Аналитика дейтов

🚀 Endpoint

```
POST https://functions.yandexcloud.net/d4e5gh8buua0mcs5sqh4
```

📌 Назначение

Функция принимает **серии дневниковых записей о свиданиях** (текст ± голос (пока заглушка)) и возвращает единый Markdown-отчёт:

1. Итог (сводка состояния).
2. Позитивные / негативные изменения.
3. Конкретные советы next-steps.
4. Инсайты (частые эмоции, темы, метрики).
5. *Опционально* — готовое «Сообщение для собеседника», если передан `feedback_to_match`.

📩 Пример запроса (пять объёмных записей)

```
curl -X POST https://functions.yandexcloud.net/d4e5gh8buua0mcs5sqh4 \
  -H "Content-Type: application/json" \
  -d '{
      "user_id": "demo-user-001",
      "entries": [
        {
          "date": "2025-07-14",
          "date_title": "Долгая прогулка по набережной",
          "journal_text": "Мы встретились около пяти вечера и почти три часа гуляли вдоль реки...",
          "emotions": ["счастье", "радость", "спокойствие"]
        },
        {
```

```
        "date": "2025-07-15",
        "date_title": "Кофейня с разговором о будущем",
        "journal_text": "Разговор шёл чуть натянуто, я задавал открытые во
просы...",
        "emotions": ["любопытство", "неуверенность"]
      },
      {
        "date": "2025-07-16",
        "date_title": "Пикник на холме",
        "journal_text": "Она принесла домашний пирог, наблюдали закат...",
        "emotions": ["тепло", "благодарность", "робость"]
      },
      {
        "date": "2025-07-17",
        "date_title": "Кино и расхождение во вкусах",
        "journal_text": "Фильм оказался тяжёлым, после сеанса возникло н
апряжение...",
        "emotions": ["напряжение", "честность"],
        "feedback_to_match": {
          "liked": ["открытость в обсуждении чувств", "честность"],
          "not_liked": ["резкая смена настроения после фильма"],
          "feedback_style": "kind_honest"
        }
      },
      {
        "date": "2025-07-18",
        "date_title": "День тишины и восстановления",
        "journal_text": "Медитировал дома, слушал подкаст о самоценност
и...",
        "emotions": ["спокойствие", "восстановление"]
      }
    ]
  }'
```

📩 Пример ответа

{
"report": "## Итог\nВы стали чаще описывать чувства прямо и отмечать моменты радости...\n\n### Позитивные изменения\n• Увеличилось чувство спокойствия на свиданиях\n• Начали открыто обсуждать эмоции партнёра\n\n### Негативные изменения\n• Всё ещё появляется неуверенность, когда общение идёт вяло\n\n### Советы\n• Перед встречей подготовьте 2-3 темы, чтобы снять тревогу...\n• После сложных фильмов предлагайте лёгкую активность для разрядки...\n\n### Инсайты\n• Частые эмоции: счастье, спокойствие, неуверенность\n• Частые темы: прогулки, честность, границы\n• Записей за 30 дней: 5\n\n### Сообщение для собеседника\n«Спасибо за открытость! Мне понравилась твоя честность...»"
}

> Поле report содержит готовый Markdown-текст — можно сразу показывать в приложении / веб-демо.

⚙️ Параметры

Поле	Тип / Формат	Описание
`user_id`	string	Идентификатор пользователя
`entries`	array	Список записей (≥1)
`date`	YYYY-MM-DD	Дата события
`date_title`	string	Короткий заголовок записи
`journal_text`	string	Текст заметки (до нескольких абзацев)
`voice_note_url`	string (opt)	URL аудио — пока возвращается заглушка-подпись
`emotions`	array	Субъективные эмоции (слова)
`feedback_to_match`	object (opt)	Блок фидбэка для собеседника: • `liked` / `not_liked` — массивы строк • `feedback_style` — `kind_honest` / `playful` / `reflective`

```
import os, re, json
from datetime import datetime, timedelta
from typing import List, Dict
from yandex_cloud_ml_sdk import YCloudML

sdk = YCloudML(folder_id=YC_FOLDER_ID, auth=YC_API_KEY)


# ── helpers ──────────────────────────────────────────────────────────────
def preprocess_entries(raw: List[Dict]) → List[Dict]:
    """Гарантируем, что у каждой записи есть поля и сортируем по дате."""
    for e in raw:
        e.setdefault("journal_text", "")
        e.setdefault("voice_text", "")
        e.setdefault("emotions", [])
    return sorted(raw, key=lambda x: x["date"])


def calc_trends(entries: List[Dict]) → Dict:
    """Частые эмоции / темы за 3 последних записи и метрики за 30 дней."""
    recent = entries[-3:]
    emotions = {emo for e in recent for emo in e.get("emotions", [])}

    themes = set()
    for e in recent:
        txt = (e.get("journal_text", "") + e.get("voice_text", "")).lower()
        themes.update(
            w for w in re.findall(r"[а-яё]{5,}", txt)
        )

    last_30_cut = (datetime.utcnow() - timedelta(days=30)).date().isoformat()
    last_30_cnt = sum(1 for e in entries if e["date"] >= last_30_cut)

    return {
        "recent_cnt": len(recent),
```

```
        "freq_emotions": ", ".join(sorted(emotions)) or "—",
        "freq_themes": ", ".join(sorted(themes)) or "—",
        "cnt_30d": last_30_cnt
    }


# ── prompt builder ──────────────────────────────────────────────────────
def build_batch_prompt(user_id: str, entries: List[Dict]) -> str:
    trends = calc_trends(entries)

    # краткий список всех записей
    bullets = []
    for e in entries:
        fb = e.get("feedback_to_match")
        fb_tag = (
            f" • feedback: liked {', '.join(fb.get('liked', []))}; "
            f"not liked {', '.join(fb.get('not_liked', []))}"
            if fb else ""
        )
        bullets.append(
            f"- {e['date']} | {e['date_title']} | emotions: "
            f"{', '.join(e['emotions']) or '—'}{fb_tag}\n  "
            f"{e['journal_text'][:120]}..."
        )
    entries_block = "\n".join(bullets)

    return f"""
Ты – AI-коуч по знакомствам. На основе серии записей сформируй единый с

## Входные данные
Пользователь: {user_id}
Записей получено: {len(entries)}

### Перечень записей
{entries_block}
```

```
### Агрегированные тренды
- Частые эмоции: {trends['freq_emotions']}
- Частые темы: {trends['freq_themes']}
- Записей за 30 дней: {trends['cnt_30d']}

### Требования к отчёту. Он должен быть обширным, каждый пункт не мене
1. **Итог** (2-3 предложения).
2. **Позитивные изменения** — маркированный список.
3. **Негативные изменения** — маркированный список.
4. **Советы** — конкретные next-steps.
5. **Инсайты** — метрики / наблюдения.
""".strip()


# ── GPT call ──────────────────────────────────────────────────────────────
def call_gpt(prompt: str) → str:
    chat = [{"role": "user", "text": prompt}]
    res  = sdk.models.completions("yandexgpt").configure(temperature=0.7).run(
    txt  = res[0].text.strip()

    # убираем возможные ```markdown```-ограничители
    if txt.startswith("```"):
        txt = re.sub(r"^```[a-z]*", "", txt, flags=re.I).strip()
        txt = re.sub(r"```$", "", txt).strip()
    return txt
```