025-05-17 14:57:18	2025-05-17 00:00:00	2025-05-18 02:21:01	NaN	131.000000	NaN	NaN
max	NaN	NaN	NaN	2025-06-01 23:59:52	2025-06-01 00:00:00	2025-06-09 20:24:42	NaN	524.000000	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	34.472458	NaN	NaN

In [5]:
```
# Получаем данные по типу платформы для каждого пользователя
df_user_and_platform = df[['user_id', 'platform']].drop_duplicates()

# Доли платформ
df_user_and_platform['platform'].value_counts(normalize=True)
```

Out[5]:
```
platform
mobile    0.700868
web       0.299132
Name: proportion, dtype: float64
```

In [6]:
```
# Доли типов активов
df['asset'].value_counts(normalize=True)
```

Out[6]:
```
asset
option    0.337123
stock     0.335434
crypto    0.327443
Name: proportion, dtype: float64
```

In [7]:
```
# Доли активов по степени рисков
df['risk_level'].value_counts(normalize=True)
```

Out[7]:
```
risk_level
high      0.502859
medium    0.345959
low       0.151182
Name: proportion, dtype: float64
```

In [8]:
```
# Выводим названия всех возможных событий
df['event_name'].unique()
```

Out[8]:
```
array(['install / open_web', 'introduction', 'registration', 'main_page',
       'onboarding_complete', 'first_deposit', 'asset_purchase',
       'second_deposit'], dtype=object)
```

Дубликаты в данных

Убедимся, что в данных нет явных дубликатов и что в одну дату и временя не произошли различные события:

In [9]:
```
# Выводим количество полных дубликатов
print("Количество полных дубликатов:", df.duplicated().sum())
```

```
Количество полных дубликатов: 0
```

In [10]:
```
# создадим список столбцов
list_name_columns = df.columns

# Найдем количество дубликатов по всем столбцам, кроме 4-х последних
print("Количество неявных дубликатов:", df.duplicated(subset=list_name_columns[:-4], keep='first').sum())
```

```
Количество неявных дубликатов: 0
```

Корректность данных

Проверим, что для каждого пользователя есть только одна отметка о принадлежности к определенной стране и каждый пользователь использует только 1 платформу:

In [11]:
```
# Группируем данные по пользователям и для каждого находим число уникальных кодов стран,
# оставляем тех пользователей, которые имеют больше 1 страны и считаем их количество
print("Количество пользователей с более чем одной страной:")
sum(df.groupby('user_id')['country_code'].nunique() > 1)
```

```
Количество пользователей с более чем одной страной:
```

Out[11]:
```
0
```

In [12]:
```
# Группируем данные по пользователям и для каждого находим число уникальных платформ,
# оставляем тех пользователей, которые имеют больше 1 платформы
print("Количество пользователей с более чем одной платформой:")
sum(df.groupby('user_id')['platform'].nunique() > 1)
```

```
Количество пользователей с более чем одной платформой:
```

Out[12]:
```
0
```

Убедимся, что в данных нет событий, которые произошли до регистрации:

In [13]:
```
print("Количество пользователей с событиями, которые произошли до регистрации:")
df[df['event_ts'] < df['first_dt']]['user_id'].count()
```

```
Количество пользователей с событиями, которые произошли до регистрации:
```

Out[13]:
```
0
```

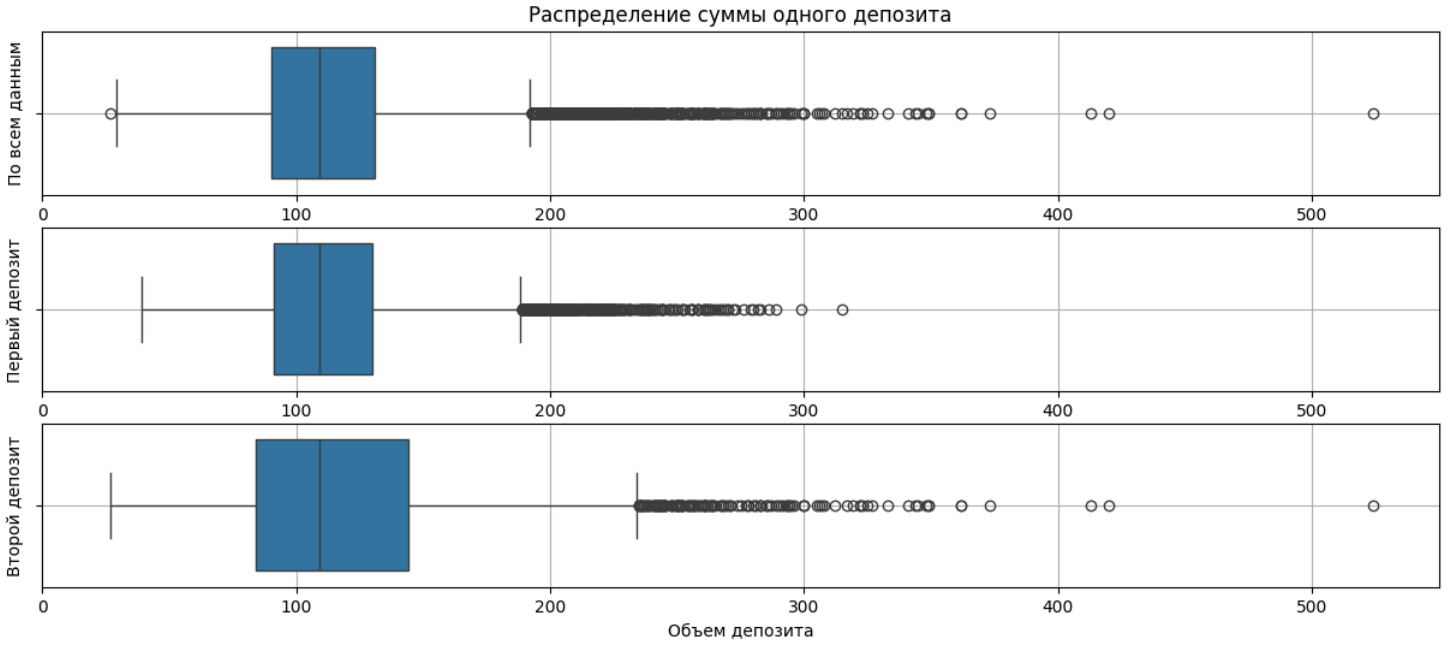
In [14]:
```
print(f"Период регистрации: с {df['first_dt'].dt.date.min()} по {df['first_dt'].dt.date.max()}")
print(f"Период событий: с {df['event_ts'].dt.date.min()} по {df['event_ts'].dt.date.max()}")
```

```
Период регистрации: с 2025-04-01 по 2025-06-01
Период событий: с 2025-04-01 по 2025-06-09
```

Распределение суммы депозита

In [15]:
```
# Посмотрим на распределение суммы депозита по все данным, только для первого депозита и только для второго:

# Строим два графика
fig, axes = plt.subplots(3, 1, figsize=(15, 6))

# Строим горизонтальные диаграммы размаха
sns.boxplot(df, x='amount', ax=axes[0])
sns.boxplot(df[df['event_name'] == 'first_deposit'], x='amount', ax=axes[1])
sns.boxplot(df[df['event_name'] == 'second_deposit'], x='amount', ax=axes[2])

# Добавляем заголовок и подписи
axes[0].set_title('Распределение суммы одного депозита')
axes[0].set_xlabel(' ')
axes[0].set_ylabel('По всем данным')
axes[0].set_xlim(0, 550)

axes[1].set_xlabel(' ')
axes[1].set_ylabel('Первый депозит')
axes[1].set_xlim(0, 550)

axes[2].set_xlabel('Объем депозита')
axes[2].set_ylabel('Второй депозит')
axes[2].set_xlim(0, 550)

# Добавляем сетку
axes[0].grid(True)
axes[1].grid(True)
```

```
axes[2].grid(True)

# Отображаем график
plt.show()
```



Для первого депозита диаграмма более компактна:

- первый депозит вероятно делается небольшими суммами, поскольку пользователи знакомятся с платформой.
- 50% пользователей инвестируют около 90–130 у.е., вероятно эта сумма комфортный и безопасный диапазон для начала.
- выбросы больше 200 могут указывать на опытных инвесторов, сразу начинающих с крупных сумм.

Для второго депозита диаграмма шире, она охватывает как более низкие, так и более высокие значения:

- те, кто пополнили второй раз, вероятно, получили положительный опыт и готовы рисковать больше. Это объяснет более высокие значения.
- А пользователи с негативным опытом готовы пробовать еще , но с менее существенными суммами. Это объясняет наличие более низких значений.

Выбросы могут искажать средние значения, это стоит учитывать при дальнейшем исследовании.

Промежуточные выводы:

1. Названия столбцов отражают их содержимое, они понятны и удобны для работы.
2. Пропуски. Три столбца `amount` , `asset` и `risk_level` содержат пропуски, которые обусловлены спецификой данных.
3. Типы данных корректны.
4. Типы платформ: mobile и web. Большая часть пользователей использует для взаимодействия мобильную версию приложения (70%).
5. Временной промежуток привлечения пользователей: 01.04.2025 - 01.06.2025, что соответствует исходным данным. При этом 50% пользователей зарегистрировались в первый месяц, 50% - во второй.
6. Временной промежуток событий пользователей чуть больше: 01.04.2025 - 09.06.2025
7. Сумма пополнения депозита: от 27 у.е. до 524 у.е. Среднее значение (113 у.е.) несколько выше медианного (109 у.е.), что говорит о наличии в данных высоких значений.
8. Типы приобретенных активов (`stock` - акции, `option` - опционы, `crypto` - криптовалюты) распределены примерно одинаково. Опционы чуть более популярны (33,7%) , а криптовалюта - менее (32,7%).
9. Уровень риска активов. Наиболее полулярны среди пользователей активы с высоким уровнем риска `high` , их доля больше 50%. На активы с низким уровнем риска `low` приходится только 15,1%
10. Дубликаты в данных не выявлены.
11. Данные корректны. Каждый пользователь "привязан" к конкретной стране и типу устройст. Все события произошли после регистаций.

Исследовательский анализ исторических данных

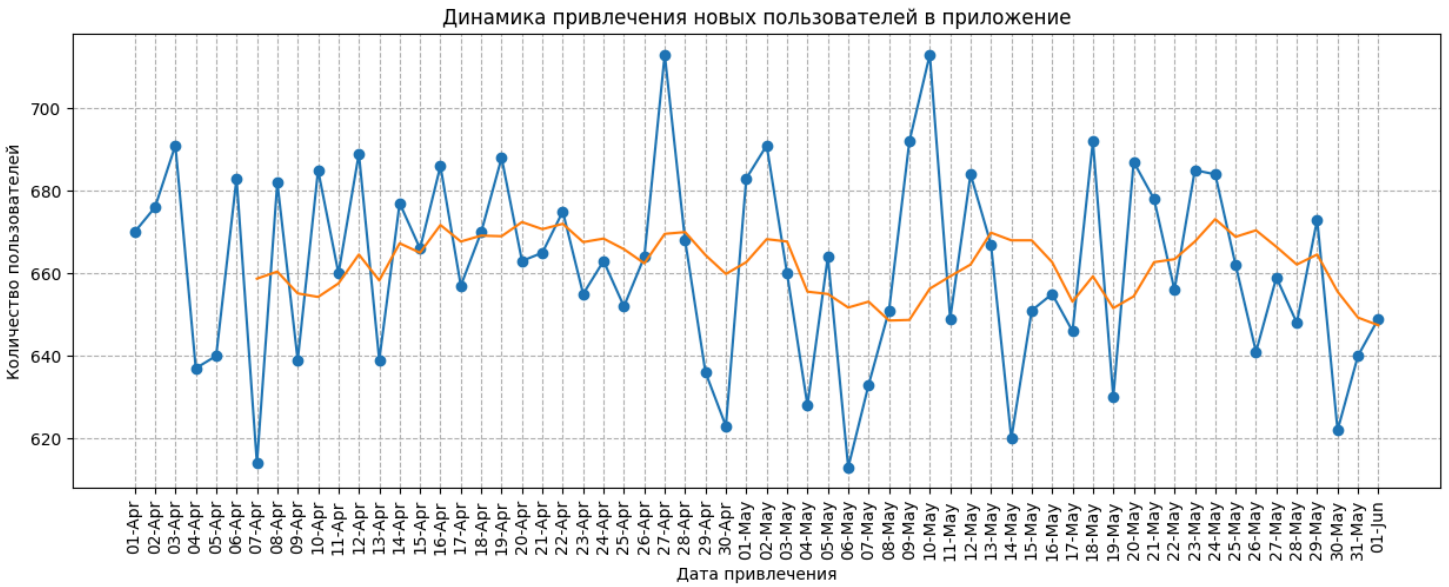
Анализ новых пользователей.

Посмотрим на динамику привлечения новых пользователей в приложение.

In [16]:
```
# Подготовим данные
# Для каждого дня привлечения расчитаем количесво привлеченных пользователей и переименуем поле в 'count_user'
df_count_user = df.groupby('first_dt')['user_id'].nunique().reset_index(name='count_user')
```

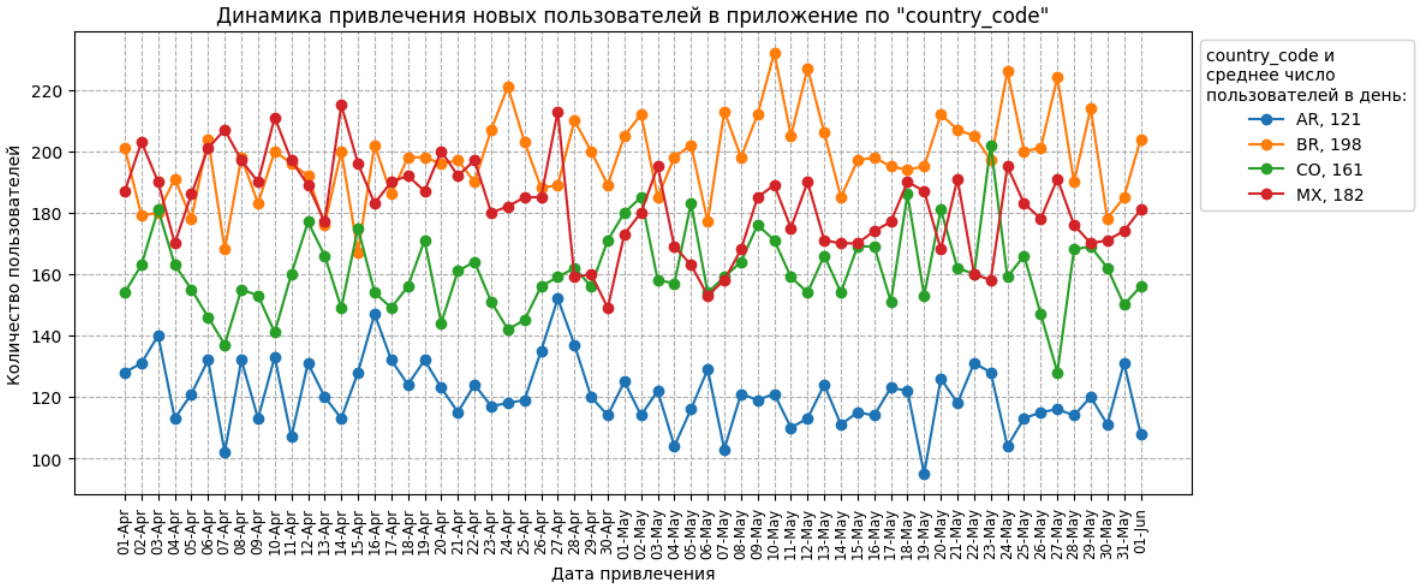
In [17]:
```
# Задаем размер графика
plt.figure(figsize=(15, 5))

# Строим линейный график
plt.plot(df_count_user['first_dt'], df_count_user['count_user'], marker='o')

# Строим линейный график скользящей средней
plt.plot(df_count_user['first_dt'], df_count_user['count_user'].rolling(7).mean(), color='tab:orange', label='Скользящая средняя (окно 7 дней)')


# Добавляем заголовок
plt.title('Динамика привлечения новых пользователей в приложение')

# Добавлем подписи осей
plt.xlabel('Дата привлечения')
```

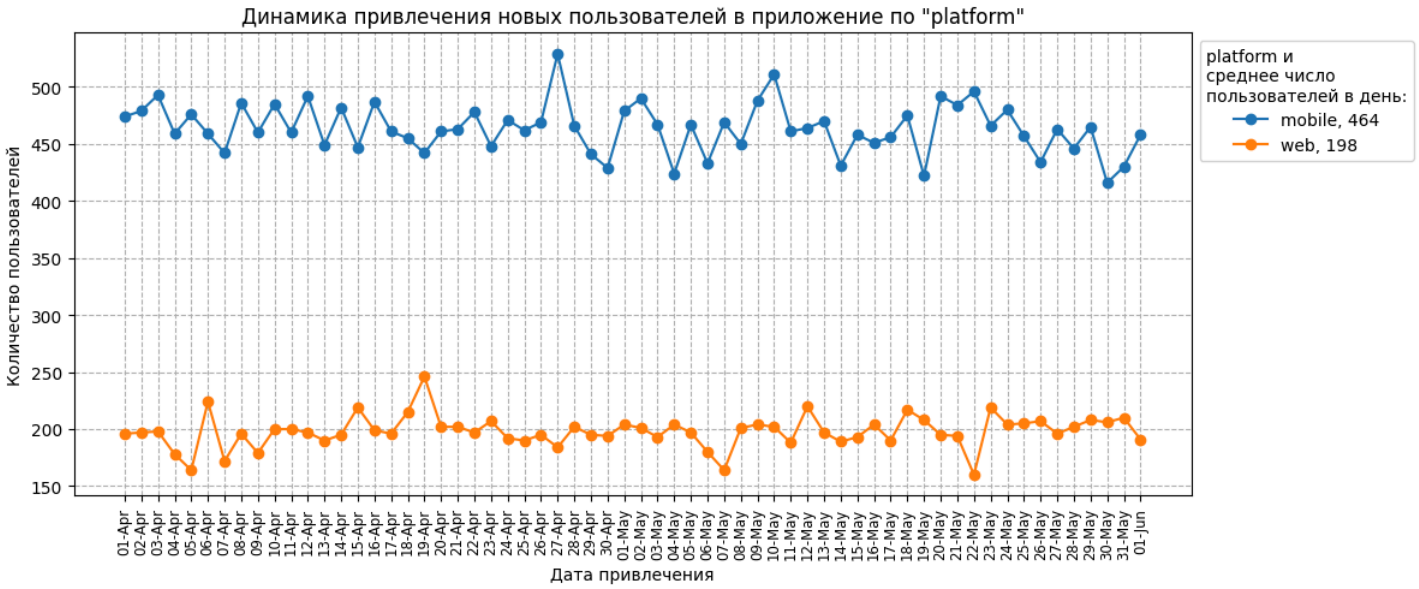
```
plt.ylabel('Количество пользователей')

# Отображаем дополнительные линии
plt.grid(linestyle='--')

# Создаем список меток
labels = df_count_user['first_dt'].sort_values().dt.strftime('%d-%b')
# Передаем позиции,  метки и угол поворота
plt.xticks(df_count_user['first_dt'], labels, rotation=90)

# Отображаем график
plt.show()

print("Минимальное число привлеченных пользователей:", df_count_user['count_user'].min())
print("Максимальное число привлеченных пользователей:", df_count_user['count_user'].max())
print("Среднее количество привлеченных пользователей:", int(df_count_user['count_user'].mean().round()))
```




```
Минимальное число привлеченных пользователей: 613
Максимальное число привлеченных пользователей: 713
Среднее количество привлеченных пользователей: 662
```

In [18]:
```
# Создаем список с названиями полей
list_col = ['country_code', 'platform']

# В цикле строим линейный график
for col in list_col:
    # Для каждого дня привлечения расчитаем количество привлеченных пользователей и переименуем поле в 'count_user'
    df_count_user = df.groupby([col,'first_dt'])['user_id'].nunique().reset_index(name='count_user')

    # Получаем уникальные значения для  'col'
    segments = df_count_user[col].unique()

    # Задаем размер графика
    plt.figure(figsize=(12, 5))

    # Для каждого уникального сегмента строим линейный график
    for segment in segments:
        # Фильтруем данные
        df_plot = df_count_user[df_count_user[col] == segment]
        # Строим линейный график
        plt.plot(df_plot['first_dt'], df_plot['count_user'],
                 marker='o',
                 label=segment+f", {int(df_plot['count_user'].mean().round())}")

    # Добавляем заголовок
    plt.title(f'Динамика привлечения новых пользователей в приложение по "{col}"')

    # Добавлем подписи осей
    plt.xlabel('Дата привлечения')
    plt.ylabel('Количество пользователей')

    # Отображаем дополнительные линии
    plt.grid(linestyle='--')

    # Создаем список меток
    labels = df_plot['first_dt'].sort_values().dt.strftime('%d-%b')
    # Передаем позиции,  метки и угол поворота
    plt.xticks(df_plot['first_dt'], labels, rotation=90, fontsize='small')

    # Добавляем легенду
    plt.legend(title=f'{col} и \nсреднее число \nпользователей в день:', bbox_to_anchor=(1, 1))

    # Отображаем график
    plt.show()
```





Ежедневно в среднем привлекается по 662 пользователя. На протяжении рассматриваемого периода этот показатель варьируется в пределах +- 7,5%: от 613 до 713 пользователей в день.

По платформам:

- для `web-версии` приложения в среднем ежедневно привлекается по 198 пользователей в день, что составляем около 30% от общего числа привлеченных.
- для `mobile-версии` ежедневно в среднем 464 пользователя или 70% от общего числа.

По странам:

- `BR-Бразилия` и `MX-Мексика` имеют самые высокие показатели по числу привлеченных пользователей в день: Бразилия - 198, Мексика - 182 пользователя в среднем за день. В начале периода мы наблюдаем, что графики для этих стран находятся на одном уровне, что говорит о примерно одинаковом объеме привлечения в этот период, а во второй половине - наблюдаем снижение активности пользователей из Мексики.
- `CO-Колумбия` занимает 3 место по объему - в среднем 161 пользователь в день.
- Для `AR-Аргентины` характерно наименьшее ежедневное число привлеченных пользователей - 121 пользователь в среднем за день. График для этой страны расположен ниже остальных на протяжении всего рассматриваемого периода.

Анализ воронок событий

Для начала необходимо определить порядок событий в приложении:

In [19]:
```
# Проверим совпадают ли события для mobile и web-версии
set(df[df['platform'] == 'web']['event_name']) == set(df[df['platform'] == 'mobile']['event_name'])
```

Out[19]:
```
True
```

In [20]:
```
# Выводим названия всех возможных событий
df['event_name'].unique()
```

Out[20]:
```
array(['install / open_web', 'introduction', 'registration', 'main_page',
       'onboarding_complete', 'first_deposit', 'asset_purchase',
       'second_deposit'], dtype=object)
```

In [21]:
```
# Создаем список событий
list_event = ['install / open_web', # установка или открытие web сайта
              'introduction', # вероятно страница приветствия
              'registration', # регистрация
              'main_page', # главная страница
              'onboarding_complete', # завершение онбординга
              'first_deposit', # первый депозит
              'asset_purchase', # покупка активов
```

```
            'second_deposit' # второй депозит
        ]
```

In [22]:
```
# Проверим, что порядок событий в созданном списке корректный

# Найдем одного любого пользователя с максимальным количеством уникальных событий
user_total_count_event = (df.groupby('user_id')['event_name'].nunique() == 8).nlargest(1).reset_index()

# Выберем все записи для этого пользователя и отсортируем по возрастанию времени событий
user_total_event = df[df['user_id'].isin(user_total_count_event['user_id'])].sort_values(by='event_ts')

# Удаляем дубликаты в поле 'event_name' (если они есть)
list_user_total_event = user_total_event['event_name'].drop_duplicates().to_list()

# Сравниваем наш создный список 'list_event' с полученным 'list_user_total_event'
if list_event == list_user_total_event:
    print("Список list_event корректный. Порядок событий совпадает")
else:
    print("Список list_event не корректный или порядок событий не совпадает")
```

```
Список list_event корректный. Порядок событий совпадает
```

Анализ пользовательского пути в целом

In [23]:
```
# Считаем число уникальных пользователей на каждом шаге

# Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
df_funnel = df.groupby('event_name')['user_id'].nunique().reindex(list_event).reset_index()

# Переименовываем поля
df_funnel = df_funnel.rename(columns = {'event_name':'step', 'user_id':'users'})

# Выводим получившийся датафрейм
df_funnel
```

Out[23]:

	step	users
0	install / open_web	41032
1	introduction	41032
2	registration	38133
3	main_page	35040
4	onboarding_complete	34337
5	first_deposit	27685
6	asset_purchase	15392
7	second_deposit	5408

In [24]:
```
# Добавляем поле с конверсией для классической воронки и округляем до 1 знака после запятой
# Каждое значение делим на первое
df_funnel['conversion_from_first_%'] = df_funnel['users'].div(df_funnel.loc[0]['users']).round(3) * 100

# Добавляем поле с конверсией для step-by-step воронки и округляем до 1 знака после запятой
# Каждое значение делим на предыдущее
df_funnel['conversion_from_previous_%'] = df_funnel['users'].div(df_funnel['users'].shift(1)).round(3) * 100

# Выводим результат
df_funnel
```

Out[24]:

	step	users	conversion_from_first_%	conversion_from_previous_%
0	install / open_web	41032	100.0	NaN
1	introduction	41032	100.0	100.0
2	registration	38133	92.9	92.9
3	main_page	35040	85.4	91.9
4	onboarding_complete	34337	83.7	98.0
5	first_deposit	27685	67.5	80.6
6	asset_purchase	15392	37.5	55.6
7	second_deposit	5408	13.2	35.1

In [25]:
```
# Строим два графика
fig, axes = plt.subplots(2, 1, figsize=(12, 6))

# Классическая воронка:

# Опускаем первый этап 'install / open_web'
axes[0].bar(df_funnel.loc[1:]['step'], df_funnel.loc[1:]['conversion_from_first_%'])

axes[0].set_title('Классическая воронка: конверсия от первого этапа\n')
axes[0].set_ylabel('Конверсия\nот первого этапа (%)')

# Добавляем подписи барам
axes[0].bar_label(axes[0].containers[0], fmt='%.1f%%', fontsize=10)

# Убираем границы
axes[0].spines['top'].set_visible(False)
axes[0].spines['right'].set_visible(False)

# Установим максимальное значение
axes[0].set_ylim(0, 100)


# step-by-step воронка:

# Опускаем первый этап 'install / open_web'
```

```
axes[1].bar(df_funnel.loc[1:]['step'], df_funnel.loc[1:]['conversion_from_previous_%'])
axes[1].set_title('\nStep-by-step воронка: конверсия между этапами\n')
axes[1].set_xlabel('Этапы воронки')
axes[1].set_ylabel('Конверсия\nот предыдущего этапа (%)')

# Добавляем подписи барам
axes[1].bar_label(axes[1].containers[0], fmt='%.1f%%', fontsize=10)

# Убираем границы
axes[1].spines['top'].set_visible(False)
axes[1].spines['right'].set_visible(False)
# Установим максимальное значение
axes[1].set_ylim(0, 100)


# Автоматическая настройка отступов между графиками
plt.tight_layout()

plt.show()
```

Вывод:

Классическая воронка

Конверсия от первого шага до второго депозита составляет 13.2%.

1. Все пользователи, которые установили приложение или открыли его web-версию в полном объеме доходят до второго шага 'introduction' - страницы приветствия. Что вероятно является подтверждение того, что сайт - загружается, приложение - открывается, то есть не наблюдаются какие-либо технические сложности для пользователей.
2. На этапе регистрации 'registration' приложение теряет 7.1% пользователей. Приложение достаточно специфично и установка подобного приложения, скорее всего, является обдуманным действием, поэтому потеря даже небольшого числа пользователей на этапе регистрации на мой взгляд должна насторожить. Возможно проблемы с регистрацией возникают у определенной группы пользователей, например только в web или для определенной страны и т.д.
3. До главной страницы не доходят 14.6% всех пользователей, а обучающий онбординг не заканчивают 16.3%
4. Активно воронка начинает проседать на трех последних этапах: до первого депозита доходит 67.5% - возможно высокий порог для входа, покупка активов - лишь 37.5% - отсутствие достаточных знаний в области или технические барьеры, а повторный депозит - 13.2%

Step-by-tep воронка

1. Первые два этапа - аналогично классической воронке
2. Практически все пользователи (потеря 1%) прошедшие процесс регистрации открываю главную страницу приложения.
3. Практически все пользователи (потеря 2%) открывшие главную страницу приложения проходят обучающий онбординг. Это может говорить о высокой заинтересованности пользователей.
4. Однако при переходе на следующий шаг продукт теряет почти 20% пользователей. Первый депозит открывают лишь 80.6% пользователей прошедших обучение. Возможные причины: обучение слишком поверхностное и не дает пользователям достаточно знаний для дальнейшего использования продукта либо информация онбординга слишком сложна для начинающих инвесторов, а непонимание специфики не позволяет двигаться дальше.
5. После первого пополнения депозита лишь 55.6% пользователей совершают покупки активов. Возможно после пополнения депозита часть пользователей вновь возвращаются к онбордингу, составляют план инвестиций и, за рассматриваемый период, еще просто не успели перейти к покупкам. Возможен ограниченный выбор активов. Возможно пользователи ждут идеального момента для входа (снижение стоимости актива) и не торопятся покупать.
6. И 35.1% из тех, кто уже покупал активы вновь пополняют депозиты. С точки зрения продукта такая конверсия конечно низкая, продукту выгодно, чтобы пользователи пополняли счет и совершали покупки. Но мы не знаем сколько пользователи вкладывают в первый раз, возможно первые депозиты достаточно существенные и за такой короткий промежуток времени нет необходимости вкладывать повторно. Либо пользователи вкладывают уже разаботанные средства повторно без необходимости пополнять депозит.

Потери на последних шагах воронок могут быть связаны с особенностями поведения пользователей из разных стран, спецификой экономик стран. Кроме того, стоит проверить работоспособность приложения на различных этапах для различных типов приложения. Проверим, есть ли различия в пользовательском поведении для различных стран и типов приложений.

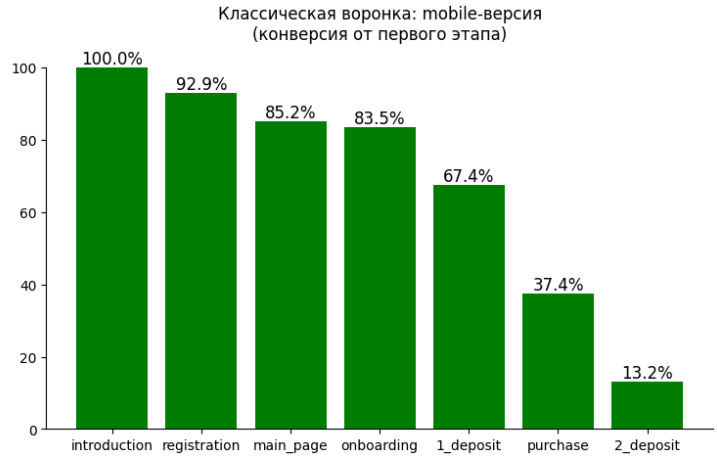
Анализ пользовательского пути для mobile и web версий

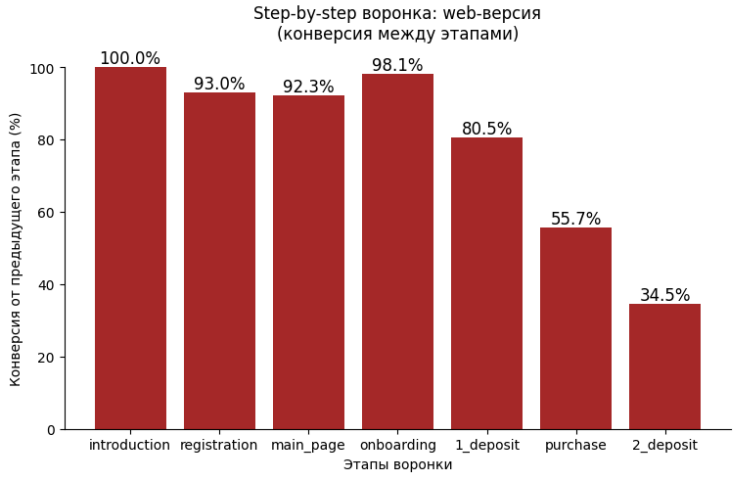
In [26]:
```
# Считаем число уникальных пользователей на каждом шаге для разных платформ:

# Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
df_funnel_mobile =\
    df[df['platform'] == 'mobile'].groupby('event_name')['user_id'].nunique().reindex(list_event).reset_index()
```

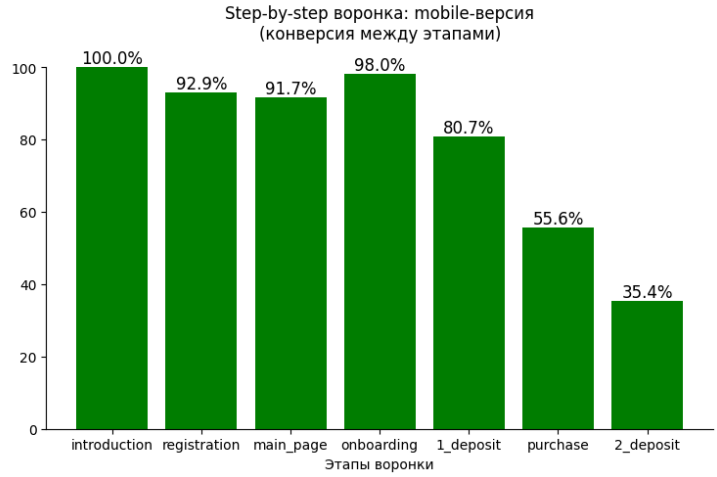
```
df_funnel_web = df[df['platform'] == 'web'].groupby('event_name')['user_id'].nunique().reindex(list_event).reset_index()

# Переименовываем поля
df_funnel_mobile = df_funnel_mobile.rename(columns = {'event_name':'step', 'user_id':'users'})
df_funnel_web = df_funnel_web.rename(columns = {'event_name':'step', 'user_id':'users'})


# Добавляем поле с конверсией для каждой платформы:

# Добавляем поле с конверсией для классической воронки и округляем до 1 знака после запятой
# Каждое значение делим на первое
df_funnel_mobile['conversion_from_first_%'] =\
    df_funnel_mobile['users'].div(df_funnel_mobile.loc[0]['users']).round(3) * 100
df_funnel_web['conversion_from_first_%'] =\
    df_funnel_web['users'].div(df_funnel_web.loc[0]['users']).round(3) * 100

# Добавляем поле с конверсией для step-by-step воронки и округляем до 1 знака после запятой
# Каждое значение делим на предыдущее
df_funnel_mobile['conversion_from_previous_%'] = df_funnel_mobile['users'].div(df_funnel_mobile['users'].shift(1)).round(3) * 100
df_funnel_web['conversion_from_previous_%'] = df_funnel_web['users'].div(df_funnel_web['users'].shift(1)).round(3) * 100
```

In [27]:

```
# Создаем список событий в укороченном варианте
new_labels = ['introduction', 'registration', 'main_page', 'onboarding', '1_deposit', 'purchase', '2_deposit']

# Строим четыре графика
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Классическая воронка для web
axes[0,0].bar(df_funnel_web.loc[1:]['step'], df_funnel_web.loc[1:]['conversion_from_first_%'], color='brown')
axes[0,0].set_title('Классическая воронка: web-версия\n(конверсия от первого этапа)\n')
#axes[0,0].set_xlabel('Этапы воронки')
axes[0,0].set_ylabel('Конверсия от первого этапа (%)')
# Добавляем подписи барам
axes[0,0].bar_label(axes[0,0].containers[0], fmt='%.1f%%', fontsize=12)
# Убираем границы
axes[0,0].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[0,0].set_ylim(0, 100)
# Установим новые подписи
axes[0,0].set_xticks(df_funnel_mobile.loc[1:]['step'] , new_labels)



# step-by-step воронка для web
axes[1,0].bar(df_funnel_web.loc[1:]['step'], df_funnel_web.loc[1:]['conversion_from_previous_%'], color='brown')
axes[1,0].set_title('\nStep-by-step воронка: web-версия\n(конверсия между этапами)\n')
axes[1,0].set_xlabel('Этапы воронки')
axes[1,0].set_ylabel('Конверсия от предыдущего этапа (%)')
# Добавляем подписи барам
axes[1,0].bar_label(axes[1,0].containers[0], fmt='%.1f%%',  fontsize=12)
# Убираем границы
axes[1,0].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[1,0].set_ylim(0, 100)
# Установим новые подписи
axes[1,0].set_xticks(df_funnel_mobile.loc[1:]['step'] , new_labels)



# Классическая воронка для Mobile
axes[0,1].bar(df_funnel_mobile.loc[1:]['step'], df_funnel_mobile.loc[1:]['conversion_from_first_%'], color='green')
axes[0,1].set_title('Классическая воронка: mobile-версия\n(конверсия от первого этапа)\n')
#axes[0,1].set_xlabel('Этапы воронки')
#axes[0,1].set_ylabel('Конверсия от первого этапа (%)')
# Добавляем подписи барам
axes[0,1].bar_label(axes[0,1].containers[0], fmt='%.1f%%', fontsize=12)
# Убираем границы
axes[0,1].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[0,1].set_ylim(0, 100)
# Установим новые подписи
axes[0,1].set_xticks(df_funnel_mobile.loc[1:]['step'] , new_labels)


# step-by-step воронка для Mobile
axes[1,1].bar(df_funnel_mobile.loc[1:]['step'], df_funnel_mobile.loc[1:]['conversion_from_previous_%'], color='green')
axes[1,1].set_title('\nStep-by-step воронка: mobile-версия\n(конверсия между этапами)\n')
axes[1,1].set_xlabel('Этапы воронки')
#axes[1,1].set_ylabel('Конверсия от предыдущего этапа (%)')
# Добавляем подписи барам
axes[1,1].bar_label(axes[1,1].containers[0], fmt='%.1f%%', fontsize=12)
# Убираем границы
axes[1,1].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[1,1].set_ylim(0, 100)
# Установим новые подписи
axes[1,1].set_xticks(df_funnel_mobile.loc[1:]['step'] , new_labels)

# Автоматическая настройка отступов между графиками
plt.tight_layout()

plt.show()
```







Вывод:

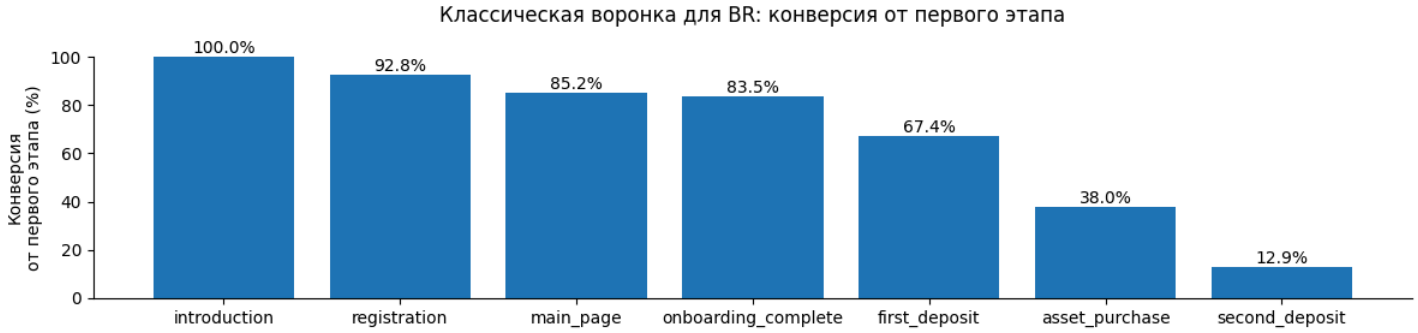
Пользовательский путь для mobile и web версий практически идентичны. Существуют незначительные различия в показателях конверсии, но в целом поведение пользователей для двух видов приложения совпадают. С технической точки зрения это может говорить об отсутствии каких-либо сложностей характерных для одного вида приложения и не характерных для другого. То есть если и существуют какие-либо технические проблемы, то пользователи разных типов приложения в одинаковой степени с ними взаимодействуют.

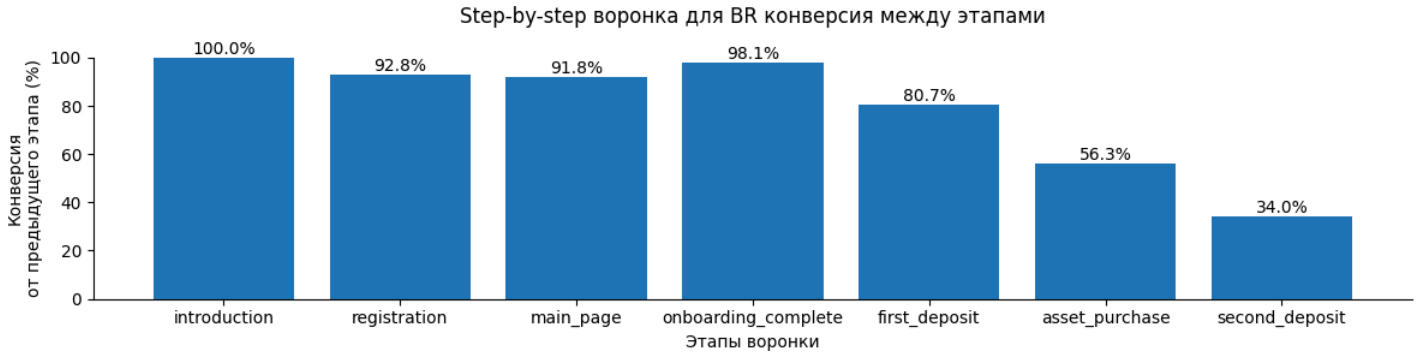
Анализ пользовательского пути для разных стран

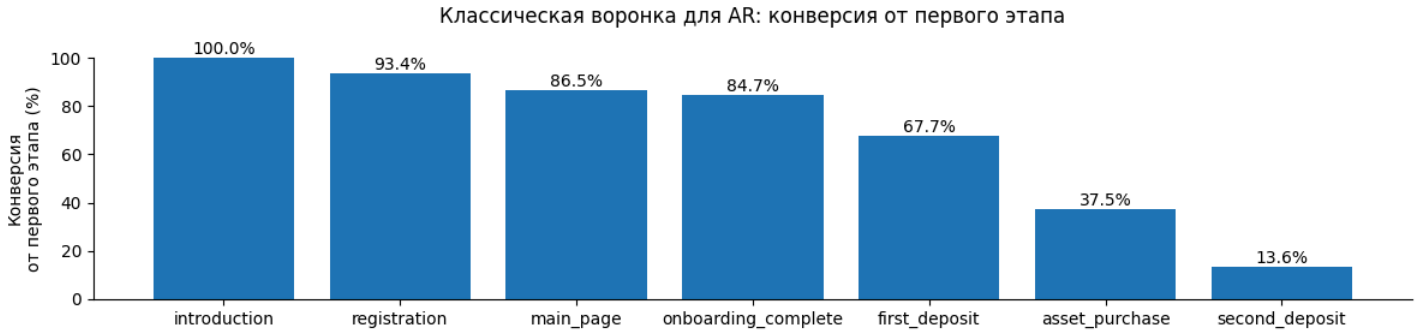
In [28]:

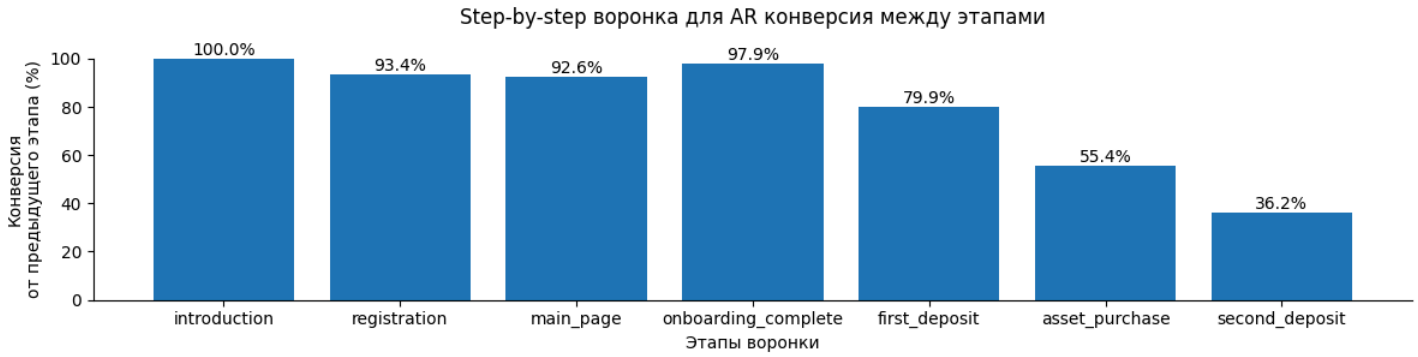
```
# Получаем уникальные значения 'country_code'
segments = df['country_code'].unique()

for segment in segments:
    # 1. Считаем число уникальных пользователей на каждом шаге для разных стран:

    # Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
    df_funnel =\
        df[df['country_code'] == segment].groupby('event_name')['user_id'].nunique().reindex(list_event).reset_index()
    # Переименовываем поля
    df_funnel = df_funnel.rename(columns = {'event_name':'step', 'user_id':'users'})


    # 2. Добавляем поле с конверсией для каждой платформы:

    # Добавляем поле с конверсией для классической воронки и округляем до 1 знака после запятой
    # Каждое значение делим на первое
    df_funnel['conversion_from_first_%'] = df_funnel['users'].div(df_funnel.loc[0]['users']).round(3) * 100

    # Добавляем поле с конверсией для step-by-step воронки и округляем до 1 знака после запятой
    # Каждое значение делим на предыдущее
    df_funnel['conversion_from_previous_%'] = df_funnel['users'].div(df_funnel['users'].shift(1)).round(3) * 100


    # 3. Строим два графика
    fig, axes = plt.subplots(2, 1, figsize=(12, 6))

    # Классическая воронка:

    # Опускаем первый этап 'install / open_web'
    axes[0].bar(df_funnel.loc[1:]['step'], df_funnel.loc[1:]['conversion_from_first_%'])
    axes[0].set_title(f'Классическая воронка для {segment}: конверсия от первого этапа\n')
    #axes[0].set_xlabel('Этапы воронки')
    axes[0].set_ylabel('Конверсия\nот первого этапа (%)')
    # Добавляем подписи барам
    axes[0].bar_label(axes[0].containers[0], fmt='%.1f%%', fontsize=10)
    # Убираем границы
    axes[0].spines[['top', 'right']].set_visible(False)
    # Установим максимальное значение
    axes[0].set_ylim(0, 100)


    # step-by-step воронка:

    # Опускаем первый этап 'install / open_web'
    axes[1].bar(df_funnel.loc[1:]['step'], df_funnel.loc[1:]['conversion_from_previous_%'])
    axes[1].set_title(f'Step-by-step воронка для {segment} конверсия между этапами\n')
    axes[1].set_xlabel('Этапы воронки')
    axes[1].set_ylabel('Конверсия\nот предыдущего этапа (%)')
```

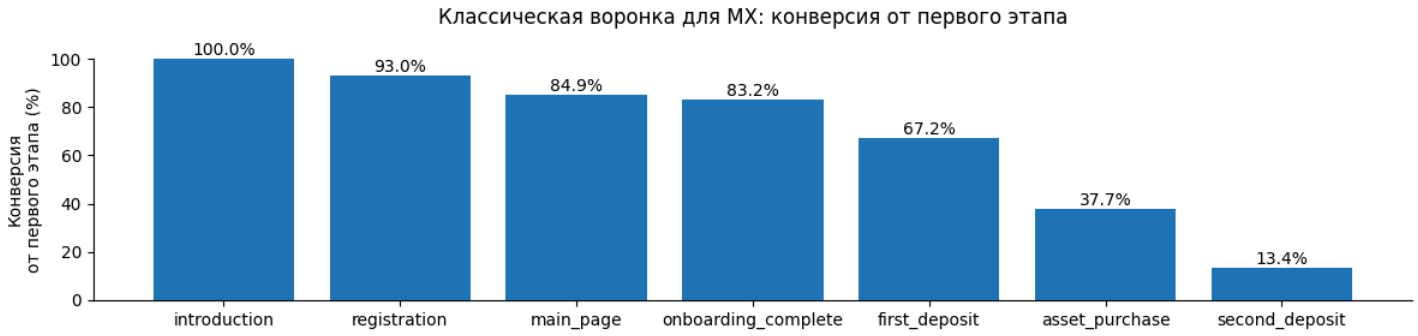
```
    # Добавляем подписи барам
    axes[1].bar_label(axes[1].containers[0], fmt='%.1f%%', fontsize=10)
    # Убираем границы
    axes[1].spines[['top', 'right']].set_visible(False)
    # Установим максимальное значение
    axes[1].set_ylim(0, 100)


    # Автоматическая настройка отступов между графиками
    plt.tight_layout()

    plt.show()
```









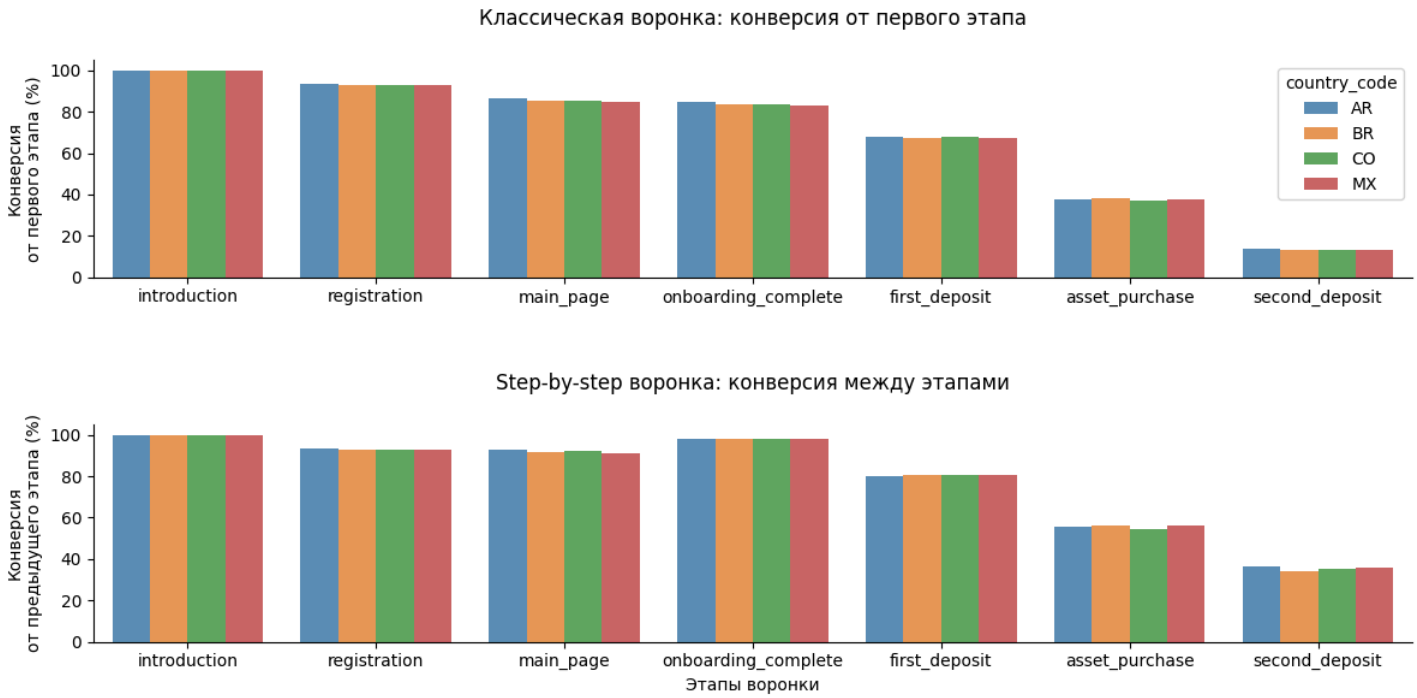


In [29]:

```
# Подготовим данные:

# Для каждой страны рассчитаем число пользователей на каждом этапе воронки
df_funnel_country = df.groupby(['country_code','event_name'])['user_id'].nunique().reset_index()

# Для корректного отображения порядка этапов воронки, преобразуем столбец 'event_name' в категориальный с указанным порядком
df_funnel_country['event_name'] = pd.Categorical(
    df_funnel_country['event_name'],
    # Указываем порядок согласно списку
    categories=list_event,
    # Учитываем порядок
    ordered=True
)

# Переименовываем поля
df_funnel_country = df_funnel_country.rename(columns = {'event_name':'step', 'user_id':'users'})

# Сортируем датафрейм по столбцу по стране и по шагу воронки
df_funnel_country = df_funnel_country.sort_values(['country_code', 'step'])

# Создаем поле с числом пользователей на первом шаге воронки (для каждой страны)
df_funnel_country['total_users'] = df_funnel_country.groupby('country_code')['users'].transform('first')

# Создаем поле с числом пользователей на предыдущем шаге воронки  (для каждой страны)
df_funnel_country['previous_users'] = df_funnel_country.groupby('country_code')['users'].shift(1)


# Добавляем поля с конверсией

# Добавляем поле с конверсией для классической воронки и округляем до 1 знака после запятой
# Каждое значение делим на первое
df_funnel_country['conversion_from_first_%'] =\
    df_funnel_country['users'].div(df_funnel_country['total_users']).round(3) * 100

# Добавляем поле с конверсией для step-by-step воронки и округляем до 1 знака после запятой
```

```
# Каждое значение делим на предыдущее
df_funnel_country['conversion_from_previous_%'] =\
    df_funnel_country['users'].div(df_funnel_country['previous_users']).round(3) * 100


# Отфильтруем первый шаг воронки
df_funnel_country = df_funnel_country[df_funnel_country['step'] != 'install / open_web']

# Сбрасываем категории, чтобы sns.barplot() не отображал отфильтрованные шаги
df_funnel_country['step'] = df_funnel_country['step'].cat.remove_unused_categories()
```

In [30]:
```
# Строим два графика
fig, axes = plt.subplots(2, 1, figsize=(12, 6))

# Классическая воронка:

# Опускаем первый этап 'install / open_web'
sns.barplot(
    data=df_funnel_country,
    x='step', y='conversion_from_first_%', hue='country_code',
    # Указываем, что рисуем на первой оси
    ax=axes[0],
    # Сделаем бары немного прозрачными
    alpha=0.8
)
axes[0].set_title('Классическая воронка: конверсия от первого этапа\n')
axes[0].set_xlabel(' ')
axes[0].set_ylabel('Конверсия\nот первого этапа (%)')
# Убираем границы
axes[0].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение axes[0].set_ylim(0, 100)


# step-by-step воронка:

# Опускаем первый этап 'install / open_web'
sns.barplot(
    data=df_funnel_country,
    x='step', y='conversion_from_previous_%', hue='country_code',
    # Указываем, что рисуем на второй оси
    ax=axes[1],
    # Сделаем бары немного прозрачными
    alpha=0.8,
    # Отключаем легенду
    legend=False
)
axes[1].set_title('\nStep-by-step воронка: конверсия между этапами\n')
axes[1].set_xlabel('Этапы воронки')
axes[1].set_ylabel('Конверсия\nот предыдущего этапа (%)')
# Убираем границы
axes[1].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение axes[1].set_ylim(0, 100)


# Автоматическая настройка отступов между графиками
plt.tight_layout()

plt.show()
```



Вывод: Воронки конверсий для пользователей из разных стран выглядят практически идентичными. Это может говорить о схожем поведении пользователей, об универсальности продукта для этих стран, а также о незаметном влиянии разных экономик этих стран на поведение пользователей в конкретный промежуток времени.

Влияние уровня риска актива на открытие второго депозита

In [31]:
```
# Создаем список событий: покупка актива и повторный депозит
list_event_part = ['asset_purchase', 'second_deposit']
# Создаем список уровней риска
list_risk_level = ['low', 'medium', 'high']
```

```
# Фильтруем датасет. Оставляем только события из списка событий
df_risk_level = df[df['event_name'].isin(list_event_part)]

# Для каждого уровня риска найдем количество пользователей, совершивших покупку актива соответствующего уровня
df_funnel_risk_level = df_risk_level.groupby(['risk_level']).agg(
    count_user_asset_purchase=('user_id', 'nunique')
).reindex(list_risk_level).reset_index()

# Выводим получившийся датафрейм
df_funnel_risk_level
```

Out[31]:

	risk_level	count_user_asset_purchase
0	low	2327
1	medium	5325
2	high	7740

In [32]:
```
# Создаем пустой словарь
dict_count_user_second_deposit = {}

for risk_level in list_risk_level:
    # Получаем список уникальных идентификаторов для выбранного уровня риска
    list_user = df_risk_level[df_risk_level['risk_level'] == risk_level]['user_id'].unique()

    # Фильтруем датасет по идентификаторам пользователей и по событию 'second_deposit'
    df_risk_level_filtered = df_risk_level[
        (df_risk_level['user_id'].isin(list_user)) & (df_risk_level['event_name'] == 'second_deposit')
    ]

    # Считаем количество пользователей с повторным депозитом для выбранного уровня риска
    count_user_second_deposit = df_risk_level_filtered['user_id'].nunique()

    # Записываем данные в словарь
    dict_count_user_second_deposit[risk_level] =  count_user_second_deposit


# Преобразыбаем словарь в датафрейм
df_funnel_second_deposit = pd.DataFrame(
    # Преобразуем словарь в список пар ключ+значение
    list(dict_count_user_second_deposit.items()),
    # Задаем назбания полей
    columns=['risk_level', 'count_user_second_deposit']
)

# Выводим получившийся датафрейм
df_funnel_second_deposit
```

Out[32]:

	risk_level	count_user_second_deposit
0	low	1080
1	medium	2288
2	high	2040

In [33]:
```
# Объединяем датафреймы
df_funnel_second_deposit = df_funnel_risk_level.merge(df_funnel_second_deposit, on='risk_level')
# Добавляем поле с конверсией второй депозит
df_funnel_second_deposit['conversion_%'] =\
    100 * df_funnel_second_deposit['count_user_second_deposit']  / df_funnel_second_deposit['count_user_asset_purchase']

# Выводим итоговый датафрейм
df_funnel_second_deposit
```

Out[33]:

	risk_level	count_user_asset_purchase	count_user_second_deposit	conversion_%
0	low	2327	1080	46.411689
1	medium	5325	2288	42.967136
2	high	7740	2040	26.356589

Вывод:
Чем выше уровень риска первого актива, тем ниже конверсия во второй депозит.

Наибольшая часть пользователей выбирает для первого актива финансовые инструменты с высоким уровнем риска. При этом лишь 26% из них решаются на повторный депозит. Почему?

Основные гипотезы низкой конверсии во второй депозит у рискованных инвесторов:

1. Рискованные активы подвержены резким колебаниям. Пользователь мог потерять существенную часть депозита, разочароваться и уйти из приложения.
2. Инвестор ждет идеального момента для входа, например, сильного падения перед покупкой, или наблюдает за динамикой, прежде чем решить куда вкладывать. Из-за чего цикл пополнения растягивается.

Анализ ключевой метрики на исторических данных

Динамика средней суммы всех депозитов (объем инвестирования) на одного пользователя в зависимости от дня регистрации пользователей в приложении

In [34]:
```
# Подготовим данные

# Получаем среднее значение 'amount' по всем данным
mean_total_amount = df.groupby('user_id')['amount'].sum().mean()

# Для каждого дня привлечения и пользователя получаем суммарное значение 'amount'
df_amount = df.groupby(['first_dt', 'user_id'])['amount'].sum().reset_index()
```

```
#  Для каждого дня привлечения получаем среднее значение 'amount'  и переименуем поле в 'mean_amount'
df_mean_amount = df_amount.groupby('first_dt')['amount'].mean().reset_index(name='mean_amount')
```

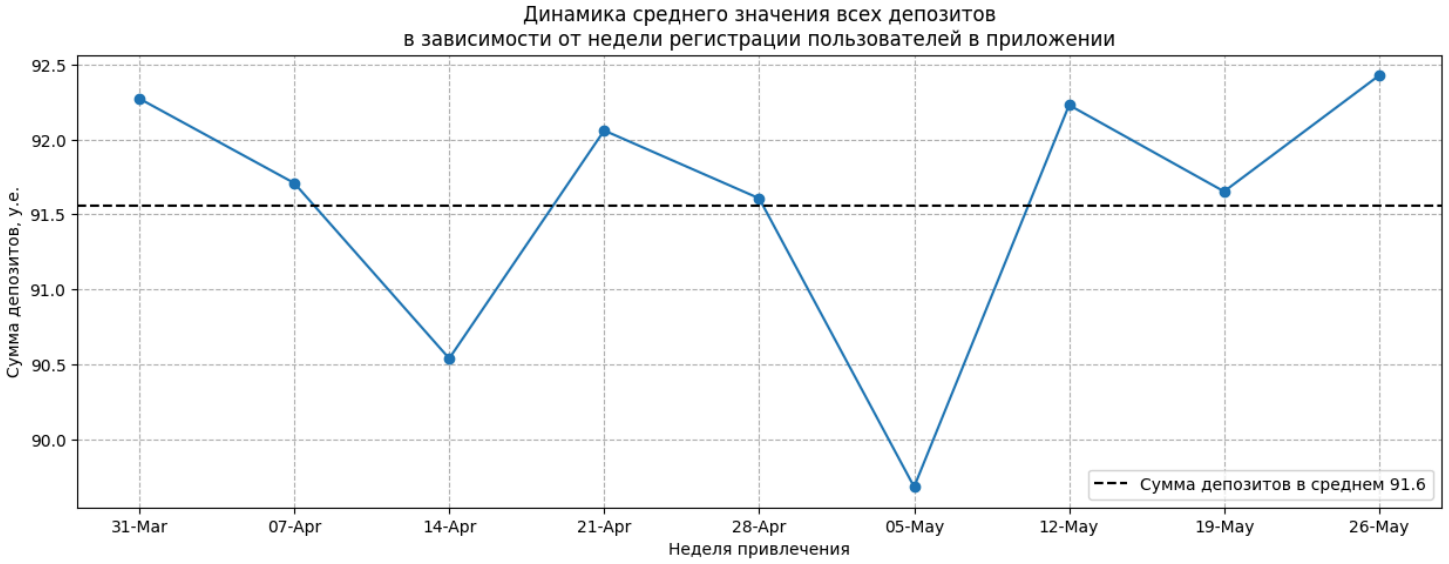
In [35]:
```
# Задаем размер графика
plt.figure(figsize=(15, 5))

# Строим линейный график
plt.plot(df_mean_amount['first_dt'], df_mean_amount['mean_amount'], marker='o')

# Строим линейный график скользящей средней
plt.plot(df_mean_amount['first_dt'],
         df_mean_amount['mean_amount'].rolling(7).mean(),
         color='tab:orange',
         label='Скользящая средняя (окно 7 дней)')

# Добавляем горизонтальную линию – среднее значение по всем данным
plt.axhline(y=mean_total_amount, color='black', linestyle='--', label=f'Сумма депозитов в среднем {mean_total_amount:.1f}')

# Вертикальные линии для обозначения выходных дней:
# Получаем список выходных дней
weekends = [date for date in df_mean_amount['first_dt'] if date.weekday() >= 5]
# Строим одну линию (для первой даты) с меткой
# Находим индекс даты и соответствующее значение Y
idx = np.where(df_mean_amount['first_dt'] == weekends[0])[0][0]
y_value = df_mean_amount['mean_amount'][idx]
plt.vlines(x=weekends[0], ymin=df_mean_amount['mean_amount'].min()-1, ymax=y_value,
           colors='#1f77b4', linestyles=':', alpha=0.7, label='Выходные дни')
#plt.axvline(weekends[0], color='blue', linestyle=':', alpha=0.7, label='Выходные дни')
# Строим вертикальные линии (для выходных дней) без меток
for weekend in weekends[1:]:
    # Находим индекс даты и соответствующее значение Y
    idx = np.where(df_mean_amount['first_dt'] == weekend)[0][0]
    y_value = df_mean_amount['mean_amount'][idx]
    # Линии до основного графика
    plt.vlines(x=weekend, ymin=df_mean_amount['mean_amount'].min()-1, ymax=y_value,
               colors='#1f77b4', linestyles=':', alpha=0.7)
    #
    #plt.axvline(weekend, color='blue', linestyle=':', alpha=0.7)

# Устанавливаем ограничения по оси Y
plt.ylim(df_mean_amount['mean_amount'].min()-1, df_mean_amount['mean_amount'].max()+1)

# Добавляем заголовок
plt.title('Динамика среднего значения всех депозитов\nв зависимости от дня регистрации пользователей в приложении')

# Добавлем подписи осей
plt.xlabel('Дата привлечения')
plt.ylabel('Сумма депозитов, у.е.')

# Отображаем дополнительные линии
plt.grid(linestyle='--', alpha=0.3)

# Добавляем легенду
plt.legend(loc=2)

# Создаем список меток
labels = df_mean_amount['first_dt'].sort_values().dt.strftime('%d-%b')
# Передаем позиции,  метки и угол поворота
plt.xticks(df_mean_amount['first_dt'], labels, rotation=90)

# Отображаем график
plt.show()
```

Наблюдается высокая волатильность значений, причинами которой могут быть:

- малое количество новых пользователей в отдельные дни привлечения
- рекламные каналы привлекли пользователей различной платежеспособности
- внешние факторы например, новости рынка ценных бумаг
- несмотря на то, что некоторые высокие значения приходятся на выходные дни, четкой зависимости от дней недели не наблюдается.

Динамика средней суммы всех депозитов (объем инвестирования) на одного пользователя в зависимости от дня регистрации пользователей в приложении (недельные когорты)

In [36]:
```
# Подготовим данные:

# Получаем среднее значение 'amount' по всем данным
mean_total_amount = df.groupby('user_id')['amount'].sum().mean()

# Создаем поле в первой неделей активности пользователей
df['first_week'] = df['first_dt'].dt.to_period('W').dt.start_time

# Для каждой недели привлечения и пользователя получаем суммарное значение 'amount'
df_amount = df.groupby(['first_week', 'user_id'])['amount'].sum().reset_index()

#  Для каждой недели привлечения получаем среднее значение 'amount'  и переименуем поле в 'mean_amount'
df_mean_amount = df_amount.groupby('first_week')['amount'].mean().reset_index(name='mean_amount')


# Визуализация:

# Задаем размер графика
plt.figure(figsize=(15, 5))

# Строим линейный график
plt.plot(df_mean_amount['first_week'], df_mean_amount['mean_amount'], marker='o')

# Добавляем горизонтальную линию – среднее значение по всем данным
plt.axhline(y=mean_total_amount, color='black', linestyle='--',
            label=f'Сумма депозитов в среднем {mean_total_amount:.1f}')

# Добавляем заголовок
plt.title(
    'Динамика среднего значения всех депозитов\nв зависимости от недели регистрации пользователей в приложении'
)

# Добавлем подписи осей
plt.xlabel('Неделя привлечения')
plt.ylabel('Сумма депозитов, у.е.')

# Отображаем дополнительные линии
plt.grid(linestyle='--')

# Добавляем легенду
plt.legend(loc=4)

# Создаем список меток
labels = df_mean_amount['first_week'].sort_values().dt.strftime('%d-%b')
# Передаем позиции,  метки и угол поворота
plt.xticks(df_mean_amount['first_week'], labels)

# Отображаем график
plt.show()
```



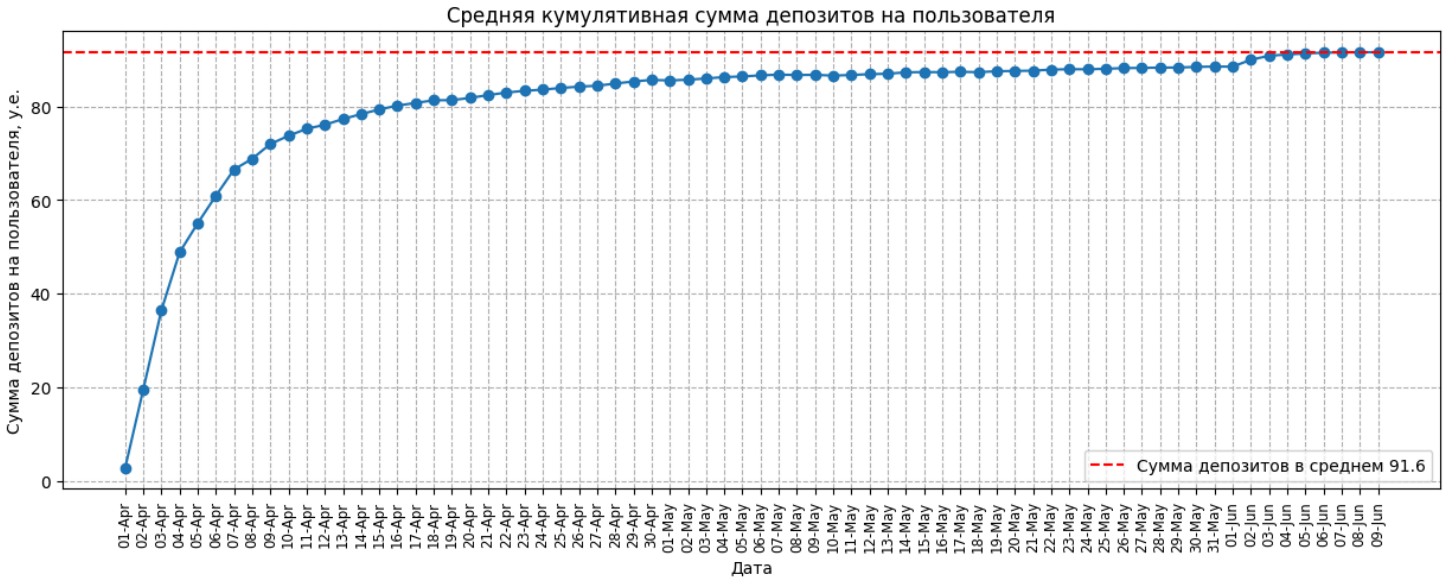
Кумулятивный график средней суммы всех депозитов (объем инвестирования) на одного пользователя

In [37]:
```
# Создаем поле с датой события (без времени)
df['event_dt'] =  pd.to_datetime(df['event_ts'].dt.date)
```

In [38]:
```
# Подготовим данные

# Создаем список уникальных отсортированных дат
list_event_date = np.sort(df['event_dt'].unique())

# Пустой словарь для результатов
records = {}

# Расчет кумулятивной метрики
for current_date in list_event_date:

    # Фильтруем все события, оставляем только события до текущей даты
    active_to_date = df[df['event_dt'] <= current_date]

    # Получаем среднее значение 'amount' на пользователя на текущий день
    group_avg = active_to_date.groupby(['user_id'])['amount'].sum().mean()

    # Добавляем данныk в словарь
    records[current_date] = group_avg
```

```
# Преобразыбаем словарь в датафрейм
df_mean_deposit = pd.DataFrame(
    # Преобразуем словарь в список пар ключ+значение
    list(records.items()),
    # Задаем названия полей
    columns=['date', 'mean_deposit']
)
```

In [39]:
```
# Построим график
plt.figure(figsize=(15, 5))

# Строим линейный график
plt.plot(df_mean_deposit['date'], df_mean_deposit['mean_deposit'], marker='o')

# Добавляем горизонтальную линию — среднее значение по всем данным
plt.axhline(y=mean_total_amount, color='red', linestyle='--',
            label=f'Сумма депозитов в среднем {mean_total_amount:.1f}')

# Добавляем название
plt.title('Средняя кумулятивная сумма депозитов на пользователя')

# Добавляем подписи осей
plt.xlabel('Дата')
plt.ylabel('Сумма депозитов на пользователя, у.е.')

# Отображаем дополнительные линии
plt.grid(linestyle='--')

# Создаем список меток
labels = df_mean_deposit['date'].sort_values().dt.strftime('%d-%b')
# Передаем позиции,  метки и угол поворота
plt.xticks(df_mean_deposit['date'], labels, rotation=90, fontsize='small')

# Добавляем легенду
plt.legend(loc=4)

plt.show()

# Дополнительная информация
print('Средняя сумма депозитов: ', df.groupby('user_id')['amount'].sum().mean().round(2))
print('Стандартное отклонение: ', round(np.std(df.groupby('user_id')['amount'].sum()), 2))
```




```
Средняя сумма депозитов:  91.56
Стандартное отклонение:  80.39
```

Вывод:

График демонстрирует быстрый рост в первую неделю с последующей стагнацией. Возможно сказывается новизна сервиса, возможно бонусы за первые инвестиции (например, "Внеси сумму на депозит и получить акции в подарок"), возможно воодушевляющий онбординг обещающий быстрый доход. При этом эффект эйфории быстро проходит. Возможно заканчиваются "свободные"средства, возможно пользователи психологически не готовы трать больше 90 у.е. на инвестиции, возможно им не хватает знаний в области для дальнейшего использования сервиса или их отпугнули комиссии, налоги с прибыли и т.д.

Средняя сумма депозита 91.56 у.е. и очень высокий показатель разброса 80.39 говорят о нестабильности данных - часть депозитов намного больше среднего, а часть намного меньше.

Часть 2

Исследование результатов A/B эксперимента

Загрузка данных

In [40]:
```
# Загружаем данные с результатами эксперимента из CSV-файла
url='https://drive.google.com/uc?export=download&id=1q64LprPDDo2myfTfStu6-KZArZfPstNZ'
df_abt = pd.read_csv(url, parse_dates=['first_dt', 'event_ts', 'first_ts'])
```

In [41]:
```
# Получаем основную информацию
df_abt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54805 entries, 0 to 54804
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   user_id       54805 non-null  object
 1   country_code  54805 non-null  object
 2   platform      54805 non-null  object
 3   first_ts      54805 non-null  datetime64[ns]
 4   first_dt      54805 non-null  datetime64[ns]
 5   event_ts      54805 non-null  datetime64[ns]
 6   event_name    54805 non-null  object
 7   ab_test       54805 non-null  object
 8   group         54805 non-null  object
 9   amount        7843 non-null   float64
 10  asset         3750 non-null   object
 11  risk_level    3750 non-null   object
dtypes: datetime64[ns](3), float64(1), object(8)
memory usage: 5.0+ MB
```

In [42]:
```
# Выводим статистики для всех полей
df_abt.describe(include='all')
```

Out[42]:

	user_id	country_code	platform	first_ts	first_dt	event_ts	event_name	ab_test	group	amount	asset	risk_level
count	54805	54805	54805	54805	54805	54805	54805	54805	54805	7843.000000	3750	3750
unique	9415	4	2	NaN	NaN	NaN	8	1	2	NaN	3	3
top	a9907642-62cf-4788-960f-b1da223a8bdb	BR	mobile	NaN	NaN	NaN	install / open_web	onboarding_test	control	NaN	stock	high
freq	8	17369	38640	NaN	NaN	NaN	9415	54805	28085	NaN	1297	1630
mean	NaN	NaN	NaN	2025-06-09 02:35:07.728692992	2025-06-08 11:12:11.560988928	2025-06-09 15:31:20.440452608	NaN	NaN	NaN	109.620681	NaN	NaN
min	NaN	NaN	NaN	2025-06-02 00:27:50	2025-06-02 00:00:00	2025-06-02 00:27:50	NaN	NaN	NaN	25.000000	NaN	NaN
25%	NaN	NaN	NaN	2025-06-05 14:35:08	2025-06-05 00:00:00	2025-06-06 00:20:38	NaN	NaN	NaN	62.000000	NaN	NaN
50%	NaN	NaN	NaN	2025-06-08 22:27:05	2025-06-08 00:00:00	2025-06-09 15:32:06	NaN	NaN	NaN	97.000000	NaN	NaN
75%	NaN	NaN	NaN	2025-06-12 16:04:27	2025-06-12 00:00:00	2025-06-13 04:03:54	NaN	NaN	NaN	130.500000	NaN	NaN
max	NaN	NaN	NaN	2025-06-15 23:58:39	2025-06-15 00:00:00	2025-06-22 13:37:05	NaN	NaN	NaN	1050.000000	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.604702	NaN	NaN

In [43]:
```
# Получаем данные по типу платформы для каждого пользователя
df_user_and_platform_abt = df_abt[['user_id', 'platform']].drop_duplicates()

# Доли платформ
df_user_and_platform_abt['platform'].value_counts(normalize=True)
```

Out[43]:
```
platform
mobile    0.704302
web       0.295698
Name: proportion, dtype: float64
```

In [44]:
```
# Доли типов активов
df_abt['asset'].value_counts(normalize=True)
```

Out[44]:
```
asset
stock     0.345867
crypto    0.329333
option    0.324800
Name: proportion, dtype: float64
```

In [45]:
```
# Доли активов по степени рисков
df_abt['risk_level'].value_counts(normalize=True)
```

Out[45]:
```
risk_level
high      0.434667
medium    0.357067
low       0.208267
Name: proportion, dtype: float64
```

In [46]:
```
# Выводим названия всех возможных событий
df_abt['event_name'].unique()
```

Out[46]:
```
array(['install / open_web', 'introduction', 'registration', 'main_page',
       'onboarding_complete', 'first_deposit', 'asset_purchase',
       'second_deposit'], dtype=object)
```

In [47]:
```
# Выводим названия всех стран
df_abt['country_code'].unique()
```

Out[47]:
```
array(['BR', 'MX', 'CO', 'AR'], dtype=object)
```

In [48]:
```
# Выводим названия всех возможных тестов
df_abt['ab_test'].unique()
```

Out[48]:
```
array(['onboarding_test'], dtype=object)
```

Дубликаты в данных

Убедимся, что в данных нет явных дубликатов и что в одну дату и временя не произошли различные события:

In [49]:
```
# Выводим количество полных дубликатов
print("Количество полных дубликатов:", df_abt.duplicated().sum())
```

```
Количество полных дубликатов: 0
```

In [50]:
```
# создадим список столбцов
list_name_columns = df_abt.columns

# Найдем количество дубликатов по всем столбцам, кроме 6-х последних
# То есть проверим, что в данных нет пользователей с разными событиями в одно и тоже время
print("Количество неявных дубликатов:", df_abt.duplicated(subset=list_name_columns[:-6], keep='first').sum())
```

```
Количество неявных дубликатов: 0
```

Корректность данных

Проверим, что для каждого пользователя есть только одна отметка о принадлежности к определенной стране , каждый пользователь использует только 1 платформу и нет пользователей отнесенных в обе группы:

In [51]:
```
# Группируем данные по пользователям и для каждого находим число уникальных кодов стран,
# оставляем тех пользователей, которые имеют больше 1 страны и считаем их количество
print("Количество пользователей с более чем одной страной:")
sum(df_abt.groupby('user_id')['country_code'].nunique() > 1)
```

```
Количество пользователей с более чем одной страной:
```

Out[51]:
```
0
```

In [52]:
```
# Группируем данные по пользователям и для каждого находим число уникальных платформ,
# оставляем тех пользователей, которые имеют больше 1 платформы
print("Количество пользователей с более чем одной платформой:")
sum(df_abt.groupby('user_id')['platform'].nunique() > 1)
```

```
Количество пользователей с более чем одной платформой:
```

Out[52]:
```
0
```

In [53]:
```
# Группируем данные по пользователям и для каждого находим число уникальных тестовых групп,
# оставляем тех пользователей, которые имеют больше 1 группы
print("Количество пользователей, попавших в обе тестовые группы:")
sum(df_abt.groupby('user_id')['group'].nunique() > 1)
```

```
Количество пользователей, попавших в обе тестовые группы:
```

Out[53]:
```
0
```

Промежуточные выводы:

1. Названия столбцов отражают их содержимое, они понятны и удобны для работы.
2. Данные содержат 54805 записей
3. Пропуски. Три столбца `amount` , `asset` и `risk_level` содержат пропуски, которые обусловлены спецификой данных.
4. Типы данных корректны.
5. Типы платформ: `mobile` и `web` . Большая часть пользователей использует для взаимодействия мобильную версию приложения (70%).
6. Временной промежуток привлечения пользователей: 02.06.2025 - 15.06.2025 - две полные недели с понедельника по воскресенье. При этом 50% пользователей зарегистрировались в 1 неделю, 50% - во вторую.
7. Временной промежуток событий пользователей чуть шире: 02.06.2025 - 22.06.2025
8. Сумма пополнения депозита: от 25 у.е. до 1050 у.е. Верхняя граница выше в 2 раза по сравнению с предэкспериментальным периодом. Среднее значение (110 у.е.) несколько выше медианного (97 у.е.), что говорит о наличии в данных высоких значений.
9. Типы приобретенных активов (`stock` - акции, `option` - опционы, `crypto` - криптовалюты) распределены примерно одинаково. Акции самые популярные (34.6%). В отличие от предпериода, где лидировали опционы. Опционы наименее популярны (32.5%). В отличие от предпериода, где последнее место занимала криптовалюта.
10. Уровень риска активов. Наиболее полулярны среди пользователей активы с высоким уровнем риска `high` , их доля больше 43,5% (что 7% ниже предпериода). На активы с низким уровнем риска `low` приходится 20,9% (что выше предпериода примерна на 6%). Активы среднего уровня риска 35,7%(что на 1% выше предпериода)
11. Дубликаты в данных не выявлены.
12. Данные корректны. Каждый пользователь "привязан" к конкретной стране и типу платформы и тестовой группе.

Анализ аудитории эксперимента.

Проверим корректность распределения новых пользователей по группам A/B-эксперимента

In [54]:
```
# Убедимся, что среди пользователей нет "старичков"
# Найдем общие идентификаторы пользователей среди двух датасетов и выведем их количество
len(set(df['user_id']) & set(df_abt['user_id']))
```

Out[54]:
```
0
```

Проверка распределения пользователей

Проверим, соответствует ли фактическое распределение пользователей между группами заявленному 50/50:

In [55]:
```
# Рассчитаем количество уникальных пользователей в каждой из групп:

# Группируем и агрегируем данные, переименовываем поле
df_user_test = df_abt.groupby(['group']).agg({'user_id':'nunique'})

# Переименовываем столбец на более подходящее имя
df_user_test.columns = ['user_count_uniq']

# Выводим результат
df_user_test
```

Out[55]:

group	user_count_uniq
control	4847
test	4568

In [56]:
```
# Расчитываем процентную разницу между группами:

# Подготовим данные
```

```
count_user_a = df_user_test.loc['control', 'user_count_uniq']
count_user_b = df_user_test.loc['test', 'user_count_uniq']

# Производим расчет
percentage_difference = 100 * fabs(count_user_a - count_user_b) / count_user_a

# Выводим результат
print(f'Процентная разница в количестве пользователей в группах control и test: {round(percentage_difference,3)}%')
```

```
Процентная разница в количестве пользователей в группах control и test: 5.756%
```

Между группами существует разница 279 пользователей. Проверим, является ли дисбаланс между группами (4 847 и 4 568) случайностью или это ошибка в распределении. Обозначим n_a и n_b - количество пользователей в группах control и test соответственно, а total_observed - общее количество. Тогда гипотезы будут выглядеть так:

- H0: n_a = n_b , то есть разницы между control-группой и test-группой нет
- H1: n_a != n_b , то есть разницы между control-группой и test-группой есть. Воспользуемся хи-квадрат тестом:

In [57]:
```
# уровень значимости
alpha = 0.05

# Фактическое распределение
n_a = df_user_test.loc['control', 'user_count_uniq']
n_b = df_user_test.loc['test', 'user_count_uniq']
observed = [n_a, n_b]

# Планируемое распределение
total_observed = df_user_test['user_count_uniq'].sum()
expected = [total_observed / 2, total_observed / 2]

chi2, p_value_chi2 = chisquare(observed, f_exp=expected)

if p_value_chi2 > alpha:
    print(f'p-value={p_value_chi2:.4f} > {alpha}')
    print('Нулевая гипотеза находит подтверждение - разница между группами незначима! Дисбаланс - это случайность')
else:
    print(f'p-value={p_value_chi2:.4f} < {alpha}')
    print('⚠Нулевая гипотеза не находит подтверждения - разница между группами значима!')
```

```
p-value=0.0040 < 0.05
⚠Нулевая гипотеза не находит подтверждения - разница между группами значима!
```

In [58]:
```
# Задаем область и оси
fig, ax = plt.subplots(figsize=(8, 4))

# Строим столбчатую диаграмму
ax.bar(df_user_test.index, df_user_test['user_count_uniq'])

# Задаем название и подписываем оси
ax.set_title(f'Распределение пользователей по группам.\nПроцентная разница между группами составляет {round(percentage_difference,3)}% \n')
ax.set_xlabel('Группы')
ax.set_ylabel('Количество пользователей')

# Убираем границы
ax.spines[['top', 'right']].set_visible(False)

# Подписываем бары
plt.bar_label(ax.containers[0])

# Отображаем график
plt.show()
```

Проверка пересечений пользователей

Проверим, что группы control и test - независимы. Для этого убедимся, что никто из пользователей случайно не попал в обе группы одновременно. Используем пересечение множеств для нахождения общих элементов:

In [59]:
```
# Найдем множество пользователей для каждой группы
set_a = set(df_abt['user_id'][df_abt['group'] == 'control'])
set_b = set(df_abt['user_id'][df_abt['group'] == 'test'])

# Найдем пересечение  множеств
set_a.intersection(set_b)
```

Out[59]:
```
set()
```

Пересечение множеств пользователей пусто, то есть нет ни одного пользователи, попавшего одновременно в обе группы.

Проверка равномерности разделения пользователей по типам платформ

In [60]:
```
# Группируем и агрегируем данные
df_group_platform = df_abt.groupby(['platform', 'group'])['user_id'].nunique().unstack(fill_value=0)

# Найдем доли, разделив на общее количество пользователей, и преобразуем в проценты, результат округлим
df_group_platform = (100 * df_group_platform / df_abt['user_id'].nunique()).round(2)


# Производим расчет абсолютной разницы между группами
df_group_platform['absolute _diff'] = df_group_platform['test']- df_group_platform['control']

# Производим расчет относительной разницы между группами
df_group_platform['relative_diff'] = 100 * abs(df_group_platform['absolute _diff']) / df_group_platform['control']


# Выводим результат
df_group_platform
```

Out[60]:

group	control	test	absolute _diff	relative_diff
platform				
mobile	35.95	34.48	-1.47	4.089013
web	15.53	14.04	-1.49	9.594334

Проверим, что пользователи равномерно распределены по всем доступным типам платформ. Тогда гипотезы будут выглядеть так:

- H0: распределение платформ одинаково в control и test-группах
- H1: распределение платформ различается между control и test-группами.

Воспользуемся хи-квадрат тестом на независимость:

In [61]:
```
# Проверим независимость двух категориальных переменных:

# Уровень значимости
alpha = 0.05

# Создаем таблицу сопряженности: группы * платформы
contingency_table = df_group_platform[['control', 'test']].T

# Хи-квадрат тест на независимость
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Выводим результат
print('\nПроверка по типам платформ:')
if p_value > alpha:
    print(f'p-value={p_value:.4f} > {alpha}')
    print('Распределение между группами корректно!')
else:
    print(f'p-value={p_value:.4f} < {alpha}')
    print('⚠Есть статистически значимый дисбаланс!')
```

```
Проверка по типам платформ:
p-value=1.0000 > 0.05
Распределение между группами корректно!
```

Абсолютная разница между группами не превышает полутора процентов. Пользователи корректно распределены по группам и платформам. Значимых перекосов не наблюдается. Группы сопоставимы по типам платформ.

Проверка равномерности разделения пользователей по странам

In [62]:
```
# Группируем и агрегируем данные
df_group_country = df_abt.groupby(['country_code', 'group'])['user_id'].nunique().unstack(fill_value=0)

# Найдем доли, разделив на общее количество пользователей, и преобразуем в проценты, результат округлим
df_group_country = (100 * df_group_country / df_abt['user_id'].nunique()).round(2)


# Производим расчет абсолютной разницы между группами
df_group_country['absolute _diff'] = df_group_country['test']- df_group_country['control']

# Производим расчет относительной разницы между группами
df_group_country['relative_diff'] = 100 * abs(df_group_country['absolute _diff']) / df_group_country['control']


# Выводим результат
df_group_country
```

Out[62]:

group	control	test	absolute _diff	relative_diff
country_code				
AR	8.03	7.70	-0.33	4.109589
BR	16.39	15.38	-1.01	6.162294
CO	13.24	12.73	-0.51	3.851964
MX	13.82	12.70	-1.12	8.104197

Проверим, что пользователи равномерно распределены по всем странам. Тогда гипотезы будут выглядеть так:

- H0: распределение стран одинаково в control и test-группах
- H1: распределение стран различается между control и test-группами.

Воспользуемся хи-квадрат тестом на независимость:

In [63]:
```
# Проверим независимость двух категориальных переменных:
```

```
# Уровень значимости
alpha = 0.05

# Создаем таблицу сопряженности: группы * страны
contingency_table = df_group_country[['control', 'test']].T

# Хи-квадрат тест на независимость
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Выводим результат
print('\nПроверка по странам:')
if p_value > alpha:
    print(f'p-value={p_value:.4f} > {alpha}')
    print('Распределение между группами корректно!')
else:
    print(f'p-value={p_value:.4f} < {alpha}')
    print('⚠️Есть статистически значимый дисбаланс!')
```

```
Проверка по странам:
p-value=0.9998 > 0.05
Распределение между группами корректно!
```

Абсолютная разница между группами не превышает 1,1% процент. Пользователи корректно распределены по группам и странам. Значимых перекосов не наблюдается. Группы сопоставимы по категориям стран.

Регистрация пользоваталей в приложении

In [64]:
```
# Выведем статистику по группе и убедимся, что регистрация пользователей была равномерна на протяжении эксперимента
df_abt[df_abt['group'] == 'control'].describe()
```

Out[64]:

	first_ts	first_dt	event_ts	amount
count	28085	28085	28085	3882.000000
mean	2025-06-09 03:32:10.775681024	2025-06-08 12:10:41.424247808	2025-06-09 16:17:22.404949248	112.672334
min	2025-06-02 00:27:50	2025-06-02 00:00:00	2025-06-02 00:27:50	25.000000
25%	2025-06-05 15:12:16	2025-06-05 00:00:00	2025-06-06 00:53:59	89.000000
50%	2025-06-08 22:44:14	2025-06-08 00:00:00	2025-06-09 16:02:56	108.000000
75%	2025-06-12 16:56:47	2025-06-12 00:00:00	2025-06-13 06:12:57	131.000000
max	2025-06-15 23:47:04	2025-06-15 00:00:00	2025-06-22 13:37:05	322.000000
std	NaN	NaN	NaN	33.680537

In [65]:
```
# Выведем статистику по группе и убедимся, что регистрация пользователей была равномерна на протяжении эксперимента
df_abt[df_abt['group'] == 'test'].describe()
```

Out[65]:

	first_ts	first_dt	event_ts	amount
count	26720	26720	26720	3961.000000
mean	2025-06-09 01:35:09.814221568	2025-06-08 10:10:42.395209472	2025-06-09 14:42:57.380089856	106.629891
min	2025-06-02 01:08:14	2025-06-02 00:00:00	2025-06-02 01:08:14	25.000000
25%	2025-06-05 13:45:58.500000	2025-06-05 00:00:00	2025-06-05 23:37:47	35.000000
50%	2025-06-08 22:19:09	2025-06-08 00:00:00	2025-06-09 15:17:58.500000	67.000000
75%	2025-06-12 15:14:05	2025-06-12 00:00:00	2025-06-13 02:03:24.500000	130.000000
max	2025-06-15 23:58:39	2025-06-15 00:00:00	2025-06-21 12:09:56	1050.000000
std	NaN	NaN	NaN	117.146019

Промежуточные выводы

Результаты анализа аудитории эксперимента:

1. Независимость выборок:
   - пересечение между тестовой и контрольной группами отсутствует - ни один пользователь не был зафиксирован одновременно в обеих группах.
2. Корректность распределения:
   - между группами существует статистически значимая разница. Контрольная группа на 279 пользователей больше (процентная разница 5,8%)
3. Равномерность распределения:
   - пользователи равномерно распределены по типам платформ ;
   - пользователи равномерно распределены по группам стран

Дисбаланс в распределении пользователей на группы может существенно повлиять на ключевую метрику, если в одной из групп случайно окажется больше пользователей склонных к более существенным вложениям. Этот факт следует учесть при интерпретации результатов.

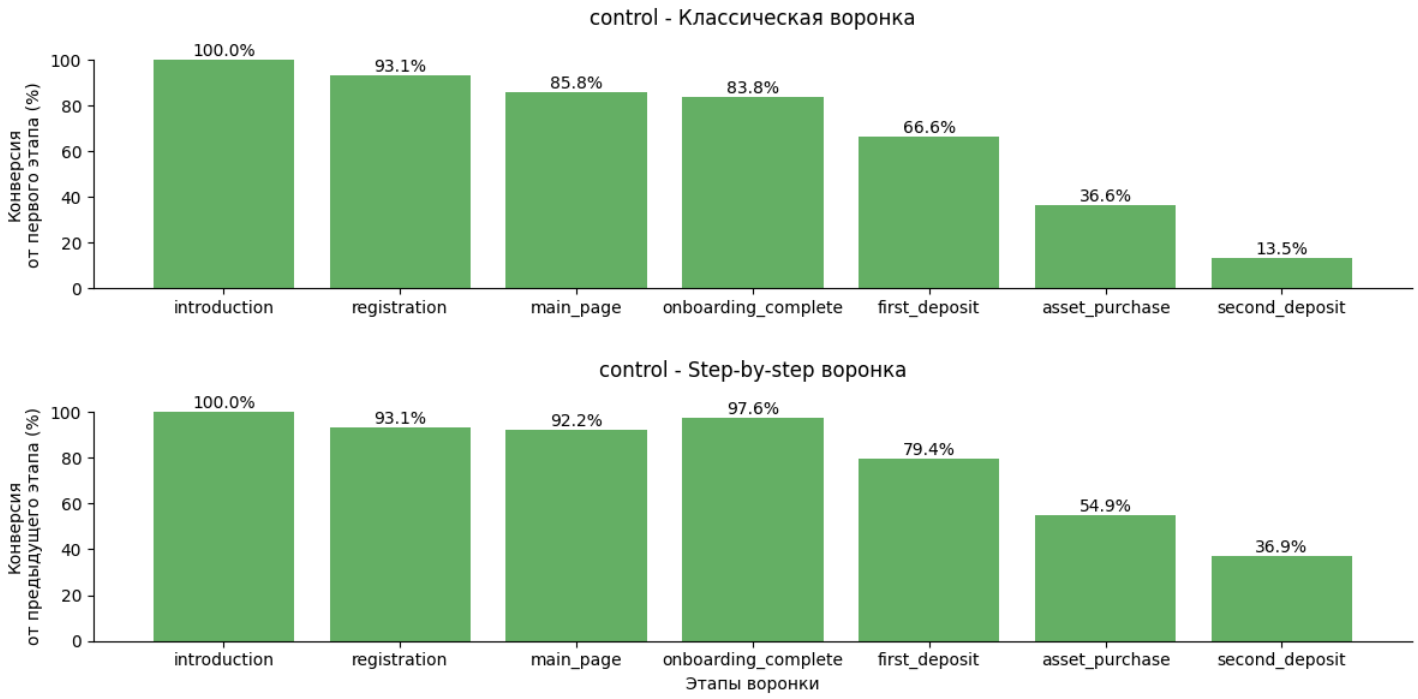
Сравнение воронок событий.

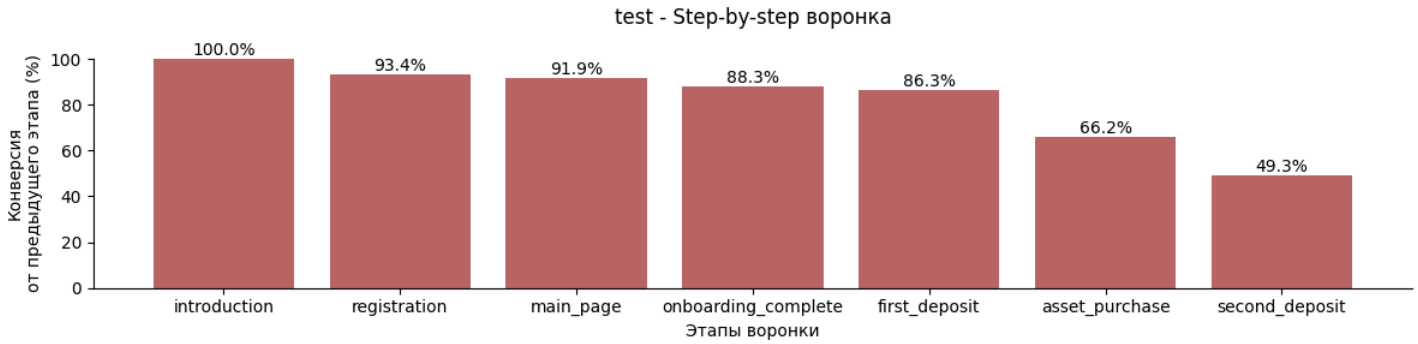
Сравним пользовательский путь новых пользователей в тестовой и контрольной группах.

Анализ пользовательского пути в целом

In [66]:
```
# создаем словарь с названиями тестовых групп и соответствующим цветом для визуализации
list_name_group = {'control':'green', 'test':'darkred'}

for name_group, color_group in list_name_group.items():

    # Считаем число уникальных пользователей на каждом шаге
    # Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
```

```
df_test = df_abt[df_abt['group'] == name_group].groupby('event_name')['user_id'].nunique().reindex(list_event).reset_index()
# Переименовываем поля
df_test = df_test.rename(columns = {'event_name':'step', 'user_id':'users'})

# Добавляем поле с конверсией для классической воронки и округляем до 1 знака после запятой
# Каждое значение делим на первое
df_test['conversion_from_first_%'] = df_test['users'].div(df_test.loc[0]['users']).round(3) * 100

# Добавляем поле с конверсией для step-by-step воронки и округляем до 1 знака после запятой
# Каждое значение делим на предыдущее
df_test['conversion_from_previous_%'] = df_test['users'].div(df_test['users'].shift(1)).round(3) * 100


# Строим два графика
fig, axes = plt.subplots(2, 1, figsize=(12, 6))

# Классическая воронка:

# Опускаем первый этап 'install / open_web'
axes[0].bar(df_test.loc[1:]['step'], df_test.loc[1:]['conversion_from_first_%'], color=color_group, alpha=0.6)
axes[0].set_title(f'{name_group} - Классическая воронка\n')
#axes[0].set_xlabel('Этапы воронки')
axes[0].set_ylabel('Конверсия\nот первого этапа (%)')
# Добавляем подписи барам
axes[0].bar_label(axes[0].containers[0], fmt='%.1f%%', fontsize=10)
# Убираем границы
axes[0].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[0].set_ylim(0, 100)


# step-by-step воронка:

# Опускаем первый этап 'install / open_web'
axes[1].bar(df_test.loc[1:]['step'], df_test.loc[1:]['conversion_from_previous_%'], color=color_group, alpha=0.6)
axes[1].set_title(f'\n{name_group} - Step-by-step воронка\n')
axes[1].set_xlabel('Этапы воронки')
axes[1].set_ylabel('Конверсия\nот предыдущего этапа (%)')
# Добавляем подписи барам
axes[1].bar_label(axes[1].containers[0], fmt='%.1f%%', fontsize=10)
# Убираем границы
axes[1].spines[['top', 'right']].set_visible(False)
# Установим максимальное значение
axes[1].set_ylim(0, 100)


# Автоматическая настройка отступов между графиками
plt.tight_layout()

plt.show()
```





Вывод:

Классическая воронка

1. Группы демонстрируют разную конверсию от первого шага до второго депозита:
   - для контрольной группы - 13.5% (значение очень близко к уровеню предпериода)
   - для тесторой группы - 21.3%
2. Этап онбординга:
   - для контрольной группы - 83.8% (значение очень близко к уровеню предпериода)
   - для тесторой группы - 75.8%
3. Этап покупки активов:
   - для контрольной группы - 36.6% (значение чуть ниже уровня предпериода 37,5)
   - для тесторой группы - 43.3%

Тестовая группа демонстрирует значительное улучшение конверсии на ключевых этапах воронки по сравнению с контрольной группой. Однако она проигрывает на этапе онбординга.

Step-by-tep воронка

1. Из регистрации в главную страницу:
   - для контрольной группы - 92.2% (значение чуть выше уровеня предпериода 91.9%)
   - для тестовой группы - 91.9%
2. С главной страницы в онбординг:
   - для контрольной группы - 97.6% (значение чуть ниже уровня предпериода 98%)
   - для тестовой группы - 88.3%
3. Из онбординга в первый депозит:
   - для контрольной группы - 79.4% (значение чуть ниже уровня предпериода 80.6%)
   - для тестовой группы - 86.3%
4. Из первого депозита в покупку активов:
   - для контрольной группы - 54.6% (значение чуть ниже уровня предпериода 55.6%)
   - для тестовой группы - 66.2%
5. Из покупки во второй депозит:
   - для контрольной группы - 36.9% (значение чуть выше уровня предпериода 35.1%)
   - для тестовой группы - 49.3%

Наблюдаем небольшое снижение конверсии из регистрации в главную страницу:

1. Уточнить, не повлияло ли изменение в продукте на этот этап. Например, анонс о необходимости пройти обязательное обучение мог отпугнуть некоторых пользователей.
2. Проверить баги, посомтреть по сегментам.

Значительная просадка наблюдается на этапе онбординга:

1. Проверить технические проблемы, например, проблема с кнопкой "Начать онбординг" или долгая загрузка.
2. Возможно часть пользователей начинают онбординг, но не завершают его из-за сложности новой информации. Но с другой стороны это является своего рода фильтром, который оставляет в продукте наиболее замотивированных пользователей. Это подтверждается высокими показателями конверсий на следующих этапах.
3. Для контроля конверсии в завершение онбординга можно рассмотреть варианты бонусов за прохождение обрординга до конца, это позволит простимулировать сомневающихся в своих силах пользователей.

Влияние новой фичи на конверсию во второй депозит с учётом уровня риска купленного актива.

Проанализируем, как категория риска купленного актива влияет на вероятность открытия второго депозита в тестовой и контрольной группах.

Влияние категории риска купленного актива на вероятность открытия второго депозита в тестовой и контрольной группах

In [67]:
```
# Список  событий: покупка активов и повторный депозит
list_event_part = ['asset_purchase', 'second_deposit']
```

```
# Список уровней риска
list_risk_level = ['low', 'medium', 'high']
# Список имен групп
list_name_group = ['control', 'test']
# Пустой словарь для записи результатов кода
dict_df = {}

# Фильтруем датасет. Оставляем только события о покупке активов и повторном депозите
df_risk_level_abt = df_abt[df_abt['event_name'].isin(list_event_part)]

# В цикле для каждой группы сформируем данные о конверсии из шага покупки актива в шаг второго депозита для каждого уровня риска:
for name_group in list_name_group:

    # 1 шаг - для текущей группы и каждого уровня риска считаем количесвто пользователей:

    # Фильтруем исходный датафрейм для текущей тестовой группы
    df_abt_filtered = df_risk_level_abt[df_risk_level_abt['group'] == name_group]

    # Для каждого уровня риска найдем количество пользователей, совершивших покупку актива соответствующего уровня
    df_funnel_risk_level = df_abt_filtered.groupby(['risk_level']).agg(
        count_user_asset_purchase=('user_id', 'nunique')
    ).reindex(list_risk_level).reset_index()


    # 2 шаг - для текущей группы и каждого уровня риска считаем количество пользователей со вторым депозитом:

    # Создаем пустой словарь
    dict_count_user_second_deposit = {}

    # В цикле проходимся по каждому уровню риска
    for risk_level in list_risk_level:

        # Получаем список уникальных идентификаторов для выбранного уровня риска
        # используя датасет отфильтрованный для текущей тестовой группы df_abt_filtered из шага 1
        list_user = df_abt_filtered[df_abt_filtered['risk_level'] == risk_level]['user_id'].unique()

        # Фильтруем датасет по идентификаторам пользователей и по событию 'second_deposit'
        df_risk_level_filtered = df_abt_filtered[
            (df_abt_filtered['user_id'].isin(list_user)) & (df_abt_filtered['event_name'] == 'second_deposit')
        ]

        # Считаем количество пользователей с повторным депозитом для выбранного уровня риска
        count_user_second_deposit = df_risk_level_filtered['user_id'].nunique()

        # Записываем данные в словарь
        dict_count_user_second_deposit[risk_level] =  count_user_second_deposit


    # Преобразываем словарь в датафрейм
    df_funnel_second_deposit = pd.DataFrame(
        # Преобразуем словарь в список пар ключ+значение
        list(dict_count_user_second_deposit.items()),
        # Задаем названия полей
        columns=['risk_level', 'count_user_second_deposit']
    )


    # 3 шаг - для текущей группы объединяем датафреймы, полученные на 1 и 2 шаге, и считаем конверсию:

    # Объединяем датафреймы по уровню риска 'risk_level'
    df_merge = df_funnel_risk_level.merge(df_funnel_second_deposit, on='risk_level')
    # Добавляем поле с конверсией второй депозит
    df_merge['conversion_%'] = 100 * df_merge['count_user_second_deposit']  / df_merge['count_user_asset_purchase']


    # 4 шаг - записываем полученные данные для текущей группы в словарь
    dict_df[name_group] = df_merge
```

In [68]:
```
print("Влияние категории риска купленного актива на открытие второго депозита для контрольной группы:")
dict_df['control']
```

```
Влияние категории риска купленного актива на открытие второго депозита для контрольной группы:
```

Out[68]:

	risk_level	count_user_asset_purchase	count_user_second_deposit	conversion_%
0	low	285	142	49.824561
1	medium	595	265	44.537815
2	high	893	247	27.659574

In [69]:
```
print("Влияние категории риска купленного актива на открытие второго депозита для тестовой группы:")
dict_df['test']
```

```
Влияние категории риска купленного актива на открытие второго депозита для тестовой группы:
```

Out[69]:

	risk_level	count_user_asset_purchase	count_user_second_deposit	conversion_%
0	low	496	285	57.459677
1	medium	744	350	47.043011
2	high	737	339	45.997286

Вывод:

Группы показывают разное поведение пользователей.

- В контрольной группе (как и на предпериоде) чем выше уровень риска первого актива, тем ниже конверсия во второй депозит. Половина пользователей из тех кто купил актив (893 из 1773) выбрали для первого актива финансовые инструменты с высоким уровнем риска. При этом лишь 27.7% из них решаются на повторный депозит.
- Тестовая группа показывает другое пользовательское поведение. Здесь пользователи примерно в равной степени отдали предпочтение активам высокого (37,3% пользователей)и среднего (37,6%) уровней риска. Кроме того, конверсия во второй депозит для пользователей с активами высокой степени риска значительно выросла до 46%

Оценка статистической значимости

Согласно расчетов, конверсии во второй депозит в тестовой группе выше, чем в контрольной. Проверим, являются ли это изменения статистически значимыми.

In [70]:
```
# Подготовим данные:

# Сохраняем в переменные
df_abt_control = dict_df['control']
df_abt_test = dict_df['test']

# Переименовываем поля со значениями конверсий
df_abt_control = df_abt_control.rename(
    columns={'count_user_asset_purchase':'control_count_user',
             'count_user_second_deposit':'control_successful_user',
             'conversion_%':'control_conversion_%'}
)
df_abt_test = df_abt_test.rename(
    columns={'count_user_asset_purchase':'test_count_user',
             'count_user_second_deposit':'test_successful_user',
             'conversion_%':'test_conversion_%'}
)


# Объединяем датафреймы
df_abt_conversion = df_abt_control.merge(df_abt_test, on='risk_level')

# Выводим результат
df_abt_conversion
```

Out[70]:

	risk_level	control_count_user	control_successful_user	control_conversion_%	test_count_user	test_successful_user	test_conversion_%
0	low	285	142	49.824561	496	285	57.459677
1	medium	595	265	44.537815	744	350	47.043011
2	high	893	247	27.659574	737	339	45.997286

Обозначим за {pA,pB} конверсию в группах control и test. Тогда гипотезы будут выглядеть так:

- H0: pA >= pB , то есть конверсия в test-группе не лучше конверсии в control-группе
- H1: pA < pB , то есть конверсия в test-группе лучше конверсии в control-группе

In [71]:
```
# Используем z-тест пропорций

# Фиксируем уровень значимости
alpha = 0.05

for risk_level in df_abt_conversion['risk_level']:

    # 1 шаг - подготовка данных:

    # Фильтруем датафрейм по текущему уровню риска
    df_filtered = df_abt_conversion[df_abt_conversion['risk_level'] == risk_level]

    # Получаем данные
    # С помощью .values конвертируем series в массив NumPy
    # Проценты переводим в доли
    n_a = df_filtered['control_count_user'].values[0]
    m_a = df_filtered['control_successful_user'].values[0]
    p_a = df_filtered['control_conversion_%'].values[0] / 100

    n_b = df_filtered['test_count_user'].values[0]
    m_b = df_filtered['test_successful_user'].values[0]
    p_b = df_filtered['test_conversion_%'].values[0] / 100


    # 2 шаг - проверяем, выполняется ли предпосылка Z-теста пропорций о достаточном размере выборок:

    # Проверяем выполнение предпосылки и выводим результат
    if (p_a*n_a > 10)and((1-p_a)*n_a > 10)and(p_b*n_b > 10)and((1-p_b)*n_b > 10):
        print(f'Предпосылка о достаточном количестве данных для {risk_level} выполняется!')
    else:
        print(f'⚠Предпосылка о достаточном количестве данных для {risk_level} НЕ выполняется!')


    # 3 шаг - рассчитаем размер эффекта - индекс Cohen's h:

    # Считаем размер эффекта
    effect_size = proportion_effectsize(p_a, p_b)

    # Выводим результат
    print(f"Размер эффекта Cohen's h для {risk_level}: {(effect_size):.6f}")


    # 4 шаг - рассчитаем фактическую мощность:

    # Считаем соотношение размеров групп
    ratio = n_b / n_a

    # Считаем фактическую мошность
    power = zt_ind_solve_power(
        effect_size=effect_size,
        nobs1=n_a,
        alpha=alpha,
        ratio=ratio
    )

    print(f"Мощность теста для {risk_level}: {power:.2f}")

    # 5 шаг - применяем z-тест пропорций для проверки гипотез:

    # Считаем статистики
    stat_ztest, p_value_ztest = proportions_ztest([m_a, m_b], [n_a, n_b], alternative='smaller')

    # Проверяем условия и интерпретируем
    if p_value_ztest > alpha/len(risk_level):
```

```
        print(f'pvalue={p_value_ztest} > {alpha/len(list_risk_level)}')
        print(f'Нулевая гипотеза для {risk_level} находит подтверждение! Вывод: нет доказательств улучшения конверсии!')
    else:
        print(f'pvalue={p_value_ztest} < {alpha/len(list_risk_level)}')
        print(f'✅ Нулевая гипотеза для {risk_level} не находит подтверждения! Вывод: конверсия статистически значимо лучше!')

    print("----------------")
```

```
Предпосылка о достаточном количестве данных для low выполняется!
Размер эффекта Cohen's h для low: -0.153261
Мощность теста для low: 0.54
pvalue=0.019536095218929204 > 0.016666666666666666
Нулевая гипотеза для low находит подтверждение! Вывод: нет доказательств улучшения конверсии!
----------------
Предпосылка о достаточном количестве данных для medium выполняется!
Размер эффекта Cohen's h для medium: -0.050288
Мощность теста для medium: 0.15
pvalue=0.18034523696923505 > 0.016666666666666666
Нулевая гипотеза для medium находит подтверждение! Вывод: нет доказательств улучшения конверсии!
----------------
Предпосылка о достаточном количестве данных для high выполняется!
Размер эффекта Cohen's h для high: -0.383055
Мощность теста для high: 1.00
pvalue=8.022633974380208e-15 < 0.016666666666666666
✅ Нулевая гипотеза для high не находит подтверждения! Вывод: конверсия статистически значимо лучше!
----------------
```

Вывод:

1. Уровни риска low и medium:
   - несмотря на то, что расчеты показали улучшение конверсий в тестовой группе по сравнению с контрольной (для активов low-риска с 49.8% до 57.5%, для активов medium-риска с 44.5% до 47.0%), их статистическая значимость не подтверждена.
   - низкая мощность теста и размер эффекта особенно для активов среднего уровня рисков.
   - возможно причиной стало значим разное распределение пользователей по группам 51,5%/48,5% вместо 50/50
2. Уровень риска high:
   - наблюдается статистически значимое улучшение с 27.7% до 46%, абсолютный прирост конверсии 18.3%
   - пользователи выбравшие высокорисковые финансовые инструменты оказались наиболее чувствительны к изменениям онбординга

Анализ метрик A/B-эксперимента.

Анализ ключевой метрики - средней суммы всех депозитов на одного пользователя

Расчет метрик

In [72]:
```
# Подготовим данные

# Создаем пустой словарь
dict_mean_total_amount_abt = {}

# В цикле проходимся по каждой группе
for name_group in list_name_group:

    # Получаем среднее значение 'amount' по всем данным для текущей тестовой группы
    mean_amount = df_abt[df_abt['group'] == name_group].groupby('user_id')['amount'].sum().mean()

    # Записываем данные в словарь
    dict_mean_total_amount_abt[name_group] =  mean_amount


# Преобразываем словарь в датафрейм
df_mean_amount_abt = pd.DataFrame(dict_mean_total_amount_abt, index=[0])

mean_control = df_mean_amount_abt.loc[0,'control']
mean_test = df_mean_amount_abt.loc[0,'test']

# Производим расчет абсолютной разницы между группами
df_mean_amount_abt['absolute _diff'] = mean_test - mean_control

# Производим расчет относительной разницы между группами
df_mean_amount_abt['relative_diff'] = 100 * fabs(mean_test - mean_control) / mean_control
```

In [73]:
```
print('Средняя сумма всех депозитов на одного пользователя:')
df_mean_amount_abt
```

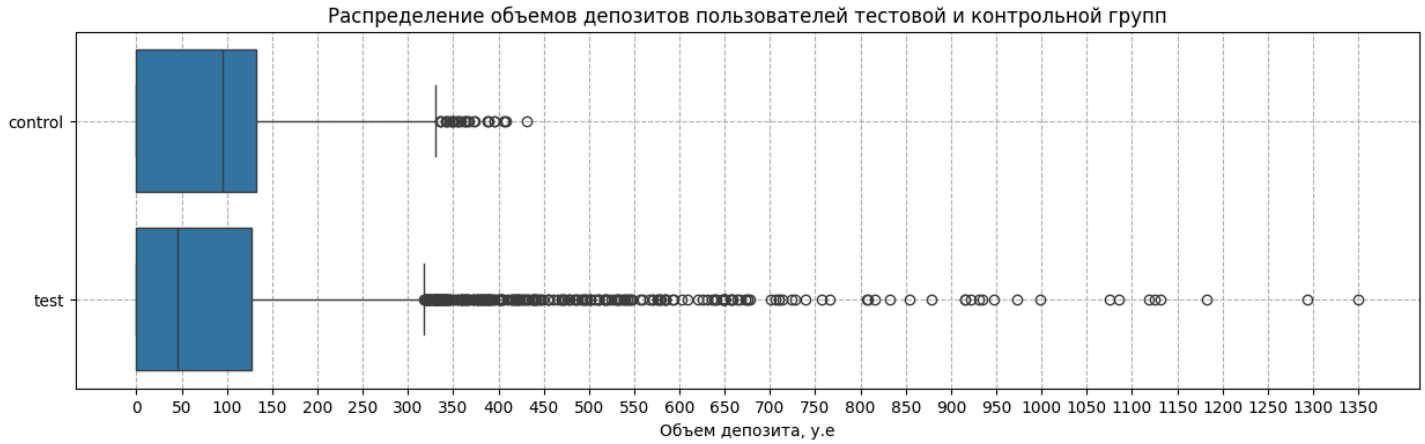
```
Средняя сумма всех депозитов на одного пользователя:
```

Out[73]:

	control	test	absolute _diff	relative_diff
0	90.240149	92.460814	2.220666	2.46084

In [74]:
```
# Посмотрим на распределение депозитов пользователей в каждой тестовой группе:

# Подготовим данные:

# Агрегируем данные по пользователям и тестовым группам
df_abt_user_agg = df_abt.groupby(['user_id', 'group'])['amount'].sum().reset_index()

df_c = df_abt_user_agg[df_abt_user_agg['group'] == 'control']['amount']
df_t = df_abt_user_agg[df_abt_user_agg['group'] == 'test']['amount']

# Визуализируем:

# Создаём контейнер графика и задаём его размер
plt.figure(figsize=(15, 4))

# Строим горизонтальную диаграмму размаха
sns.boxplot(df_abt_user_agg, y='group', x='amount')


# Добавляем заголовок и подписи
plt.title('Распределение обьемов депозитов пользователей тестовой и контрольной групп')
```

```
plt.xlabel('Объем депозита, у.е')
plt.ylabel(' ')

# Устанавливаем частые отметки на оси X (например, шаг 10)
plt.xticks(range(0, int(df_abt_user_agg['amount'].max())+1, 50))

# Добавляем сетку
plt.grid(linestyle='--')

# Отображаем график
plt.show()
```



Вывод:

1. Распределение тестовой группы имеет длинный хвост в области высоких значений. Это может говорит о том, что пользователи на онбординге получают досточно знаний и уверенности для более существенных вложений в инвестиции. Или о том, что в тестовую группу случайно пришли пользователи с более высокими возможностями.
2. При этом медиана тестовой группы смещена левее к низким значениям и находится около ~ 45 у.е., а то время как в контрольной группе - ~ 95 у.е. Это говорит о том, что типичный пользователь тестовой группы стал инвестировать меньше, возможно онбординг снизил склонность к риску
3. 50% пользователей обеих групп имеют объемы инвестиций не превышающие 125-130 у.е. (в контрольной чуть больше, чем в тестовой). Если бы в данных тестовой и контрольной групп небыло выбросов, можно было предположить, что сервис имеет потолок вложений. Поэтому вероятно у большинства пользователей существует некоторый барьер для суммы инвестиций, которые пользователи "готовы"/"могут позволить себе" потерять в случае неудачной инвестиции.

Статистическая значимость различий метрик между группами эксперимента

Согласно расчетов, среднее значение депозита в тестовой группе больше на 2.2% , чем в контрольной группе. Проверим, являются ли это разница статистически значимой. Обозначим за μ_A и μ_B средние значения в группах control и test. Тогда гипотезы будут выглядеть так:

- H0: μ_A = μ_B , то есть средние депозиты в test-группе и в control-группе равны
- H1: μ_A < μ_B , то есть средний депозит в test-группе выше, чем в control-группе

In [75]:
```
# Подготовим данные:

# Агрегируем данные по пользователям и тестовым группам
df_abt_user_agg = df_abt.groupby(['user_id', 'group'])['amount'].sum().reset_index()

# Данные для тестовых групп
group_control = df_abt_user_agg[df_abt_user_agg['group'] == 'control']['amount']
group_test = df_abt_user_agg[df_abt_user_agg['group'] == 'test']['amount']

# Зафиксируем уровень значимости
alpha=0.05

# Считаем статистики
p_value_ab = ttest_ind(group_control, group_test, alternative='less').pvalue

# Проверяем условия и интерпретируем
if p_value_ab > alpha:
    print(f'pvalue={p_value_ab} > {alpha}')
    print(f'Нулевая гипотеза находит подтверждение! Значимых различий между средними значениями нет! ')
else:
    print(f'pvalue={p_value_ab} < {alpha}')
    print(f'Нулевая гипотеза не находит подтверждения! Существуют значимые различия!')
```

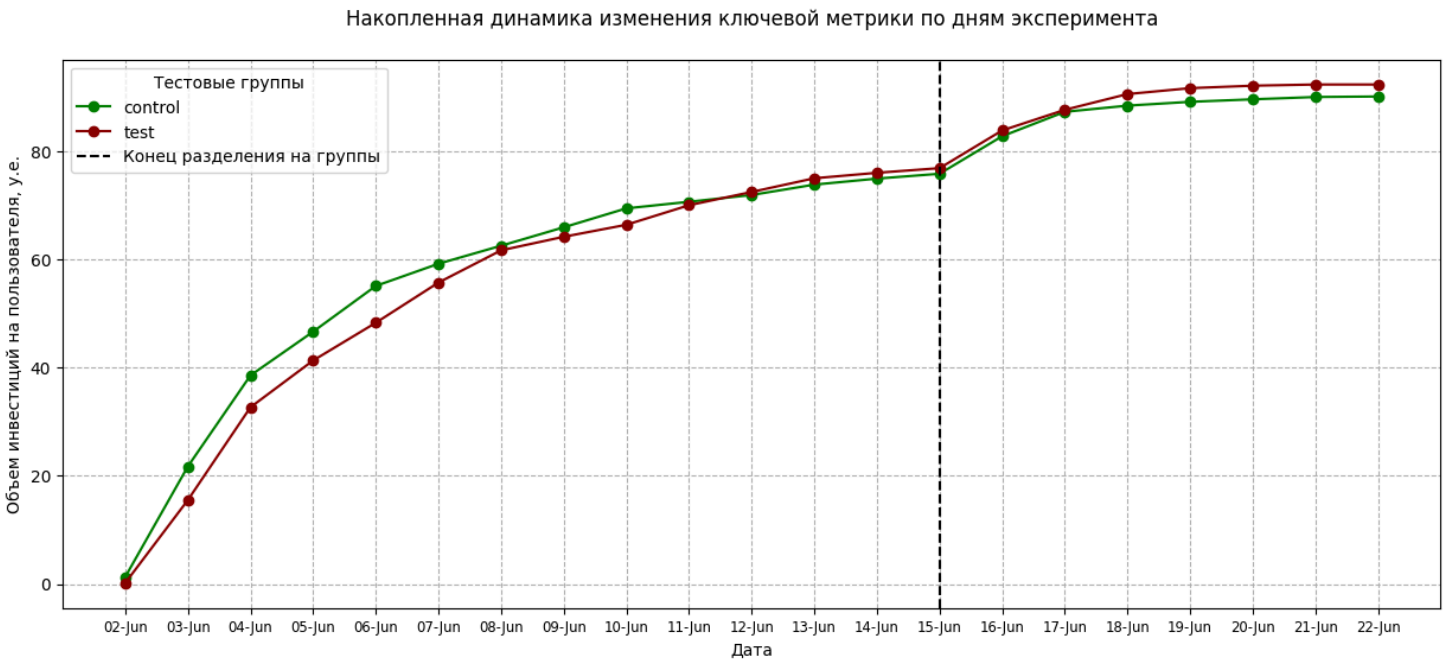
```
pvalue=0.16564696523216077 > 0.05
Нулевая гипотеза находит подтверждение! Значимых различий между средними значениями нет!
```

Накопленная динамика изменения по дням эксперимента для каждой группы

In [76]:
```
# Создаем поле с датой события
df_abt['event_dt'] =  pd.to_datetime(df_abt['event_ts'].dt.date)
```

In [77]:
```
# Подготовим данные:

# Агрегируем данные по дням активности, пользователям и тестовым группам
df_abt_agg = df_abt.groupby(['event_dt','user_id', 'group'])['amount'].sum().reset_index()

# Сортируем данные по возрастанию дат
df_abt_agg = df_abt_agg.sort_values(by='event_dt')

# Создаем список уникальных отсортированных дат
list_event_date_abt = np.sort(df_abt_agg['event_dt'].unique())

# Пустой словарь для результатов
records = {}

# Считаем кумулятивную метрику
for current_date in list_event_date_abt:
```

```
    # Фильтруем все события, оставляем записи до текущей даты
    active_to_date = df_abt_agg[df_abt_agg['event_dt'] <= current_date]

    # Получаем суммарное значение 'amount' на текущий день для каждого пользователя и группы
    group_avg = active_to_date.groupby(['user_id', 'group'])['amount'].sum().reset_index()
    # Получаем среднее значение на текущий день для каждой группы
    group_avg = group_avg.groupby('group')['amount'].mean()

    # Добавляем данныk в словарь
    records[current_date] = group_avg

# Преобразовываем словарь в датафрейм
df_mean_deposit_abt = pd.concat(records.values(), keys=records.keys())

# Внутренний уровень индекса строки переносим в столбцы
df_mean_deposit_abt = df_mean_deposit_abt.unstack().reset_index()

# Переименуем столбец
df_mean_deposit_abt = df_mean_deposit_abt.rename(columns={'index':'date'})



# Строим график:

# Создаем словарь с названиями тестовых групп и цветом для визуализации
list_name_group = {'control':'green', 'test':'darkred'}

# Построим график
plt.figure(figsize=(15, 6))

for name_group, color_group in list_name_group.items():
    # Строим линейный график
    plt.plot(df_mean_deposit_abt['date'], df_mean_deposit_abt[name_group], marker='o', color=color_group, label=name_group)

# Строим дополнительную линию
plt.axvline(x=pd.to_datetime('2025-06-15'), linestyle='--', color='black', label='Конец разделения на группы')

# Добавляем название
plt.title('Накопленная динамика изменения ключевой метрики по дням эксперимента\n')

# Добавляем подписи осей
plt.xlabel('Дата')
plt.ylabel('Объем инвестиций на пользователя, у.е.')

# Отображаем дополнительные линии
plt.grid(linestyle='--')
plt.legend(title="Тестовые группы")

# Создаем список меток
labels = df_mean_deposit_abt['date'].sort_values().dt.strftime('%d-%b')

# Передаем позиции,  метки и угол поворота
plt.xticks(df_mean_deposit_abt['date'], labels, fontsize='small')

plt.show()
```



Вывод:

1. Период со 2 по 12 июня , когда контрольная группа показывает более быстрый рост метрики: вероятно для пользователей тестовой группы начало инвестиций сдвигается из-за прохождения онбординга
2. С 12 июня тестовая группа опережает контрольную: вероятно наблюдаем эффект от онбординга.

Стабильность p-value во время эксперимента

Ранее был сформирован датафрейм `df_abt_agg` с агрегированными данными `amount` по дням активности, пользователям и тестовым группам, отсортированный по возрастанию дат `event_dt` . Используем его, чтобы рассчитать p-value по каждому дню активности:

In [78]:
```
df_abt_agg.head()
```

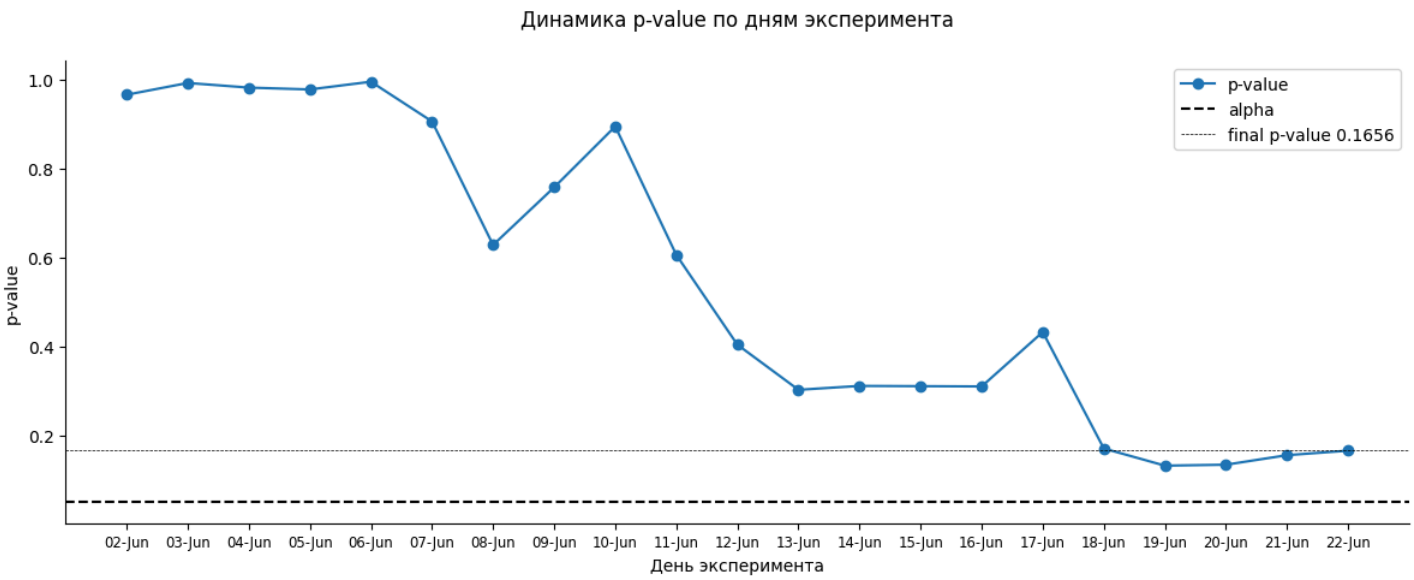
Out[78]:

	event_dt	user_id	group	amount
0	2025-06-02	00245e8c-747a-46d8-906d-dc18f373f1ce	test	0.0
478	2025-06-02	9c845b70-fd08-4a15-9192-38e637314603	control	0.0
479	2025-06-02	9cad4a46-fe67-4394-89b0-d301531b48b5	control	0.0
480	2025-06-02	9d2c7bc5-dbaf-46dc-8960-cbadb5e1a494	test	0.0
481	2025-06-02	9d9f932c-8488-49cc-81ca-9da1420107e6	test	0.0

In [79]:
```
list_event_date_abt
```

Out[79]:
```
array(['2025-06-02T00:00:00.000000000', '2025-06-03T00:00:00.000000000',
       '2025-06-04T00:00:00.000000000', '2025-06-05T00:00:00.000000000',
       '2025-06-06T00:00:00.000000000', '2025-06-07T00:00:00.000000000',
       '2025-06-08T00:00:00.000000000', '2025-06-09T00:00:00.000000000',
       '2025-06-10T00:00:00.000000000', '2025-06-11T00:00:00.000000000',
       '2025-06-12T00:00:00.000000000', '2025-06-13T00:00:00.000000000',
       '2025-06-14T00:00:00.000000000', '2025-06-15T00:00:00.000000000',
       '2025-06-16T00:00:00.000000000', '2025-06-17T00:00:00.000000000',
       '2025-06-18T00:00:00.000000000', '2025-06-19T00:00:00.000000000',
       '2025-06-20T00:00:00.000000000', '2025-06-21T00:00:00.000000000',
       '2025-06-22T00:00:00.000000000'], dtype='datetime64[ns]')
```

In [80]:
```
# Подготовим данные:

# Пустой список для результатов
pval_records = []

# Рассчитаем p-value по дням эксперимента

for current_date in list_event_date_abt:

    # Все события до текущей даты
    df_to_date = df_abt_agg[df_abt_agg['event_dt'] <= current_date]

    # Агрегируем по пользователям
    df_to_date = df_to_date.groupby(['user_id', 'group'])['amount'].sum().reset_index()

    # Сохраняем данные по каждой группе в соответствующую переменную
    control = df_to_date[df_to_date['group'] == 'control']['amount']
    test = df_to_date[df_to_date['group'] == 'test']['amount']

    # Рассчитываем p-value
    p_value_ab = ttest_ind(control, test, alternative='less').pvalue

    pval_records.append({
        'date': current_date,
        'p_value_ab': p_value_ab
    })

# Собираем в датафрейм
pvalue_df = pd.DataFrame(pval_records)


# Визуализируем:

# Задаем область и оси
fig, ax = plt.subplots(figsize=(12,5))

# Строим линейный график
ax.plot(pvalue_df['date'], pvalue_df['p_value_ab'], marker='o', label='p-value')

# Строим дополнительную линию
ax.axhline(y=0.05, linestyle='--', color='black', label='alpha')

# Строим дополнительную линию
ax.axhline(y=pvalue_df['p_value_ab'].iloc[-1],
           linestyle='--',
           linewidth=0.5, color='black',
           label=f"final p-value {pvalue_df['p_value_ab'].iloc[-1]:.4f}")

# Задаем название графика и подписи осей
ax.set_title('Динамика p-value по дням эксперимента\n')
ax.set_xlabel('День эксперимента')
ax.set_ylabel('p-value')

# Создаем список меток
labels = pvalue_df['date'].sort_values().dt.strftime('%d-%b')

# Передаем позиции,  метки и угол поворота
plt.xticks(pvalue_df['date'], labels, fontsize='small')

# Убираем границы
ax.spines[['top', 'right']].set_visible(False)

# Отображаем легенду
plt.legend()

# Регулируем отступы и расположение элементов
plt.tight_layout()

plt.show()
```



Вывод:

1. В первые дни эксперимента p-value достаточно большой, наблюдаются колебания свидетельствующие о нестабильности и невозможности делать преждевременные выводы.
2. После 19 июня p-value стабилизируется и ведет себя монотонно.

Анализ барьерной метрики - конверсии из регистрации в первый депозит

Расчет метрик

Обозначим для удобства этапы цифрами:

1. install / open_web
2. introduction
3. registration
4. main_page
5. onboarding_complete
6. first_deposit
7. asset_purchase
8. second_deposit

In [81]:
```
# Создаем список с названиями тестовых групп
list_name_group = ['control', 'test']

# Создаем список шагов воронки
list_event_part = ['registration', 'first_deposit']

# Создаем пустой словарь "Конверсия из 3 этапа в 6 этап"
dict_cr_from_3_to_6 = {}

# Создаем пустой словарь для количество пользователей на 3 этапе и на 6 этапе
dict_from_3_to_6 = {}



for name_group in list_name_group:

    # Фильтруем данные, оставив только нужные шаги воронки
    df_filtered = df_abt[df_abt['event_name'].isin(list_event_part)]

    # Считаем число уникальных пользователей на каждом шаге
    # Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
    df_test = df_filtered[
        df_filtered['group'] == name_group
    ].groupby('event_name')['user_id'].nunique().reindex(list_event_part).reset_index()

    # Переименовываем поля
    df_test = df_test.rename(columns = {'event_name':'step', 'user_id':'users'})

    # Рассчитаем конверсию
    # Поскольку шаги воронки расположены по порядку запись с индексом 0 - это регистрация, с 1 - первый депозит
    cr = df_test.loc[1,'users'] * 100 / df_test.loc[0, 'users']

    # Записываем конверсию в словарь
    dict_cr_from_3_to_6[name_group] = cr
    # Записываем датафреймы в словарь
    dict_from_3_to_6[name_group] = df_test



# Преобразываем словарь в датафрейм
dt_cr_from_3_to_6 = pd.DataFrame(dict_cr_from_3_to_6, index=[0])

# Сохраняем в переменные конверсию тестовых групп
cr_control = dt_cr_from_3_to_6.loc[0,'control']
cr_test = dt_cr_from_3_to_6.loc[0,'test']
# Производим расчет абсолютной разницы между группами
dt_cr_from_3_to_6['absolute _diff'] = cr_test - cr_control

# Производим расчет относительной разницы между группами
dt_cr_from_3_to_6['relative_diff'] = 100 * fabs(cr_test - cr_control) / cr_control
```

```
# Сохраняем датафреймы в переменные:
df_from_3_to_6_control = dict_from_3_to_6['control']
df_from_3_to_6_test = dict_from_3_to_6['test']
```

In [82]:
```
print('Конверсия из регистрации в первый депозит:')
dt_cr_from_3_to_6
```

```
Конверсия из регистрации в первый депозит:
```

Out[82]:

	control	test	absolute _diff	relative_diff
0	71.542553	70.03517	-1.507383	2.106974

Статистическая значимость различий метрик между группами эксперимента

Согласно расчетов, конверсии из регистрации в первый депозит в тестовой группе ниже, чем в контрольной. Проверим, является ли это изменение статистически значимыми.
Обозначим:

- n_a и n_b - количество зарегистрированных пользователей в control и test группах соответственно,
- m_a и m_b - количество пользователей открывших первый депозит в control и test группах соответственно,
- p_a и p_b - доли в control и test группах соответственно

Тогда гипотезы будут выглядеть так:

- H0: pA = pB , то есть конверсии в test-группе и control-группе равны
- H1: pA > pB то есть конверсия в test-группе хуже конверсии в control-группе

In [83]:
```
df_from_3_to_6_test
```

Out[83]:

	step	users
0	registration	4265
1	first_deposit	2987

In [84]:
```
# Фиксируем уровень значимости
alpha = 0.05


# 1 шаг - подготовка данных:

# Получаем данные:

# Число пользователей в контрольной группе на этапе 3 - регистрация
n_a = df_from_3_to_6_control.loc[0,'users']

# Число пользователей в контрольной группе на этапе 6 - первый депозит
m_a = df_from_3_to_6_control.loc[1,'users']

# Доля группы control
p_a = dt_cr_from_3_to_6['control'].values[0] / 100

# Число пользователей в тестовой группе на этапе 3 - регистрация
n_b = df_from_3_to_6_test.loc[0,'users']

# Число пользователей в тестовой группе на этапе 6 - первый депозит
m_b = df_from_3_to_6_test.loc[1,'users']

# Доля группы test
p_b = dt_cr_from_3_to_6['test'].values[0] / 100


# 2 шаг - проверяем, выполняется ли предпосылка Z-теста пропорций о достаточном размере выборок:

# Проверяем выполнение предпосылки и выводим результат
if (p_a*n_a > 10)and((1-p_a)*n_a > 10)and(p_b*n_b > 10)and((1-p_b)*n_b > 10):
    print(f'Предпосылка о достаточном количестве данных выполняется!')
else:
    print(f'⚠Предпосылка о достаточном количестве данных НЕ выполняется!')


# 3 шаг - рассчитаем размер эффекта - индекс Cohen's h:

# Считаем размер эффекта
effect_size = proportion_effectsize(p_a, p_b)

# Выводим результат
print(f"Размер эффекта Cohen's h: {(effect_size):.6f}")


# 4 шаг - рассчитаем фактическую мощность:

# Считаем соотношение размеров групп
ratio = n_b / n_a

# Считаем фактическую мощность
power = zt_ind_solve_power(
    effect_size=effect_size,
    nobs1=n_a,
    alpha=alpha,
    ratio=ratio
)

print(f"Мощность теста для: {power:.2f}")

# 5 шаг - применяем z-тест пропорций для проверки гипотез:

# Считаем статистики
stat_ztest, p_value_ztest = proportions_ztest([m_a, m_b], [n_a, n_b], alternative='larger')

# Проверяем условия и интерпретируем
if p_value_ztest > alpha:
```

```
    print(f'pvalue={p_value_ztest} > {alpha}')
    print(f'Нулевая гипотеза находит подтверждение! Вывод: конверсии двух групп равны!')
else:
    print(f'pvalue={p_value_ztest} < {alpha}')
    print(f'✅ Нулевая гипотеза не находит подтверждения! Вывод: конверсия значимо хуже!')

print("----------------")
```

```
Предпосылка о достаточном количестве данных выполняется!
Размер эффекта Cohen's h: 0.033151
Мощность теста для: 0.34
pvalue=0.06027172404556423 > 0.05
Нулевая гипотеза находит подтверждение! Вывод: конверсии двух групп равны!
----------------
```

Вывод:

Согласно проведенному тесту, статистически значимого различия между конверсиями двух групп нет, но важно учесть, что:

- p-value равен 0.06 и находится очень близко к уровню значимости.
- размер эффекта крайне мал - 0.033
- мощность значительно ниже стандартного порога - существует 66% вероятность не обнаружить существующий эффект

  Даже если бы тест показал значимость, то практическая значимость от такого малого эффекта сомнительна.

Конверсия из регистрации в первый депозит по платформам

Продуктовая команда опасалась, что обновленный онбординг может отпугнуть некоторых пользователей и снизить конверсию в первый депозит. Согласно расчетам, конверсия из регистрации в первый депозит действительно снизилась на 1,5%, но это снижение не является статистически значимым. Проверим есть ли падение метрики в разных сегментах - платформах и странах.

In [85]:
```
# По платформам:

# Список событий: регистрация и первый депозит
list_event_part = ['registration', 'first_deposit']

# Фильтруем датасет. Оставляем только события из списка
df_abt_part = df_abt[df_abt['event_name'].isin(list_event_part)]

# Список имен групп
list_name_group = ['control', 'test']

# Пустой словарь для записи результатов кода
dict_df = {}


# В цикле для каждой группы сформируем данные о конверсии из одного шага другой для каждого сегмента:
for name_group in list_name_group:

    # Фильтруем исходный датафрейм для текущей тестовой группы
    df_abt_filtered = df_abt_part[df_abt_part['group'] == name_group]

    # Считаем число уникальных пользователей на каждом шаге для разных платформ:
    df_funnel = df_abt_filtered.groupby(['platform','event_name'])['user_id'].nunique()

    # Записываем данные в словарь
    dict_df[name_group] =  df_funnel


#  Сохраняем в переменные
df_abt_platform_control = dict_df['control']
df_abt_platform_test = dict_df['test']

# Преобразуем мультииндексы в столбцы
df_abt_platform_control = df_abt_platform_control.unstack()#.reset_index()
df_abt_platform_test = df_abt_platform_test.unstack()#.reset_index()

# Переименуем столбы
df_abt_platform_control = df_abt_platform_control.rename(
    columns={'first_deposit':'control_first_deposit',
             'registration':'control_registration'}
)
df_abt_platform_test = df_abt_platform_test.rename(
    columns={'first_deposit':'test_first_deposit',
             'registration':'test_registration'}
)

# Добавляем поле с конверсией
df_abt_platform_control['control_conversion_%'] = (
    100 * df_abt_platform_control['control_first_deposit'] / df_abt_platform_control['control_registration']
)
df_abt_platform_test['test_conversion_%'] = (
    100 * df_abt_platform_test['test_first_deposit']  / df_abt_platform_test['test_registration']
)


# Объединяем датафреймы
df_abt_platform = df_abt_platform_control.merge(df_abt_platform_test, on='platform')


# Производим расчет абсолютной разницы между группами
df_abt_platform['absolute_diff'] = df_abt_platform['test_conversion_%'] - df_abt_platform['control_conversion_%']

# Производим расчет относительной разницы между группами
df_abt_platform['relative_diff'] = 100 * abs(df_abt_platform['absolute_diff']) / df_abt_platform['control_conversion_%']
```

In [86]:
```
print('Конверсия из регистрации в первый депозит:')
df_abt_platform
```

```
Конверсия из регистрации в первый депозит:
```

Out[86]:

event_name	control_first_deposit	control_registration	control_conversion_%	test_first_deposit	test_registration	test_conversion_%	absolute_diff	relative_diff
platform								
mobile	2250	3144	71.564885	2142	3042	70.414201	-1.150684	1.607890
web	978	1368	71.491228	845	1223	69.092396	-2.398832	3.355422

Вывод:

Конверсия из регистрации в первый депозит снижена на обеих платформах, однако для web-версии падение более существенно - относительная разница для web составляет 3.4%, для mobile 1.6%.

Конверсия из регистрации в первый депозит по странам

In [87]:
```
# По странам:

# Список событий: регистрация и первый депозит
list_event_part = ['registration', 'first_deposit']

# Фильтруем датасет. Оставляем только события из списка
df_abt_part = df_abt[df_abt['event_name'].isin(list_event_part)]

# Список имен групп
list_name_group = ['control', 'test']

# Пустой словарь для записи результатов кода
dict_df = {}


# В цикле для каждой группы сформируем данные о конверсии из одного шага другой для каждого сегмента:
for name_group in list_name_group:

    # Фильтруем исходный датафрейм для текущей тестовой группы
    df_abt_filtered = df_abt_part[df_abt_part['group'] == name_group]

    # Считаем число уникальных пользователей на каждом шаге для разных стран:
    df_funnel = df_abt_filtered.groupby(['country_code','event_name'])['user_id'].nunique()

    # Записываем данные в словарь
    dict_df[name_group] =  df_funnel


#  Сохраняем в переменные
df_abt_country_control = dict_df['control']
df_abt_country_test = dict_df['test']

# Преобразуем мультииндексы в столбцы
df_abt_country_control = df_abt_country_control.unstack()#.reset_index()
df_abt_country_test = df_abt_country_test.unstack()#.reset_index()

# Переименуем столбы
df_abt_country_control = df_abt_country_control.rename(
    columns={'first_deposit':'control_first_deposit',
             'registration':'control_registration'}
)
df_abt_country_test = df_abt_country_test.rename(
    columns={'first_deposit':'test_first_deposit',
             'registration':'test_registration'}
)

# Добавляем поле с конверсией
df_abt_country_control['control_conversion_%'] = (
    100 * df_abt_country_control['control_first_deposit'] / df_abt_country_control['control_registration']
)
df_abt_country_test['test_conversion_%'] = (
    100 * df_abt_country_test['test_first_deposit']  / df_abt_country_test['test_registration']
)


# Объединяем датафреймы
df_abt_country = df_abt_country_control.merge(df_abt_country_test, on='country_code')


# Производим расчет абсолютной разницы между группами
df_abt_country['absolute_diff'] = df_abt_country['test_conversion_%'] - df_abt_country['control_conversion_%']

# Производим расчет относительной разницы между группами
df_abt_country['relative_diff'] = 100 * abs(df_abt_country['absolute_diff']) / df_abt_country['control_conversion_%']
```

In [88]:
```
print('Конверсия из регистрации в первый депозит:')
df_abt_country
```

```
Конверсия из регистрации в первый депозит:
```

Out[88]:

event_name	control_first_deposit	control_registration	control_conversion_%	test_first_deposit	test_registration	test_conversion_%	absolute_diff	relative_diff
country_code								
AR	480	697	68.866571	486	684	71.052632	2.186061	3.174342
BR	1023	1426	71.739130	949	1344	70.610119	-1.129011	1.573773
CO	841	1167	72.065124	775	1110	69.819820	-2.245304	3.115660
MX	884	1222	72.340426	777	1127	68.944099	-3.396326	4.694921

Вывод:

Не все страны показывают падение метрики. Например, для Аргентины конверсия из регистрации в первый депозит выросла на 3.2% п.п., остальные страны показывают падение , при этом наибольшее падение метрики наблюдается для Мексики - почти 5% п.п.

Анализ вспомогательной метрики 1 - конверсии из первого депозита во второй

In [89]:
```
# Создаем список с названиями тестовых групп
list_name_group = ['control', 'test']

# Создаем список шагов воронки
list_event_part = ['first_deposit', 'second_deposit']

# Создаем пустой словарь
dict_cr_from_6_to_8 = {}

# Создаем пустой словарь для количество пользователей на 3 этапе и на 6 этапе
dict_from_6_to_8 = {}

for name_group in list_name_group:

    df_filtered = df_abt[df_abt['event_name'].isin(list_event_part)]

    # Считаем число уникальных пользователей на каждом шаге
    # Для упорядочивания записей в соответствии со списком событий применим метод .reindex()
    df_test = df_filtered[
        df_filtered['group'] == name_group
    ].groupby('event_name')['user_id'].nunique().reindex(list_event_part).reset_index()

    # Переименовываем поля
    df_test = df_test.rename(columns = {'event_name':'step', 'user_id':'users'})

    # Рассчитаем конверсию
    # Поскольку шаги воронки расположены по порядку запись с индексом 0 - это регистрация, с 1 - первый депозит
    cr = df_test.loc[1,'users'] * 100 / df_test.loc[0, 'users']

    # Записываем данные в словарь
    dict_cr_from_6_to_8[name_group] = cr
    # Записываем датафреймы в словарь
    dict_from_6_to_8[name_group] = df_test



# Преобразываем словарь в датафрейм
df_cr_from_6_to_8 = pd.DataFrame(dict_cr_from_6_to_8, index=[0])


cr_control = df_cr_from_6_to_8.loc[0,'control']
cr_test = df_cr_from_6_to_8.loc[0,'test']

# Производим расчет абсолютной разницы между группами
df_cr_from_6_to_8['absolute _diff'] = cr_test - cr_control

# Производим расчет относительной разницы между группами
df_cr_from_6_to_8['relative_diff'] = 100 * fabs(cr_test - cr_control) / cr_control


# Сохраняем датафреймы в переменные:
df_from_6_to_8_control = dict_from_6_to_8['control']
df_from_6_to_8_test = dict_from_6_to_8['test']
```

In [90]:
```
print('Конверсия из первого депозита во второй депозит:')
df_cr_from_6_to_8
```

```
Конверсия из первого депозита во второй депозит:
```

Out[90]:

	control	test	absolute _diff	relative_diff
0	20.260223	32.607968	12.347745	60.94575

Статистическая значимость различий метрик между группами эксперимента

Согласно расчетов, конверсии из первого депозита во второй депозит в тестовой группе выше, чем в контрольной. Проверим, является ли это изменение статистически значимыми. Обозначим:

- n_a и n_b - количество пользователей открывших первый депозит в control и test группах соответственно,
- m_a и m_b - количество пользователей открывших второй депозит в control и test группах соответственно,
- p_a и p_b - доли в control и test группах соответственно

  Тогда гипотезы будут выглядеть так:

- H0: pA = pB , то есть конверсии в test-группе и control-группе равны
- H1: pA < pB то есть конверсия в test-группе выше конверсии в control-группе

In [91]:
```
df_from_6_to_8_test
```

Out[91]:

	step	users
0	first_deposit	2987
1	second_deposit	974

In [92]:
```
df_from_6_to_8_control
```

Out[92]:

	step	users
0	first_deposit	3228
1	second_deposit	654

In [93]:
```
# Фиксируем уровень значимости
alpha = 0.05


# 1 шаг - подготовка данных:
```

```
# Получаем данные:

# Число пользователей в контрольной группе на этапе 6 - первый депозит
n_a = df_from_6_to_8_control.loc[0,'users']

# Число пользователей в контрольной группе на этапе 8 - второй депозит
m_a = df_from_6_to_8_control.loc[1,'users']

# Доля группы control
p_a = df_cr_from_6_to_8['control'].values[0] / 100

# Число пользователей в тестовой группе на этапе 6 - первый депозит
n_b = df_from_6_to_8_test.loc[0,'users']

# Число пользователей в тестовой группе на этапе 8 - второй депозит
m_b = df_from_6_to_8_test.loc[1,'users']

# Доля группы test
p_b = df_cr_from_6_to_8['test'].values[0] / 100


# 2 шаг - проверяем, выполняется ли предпосылка Z-теста пропорций о достаточном размере выборок:

# Проверяем выполнение предпосылки и выводим результат
if (p_a*n_a > 10)and((1-p_a)*n_a > 10)and(p_b*n_b > 10)and((1-p_b)*n_b > 10):
    print(f'Предпосылка о достаточном количестве данных выполняется!')
else:
    print(f'⚠Предпосылка о достаточном количестве данных НЕ выполняется!')


# 3 шаг - рассчитаем размер эффекта - индекс Cohen's h:

# Считаем размер эффекта
effect_size = proportion_effectsize(p_b, p_a)

# Выводим результат
print(f"Размер эффекта Cohen's h: {(effect_size):.6f}")


# 4 шаг - рассчитаем фактическую мощность:

# Считаем соотношение размеров групп
ratio = n_b / n_a

# Считаем фактическую мошность
power = zt_ind_solve_power(
    effect_size=effect_size,
    nobs1=n_a,
    alpha=alpha,
    ratio=ratio
)

print(f"Мощность теста для: {power:.2f}")

# 5 шаг - применяем z-тест пропорций для проверки гипотез:

# Считаем статистики
stat_ztest, p_value_ztest = proportions_ztest([m_a, m_b], [n_a, n_b], alternative='smaller')

# Проверяем условия и интерпретируем
if p_value_ztest > alpha:
    print(f'pvalue={p_value_ztest} > {alpha}')
    print(f'Нулевая гипотеза находит подтверждение! Вывод: конверсии двух групп равны!')
else:
    print(f'pvalue={p_value_ztest} < {alpha}')
    print(f'✅ Нулевая гипотеза не находит подтверждения! Вывод: конверсия значимо лучше!')

print("----------------")
```

```
Предпосылка о достаточном количестве данных выполняется!
Размер эффекта Cohen's h: 0.281744
Мощность теста для: 1.00
pvalue=9.677942837686964e-29 < 0.05
✅ Нулевая гипотеза не находит подтверждения! Вывод: конверсия значимо лучше!
----------------
```

Вывод:

Тест показал статистически значимое улучшение конверсии в тестовой группе:

- p-value крайне мал
- средний размер эффекта - 0.28
- мощность 100%

Конверсия из первого деппозита во второй по платформам

In [94]:
```
# Создаем список шагов воронки
list_event_part = ['first_deposit', 'second_deposit']

# Создаем список групп
list_name_group=['control', 'test']

# Задаем название сегмента
segment='platform'

# Пустой словарь для записи результатов кода
dict_df = {}

# Фильтруем датасет. Оставляем только события из списка
df_abt_part = df_abt[df_abt['event_name'].isin(list_event_part)]


# В цикле для каждой группы сформируем данные о конверсии из одного шага другой для каждого сегмента:
for name_group in list_name_group:
```

```
    # Фильтруем исходный датафрейм для текущей тестовой группы
    df_abt_filtered = df_abt_part[df_abt_part['group'] == name_group]

    # Считаем число уникальных пользователей на каждом шаге для разных сегментов:
    df_funnel = df_abt_filtered.groupby([segment,'event_name'])['user_id'].nunique()

    # Записываем данные в словарь
    dict_df[name_group] =  df_funnel


#  Сохраняеем в переменные
df_abt_segment_control = dict_df['control']
df_abt_segment_test = dict_df['test']

# Преобразуем мультииндексы в столбцы
df_abt_segment_control = df_abt_segment_control.unstack()#.reset_index()
df_abt_segment_test = df_abt_segment_test.unstack()#.reset_index()

# Переименуем столбы
df_abt_segment_control = df_abt_segment_control.rename(
    columns={'first_deposit':'control_first_deposit',
             'second_deposit':'control_second_deposit'}
)
df_abt_segment_test = df_abt_segment_test.rename(
    columns={'first_deposit':'test_first_deposit',
             'second_deposit':'test_second_deposit'}
)

# Добавляем поле с конверсией
df_abt_segment_control['control_conversion_%'] = (
    100 * df_abt_segment_control['control_second_deposit'] / df_abt_segment_control['control_first_deposit']
)
df_abt_segment_test['test_conversion_%'] = (
    100 * df_abt_segment_test['test_second_deposit']  / df_abt_segment_test['test_first_deposit']
)


# Объединяем датафреймы
df_abt_segment = df_abt_segment_control.merge(df_abt_segment_test, on=segment)


# Производим расчет абсолютной разницы между группами
df_abt_segment['absolute_diff'] = df_abt_segment['test_conversion_%'] - df_abt_segment['control_conversion_%']

# Производим расчет относительной разницы между группами
df_abt_segment['relative_diff'] = 100 * abs(df_abt_segment['absolute_diff']) / df_abt_segment['control_conversion_%']
```

In [95]:
```
print('Конверсия из первого депозита во второй депозит:')
df_abt_segment
```

```
Конверсия из первого депозита во второй депозит:
```

Out[95]:

event_name	control_first_deposit	control_second_deposit	control_conversion_%	test_first_deposit	test_second_deposit	test_conversion_%	absolute_diff	relative_diff
platform								
mobile	2250	449	19.955556	2142	697	32.539683	12.584127	63.060770
web	978	205	20.961145	845	277	32.781065	11.819920	56.389667

Вывод:

Все платформы показывают увеличение конверсии, отнсительное значение прироста 56.4% для web и 63% для mobile.

Конверсия из регистрации в первый депозит по странам

In [96]:
```
# Создаем список шагов воронки
list_event_part = ['first_deposit', 'second_deposit']

# Создаем список групп
list_name_group=['control', 'test']

# Задаем название сегмента
segment='country_code'

# Пустой словарь для записи результатов кода
dict_df = {}

# Фильтруем датасет. Оставляем только события из списка
df_abt_part = df_abt[df_abt['event_name'].isin(list_event_part)]


# В цикле для каждой группы сформируем данные о конверсии из одного шага другой для каждого сегмента:
for name_group in list_name_group:

    # Фильтруем исходный датафрейм для текущей тестовой группы
    df_abt_filtered = df_abt_part[df_abt_part['group'] == name_group]

    # Считаем число уникальных пользователей на каждом шаге для разных сегментов:
    df_funnel = df_abt_filtered.groupby([segment,'event_name'])['user_id'].nunique()

    # Записываем данные в словарь
    dict_df[name_group] =  df_funnel


#  Сохраняеем в переменные
df_abt_segment_control = dict_df['control']
df_abt_segment_test = dict_df['test']

# Преобразуем мультииндексы в столбцы
df_abt_segment_control = df_abt_segment_control.unstack()#.reset_index()
df_abt_segment_test = df_abt_segment_test.unstack()#.reset_index()

# Переименуем столбы
df_abt_segment_control = df_abt_segment_control.rename(
```

```
    columns={'first_deposit':'control_first_deposit',
             'second_deposit':'control_second_deposit'}
)
df_abt_segment_test = df_abt_segment_test.rename(
    columns={'first_deposit':'test_first_deposit',
             'second_deposit':'test_second_deposit'}
)

# Добавляем поле с конверсией
df_abt_segment_control['control_conversion_%'] = (
    100 * df_abt_segment_control['control_second_deposit'] / df_abt_segment_control['control_first_deposit']
)
df_abt_segment_test['test_conversion_%'] = (
    100 * df_abt_segment_test['test_second_deposit']  / df_abt_segment_test['test_first_deposit']
)


# Объединяем датафреймы
df_abt_segment = df_abt_segment_control.merge(df_abt_segment_test, on=segment)


# Производим расчет абсолютной разницы между группами
df_abt_segment['absolute_diff'] = df_abt_segment['test_conversion_%'] - df_abt_segment['control_conversion_%']

# Производим расчет относительной разницы между группами
df_abt_segment['relative_diff'] = 100 * abs(df_abt_segment['absolute_diff']) / df_abt_segment['control_conversion_%']
```

In [97]:
```
print('Конверсия из первого депозита во второй депозит:')
df_abt_segment
```

```
Конверсия из первого депозита во второй депозит:
```

Out[97]:

event_name	control_first_deposit	control_second_deposit	control_conversion_%	test_first_deposit	test_second_deposit	test_conversion_%	absolute_diff	relative_diff
country_code								
AR	480	103	21.458333	486	173	35.596708	14.138374	65.887570
BR	1023	193	18.866080	949	300	31.612223	12.746143	67.561163
CO	841	196	23.305589	775	258	33.290323	9.984734	42.842660
MX	884	162	18.325792	777	243	31.274131	12.948339	70.656371

Вывод:

Все страны показывают увеличение конверсии относительное значение прироста от 42% для Колумбии до 71% для Мексики.

Анализ вспомогательной метрики 2 - средней суммы всех депозитов на пользователя, который открыл хотя бы один депозит

In [98]:
```
# Подготовим данные

# Создаем список групп
list_name_group = ['control', 'test']

# Создаем пустой словарь
dict_mean_total_amount_abt = {}

# Создаем список событий
list_event_part = ['first_deposit', 'second_deposit']

# В цикле проходимся по каждой группе
for name_group in list_name_group:

    # Получаем сумму 'amount' по всем данным для текущей тестовой группы для каждого пользователя
    sum_amount = df_abt[df_abt['group'] == name_group].groupby(['user_id', 'event_dt'])['amount'].sum()

    # Для текущей группы считаем число уникальных пользователей открышвих хотя бы один депозит
    count_active_user = df_abt[(
        (df_abt['group'] == name_group) & (df_abt['event_name'].isin(list_event_part))
    )]['user_id'].nunique()

    # считаем среднее значение для текущей группы
    mean_amount = sum_amount.sum() / count_active_user

    # Записываем данные в словарь
    dict_mean_total_amount_abt[name_group] =  mean_amount


# Преобразываем словарь в датафрейм
df_mean_amount_abt = pd.DataFrame(dict_mean_total_amount_abt, index=[0])

mean_control = df_mean_amount_abt.loc[0,'control']
mean_test = df_mean_amount_abt.loc[0,'test']

# Производим расчет абсолютной разницы между группами
df_mean_amount_abt['absolute _diff'] = mean_test - mean_control

# Производим расчет относительной разницы между группами
df_mean_amount_abt['relative_diff'] = 100 * fabs(mean_test - mean_control) / mean_control
```

In [99]:
```
print('Средняя сумма всех депозитов на одного пользователя, открывшего хотя бы один депозит:')
df_mean_amount_abt
```

```
Средняя сумма всех депозитов на одного пользователя, открывшего хотя бы один депозит:
```

Out[99]:

	control	test	absolute _diff	relative_diff
0	135.5	141.399732	5.899732	4.354046

Согласно расчетов, среднее значение депозита в тестовой группе больше на 4.4% , чем в контрольной группе. Проверим, является ли это разница статистически значимой.
Обозначим за μ_A и μ_B средние значения в группах control и test. Тогда гипотезы будут выглядеть так:

- H0: μ_A = μ_B , то есть средние депозиты в test-группе и в control-группе равны

- H1: μ_A < μ_B , то есть средний депозит в test-группе выше, чем в control-группе

In [100]:
```
# Подготовим данные:

# Создаем список событий
list_event_part = ['first_deposit', 'second_deposit']

# Агрегируем данные по пользователям и тестовым группам
df_abt_user_agg = df_abt[df_abt['event_name'].isin(list_event_part)].groupby(['user_id', 'group'])['amount'].sum().reset_index()

# Подготовим данные для тестовых групп
group_control = df_abt_user_agg[df_abt_user_agg['group'] == 'control']['amount']
group_test = df_abt_user_agg[df_abt_user_agg['group'] == 'test']['amount']

# Зафиксируем уровень значимости
alpha=0.05

# Считаем статистики
p_value_ab = ttest_ind(group_control, group_test, alternative='less').pvalue

# Проверяем условия и интерпретируем
if p_value_ab > alpha:
    print(f'pvalue={p_value_ab} > {alpha}')
    print(f'Нулевая гипотеза находит подтверждение! Значимых различий между средними значениями нет! ')
else:
    print(f'pvalue={p_value_ab} < {alpha}')
    print(f'Нулевая гипотеза не находит подтверждения! Существуют значимые различия!')
```

```
pvalue=0.01734214320081285 < 0.05
Нулевая гипотеза не находит подтверждения! Существуют значимые различия!
```

Вывод:

Средняя сумма всех депозитов на пользователя, открывшего хотя бы один депозит, в тестовой группе статистически значимо больше чем в контрольной группе.

Анализ изменений суммы депозитов на платящего пользователя

Новая фича могла повлиять на поведение пользователей.

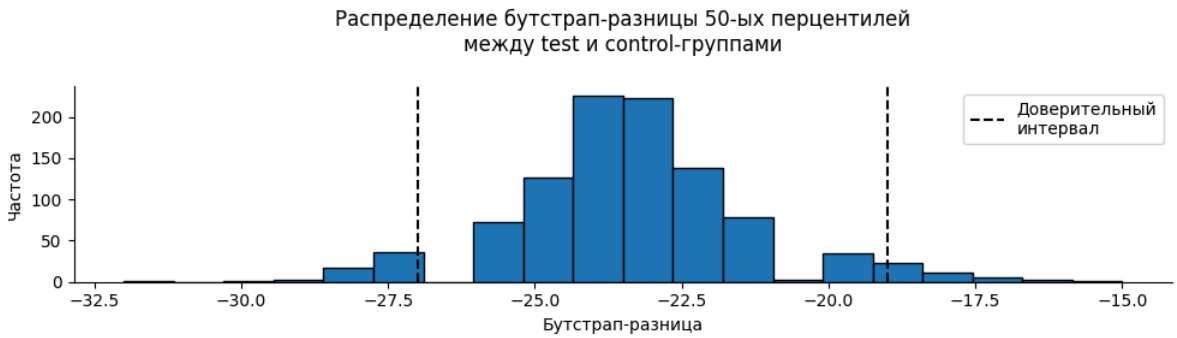
- Пользователи, которые раньше вносили небольшие суммы, могли стать более осторожными, сократить свои вложения или совсем перестать платить. Это отразится в снижении 25-го перцентиля суммы депозитов в тестовой группе.
- Пользователи, которые склонны к более крупным инвестициям, могли сильнее вовлечься в продукт и начать вносить больше средств. Это отразится в росте 75-го перцентиля.

Используя бутстрап, сравним разницы перцентилей (25, 50, 75) суммы всех депозитов на платящего пользователя в контрольной и тестовой группах.

In [101]:
```
# Создаем список перцентилей
list_percentile = [25, 50, 75]

# Сохраним в переменные данные по группам
control = group_control
test = group_test

# Фиксируем сид для воспроизводимости
np.random.seed(341)

for name_percentile in list_percentile:

    # 1. Расчет:
    # Задаем количество итераций
    n_iterations = 1000
    boot_diffs = []

    # Бутстрап-разница n-ых перцентилей между тестовой и контрольной группами
    for i in range(n_iterations):
        # Генерируем выборки того же размера с повторениями
        boot_control = np.random.choice(control, size=len(control), replace=True)
        boot_test = np.random.choice(test, size=len(test), replace=True)

        # Для каждой сгенерированной выборки рассчитываем 95 перцентиль
        control_perc = np.percentile(boot_control, name_percentile)
        test_perc = np.percentile(boot_test, name_percentile)

        # Добавляем в список разницу между значениями
        boot_diffs.append(test_perc - control_perc)

    # Преобразуем список в массив для подсчета статистик
    boot_diffs = np.array(boot_diffs)

    # Вычисляем 95 доверительный интервал для распределения разниц boot_diffs
    diff_ci = np.percentile(boot_diffs, [2.5, 97.5])


    # 2. Визуализация:

    # Задаем область и оси
    fig, ax = plt.subplots(figsize=(10,3))

    # Строим гистограмму бутстрап-теста
    ax.hist(boot_diffs, bins=20, edgecolor='black')
    # Строим дополнительную линию
    ax.axvline(x=diff_ci[0], linestyle='--', color='black', label='Доверительный\пинтервал')
    ax.axvline(x=diff_ci[1], linestyle='--', color='black')

    # Указываем заголовок и подписи осей
    plt.title(f'Распределение бутстрап-разницы {name_percentile}-ых перцентилей\nмежду test и control-группами\n')
    plt.xlabel('Бутстрап-разница')
    plt.ylabel('Частота')

    # Убираем границы
    ax.spines[['top', 'right']].set_visible(False)

    # Отображаем легенду
```

```
    plt.legend()

    # Регулируем отступы и расположение элементов
    plt.tight_layout()

    # Отображаем график
    plt.show()

    # Вывод доверительного интервала
    print(f"{name_percentile}-й перцентиль контрольной группы: {np.percentile(control, name_percentile):.2f}")
    print(f"{name_percentile}-й перцентиль тестовой группы: {np.percentile(test, name_percentile):.2f}")
    print(f"95%-й доверительный интервал разности {name_percentile}-х перцентилей [{diff_ci[0]:.2f}, {diff_ci[1]:.2f}]")

    print("-----------------------------------------------------------------------------------------------------------")
```


```
25-й перцентиль контрольной группы: 95.00
25-й перцентиль тестовой группы: 48.00
95%-й доверительный интервал разности 25-х перцентилей [-50.00, -44.00]
-----------------------------------------------------------------------------------------------------------
```




```
50-й перцентиль контрольной группы: 118.00
50-й перцентиль тестовой группы: 94.00
95%-й доверительный интервал разности 50-х перцентилей [-27.00, -19.00]
-----------------------------------------------------------------------------------------------------------
```


```
75-й перцентиль контрольной группы: 159.00
75-й перцентиль тестовой группы: 179.50
95%-й доверительный интервал разности 75-х перцентилей [11.00, 30.00]
-----------------------------------------------------------------------------------------------------------
```

Вывод:

> *Пользователи, которые раньше вносили небольшие суммы, могли стать более осторожными, сократить свои вложения или совсем перестать платить. Это отразится в снижении 25-го перцентиля суммы депозитов в тестовой группе.*

Фича статистически значимо снизила суммы депозитов в тестовой группе. С 95%-ой вероятностью пользователи, инвестирующие небольшие суммы (до 95 у.е.) под влиянием новой фичи изменят поведение и уменьшат вносимые депозиты на 44-50 у.е.

> *Пользователи, которые склонны к более крупным инвестициям, могли сильнее вовлечься в продукт и начать вносить больше средств. Это отразится в росте 75-го перцентиля.*

Фича статистически значимо увеличила суммы депозитов в тестовой группе. С 95%-ой вероятностью пользователи, инвестирующие более крупные суммы (до 159 у.е.) под влиянием новой фичи изменят поведение в сторону активных инвестиций и увеличат объемы на 11-30 у.е.

Поведение типичных пользователей инвестиционного продукта в результате фичи также значимо скорректируется. Снижение инвестиций, деление их на более мелкие и последующее постепенное инвестирование, или отказ от финансовых инвестиций в пользу материальных. Мы наблюдаем снижение 50% перцентиля в тестовой группе. с 95%-ой вероятностью типичные пользователи снизят инвестиции на 19-27 у.е. относительно 118 у.е.

Выводы

1. В рамках исследования проверялась гипотеза о влиянии нового онбординга на поведение и активность пользователей - начинающих инвесторов.
2. В эксперименте участвовали пользователи из 4 стран - Аргентины, Бразилии, Колумбии и Мексики, которые зарегистрировались в приложении со 2 по 15 июня 2025 года включительно - это две полные недели с понедельника по воскресенье.
3. Приложение доступно на двух платформах - web и mobile
4. Пользователи приобретали активы разного уровня риска
5. Всего в эксперименте участвовало 9415 пользователей, которые были распределены на тестовую (test) и контрольную (control) группы.

Результаты:

1. **Пользователи**

Пользователи равномерно распределены по странам и типам платформ. Пересечений между группами нет. НО между ними существует статистически значимая разница - контрольная группа на 279 пользователей больше (процентная разница 5,8%)

Риски:

- Дисбаланс в распределении пользователей на группы может существенно повлиять на ключевую метрику, если в одной из групп случайно окажется больше пользователей склонных к более существенным вложениям.

2. **Пользовательский путь**

В целом тестовая группа демонстрирует значительное улучшение общей конверсии 21.3% по сравнению с контрольной - 13.5%. Однако она серьезно проигрывает на этапе онбординга 75.8% против 83.8% для контрольной группы.

Риски:

- Возможен отток пользователей не завершивших онбординг из-за сложности новой информации.

Плюсы:

- Онбординг является своего рода фильтром, который "оставляет" в продукте наиболее замотивированных пользователей. Это подтверждается высокими показателями конверсий на следующих этапах.
- Онбординг ограждает консервативных и осторожных инвесторов от возможных потерь, что в свою очередь вселяет в них мысль о том, что продукт "заботится о своих клиентах!"

Что делать:

- Проверить наличие багов, например, проблема с кнопкой "Начать онбординг" или долгая загрузка.
- Рассмотреть варианты бонусов за прохождение онбординга до конца, что позволит простимулировать сомневающихся в своих силах пользоватлей.

3. **Ключевая метрика**

Средняя сумма всех депозитов на одного пользователя в тестовой группе выросла до 92.46 у.е. по сравнению с контрольной - 90.24 у.е. Это может говорит о том, что пользователи на онбординге получают досточно знаний для более существенных вложений в инвестиции. Однако это разница статистически не значима!

Причины:

- Вероятно последствия дисбаланса в распределнии пользователей на группы.
- Возможно недостаточный размер групп или времени, и как следствие низкая мощность не позволяющая зафиксировать такой эффект.
- Для пользователей тестовой группы начало инвестиций сдвигается из-за прохождения онбординга. Поэтому с учетом периода распределения пользователей по группам, тестовая группа опережает контрольную групп с 18 июня - вероятно наблюдаем эффект от онбординга.

4. **Барьерная метрика**

Конверсия из регистрации в первый депозит в тестовой группе меньше 70%, чем в контрольной 71.5%. Однако эта разница статистически не значима, то есть конверсии тестовых групп равны.

Гипотеза о том, что информация о возможных потерях и высоких рисках отпугнёт некоторых новичков, что снизит конверсию в первый депозит не подтверждается.

Анализ метрики по сегментам:

- Конверсия из регистрации в первый депозит снижена на обеих платформах, однако для web-версии падение более существенно - относительная разница для web составляет 3.4%, для mobile 1.6%.
- Не все страны показывают падение метрики. Например, для Аргентины конверсия из регистрации в первый депозит выросла на 3.2% п.п., остальные страны показывают падение, при этом наибольшее падение метрики наблюдается для Мексики - почти 5% п.п.

5. **Влияние категории риска во второй депозит**

Пользователи выбравшие высокорисковые финансовые инструменты оказались наиболее чувствительны к изменениям онбординга. Для них наблюдается статистически значимое улучшение конверсии во второй депозит с 27.7% до 46%.

Гипотеза об открытии второго депозита после нового онбординга подтверждается.

6. **Вспомогательная метрика 1**

Конверсия из первого депозита во второй в тестовой группе увеличилась до 32.61% по сравнению с контрольной группой - 20.26% . При этом тест показал статистическую значимость этих улучшений.

Анализ метрики по сегментам:

- Все платформы показывают увеличение конверсии, относительное значение прироста 56.4% для web и 63% для mobile.
- Все страны показывают увеличение конверсии отнсительное значение прироста от 42% для Колумбии до 71% для Мексики.

Гипотеза о положительном влиянии онбординга, о том что пользователи чаще будут открывать второй депозит - подтверждается!

7. **Вспомогательная метрика 2**

Средняя сумма всех депозитов на пользователя, открывшего хотя бы один депозит, в тестовой группе статистически значимо больше на 4.4%, чем в контрольной группе.

Под влиянием новой фичи пользователи тестовой группы меняют поведение и статистически значимо уменьшают вносимые депозиты на 44-50 у.е.

Пользователи, инвестирующие более крупные суммы (до 159 у.е.) под влиянием новой фичи меняют поведение в сторону активных инвестиций и увеличивают объемы на 11-30 у.е.

Поведение типичных пользователей инвестиционного продукта в результате фичи также значимо скорректируется - снижением инвестиции на 19-27 у.е.