Генератор челленджей

🚀 Endpoint

```
https://functions.yandexcloud.net/d4enjsb4p211q7r6tv1n
```

📌 Назначение

Функция принимает **все записи дневника** за выбранный период и возвращает **3-5 персональных челленджей** (мини-заданий) в формате SMART, которые помогают:

1. Прокачать слабые стороны (частые негатив-эмоции или темы).
2. Развить новые навыки коммуникации.
3. Сделать свидания более осознанными и интересными.

📩 Пример запроса

```
curl -X POST https://functions.yandexcloud.net/d4enjsb4p211q7r6tv1n \
  -H "Content-Type: application/json" \
  -d '{
      "user_id": "demo-user-001",
      "entries": [
        {
          "date": "2025-07-12",
          "date_title": "Неловкий ужин",
          "journal_text": "Задавал слишком много вопросов, чтобы заполнить тишину...",
          "emotions": ["тревога", "усталость"]
        },
        {
          "date": "2025-07-13",
```

```
        "date_title": "Кофейня",
        "journal_text": "Разговор шёл чуть натянуто...",
        "emotions": ["неуверенность"]
      },
      {
        "date": "2025-07-14",
        "date_title": "Пикник",
        "journal_text": "Сиделина пледе, наблюдали закат...",
        "emotions": ["благодарность", "робость"]
      },
      {
        "date": "2025-07-15",
        "date_title": "Час откровенных вопросов",
        "journal_text": "Честно обсудили страхи...",
        "emotions": ["смущение", "близость"],
        "feedback_to_match": {
          "liked": ["открытость", "глубина"],
          "not_liked": ["неловкое начало"],
          "feedback_style": "kind_honest"
        }
      },
      {
        "date": "2025-07-16",
        "date_title": "День рефлексии",
        "journal_text": "Перечитал заметки, понял, что боюсь тишины...",
        "emotions": ["осознанность", "решимость"]
      }
    ]
  }'
```

📩 Пример ответа

```
{
  "challenges": [
    {
```

```
      "title": "Минутная пауза",
      "description": "На следующем свидании выдержи молчание 60 секунд после её ответа и улыбнись, прежде чем продолжить.",
      "goal": "снизить тревогу перед тишиной",
      "difficulty": "medium"
    },
    {
      "title": "Честный радар",
      "description": "В течение недели после каждой встречи записывай, что ты действительно почувствовал, не фильтруя эмоции.",
      "goal": "повысить осознанность и честность с собой",
      "difficulty": "easy"
    },
    {
      "title": "Вопрос дня",
      "description": "Перед свиданием подготовь 3 открытых вопроса, касающихся ценностей партнёра, и задай минимум один.",
      "goal": "улучшить глубину диалога",
      "difficulty": "medium"
    }
  ]
}
```


Каждый объект челленджа содержит:

- `title` — короткое название.
- `description` — конкретное действие (1–2 предложения).
- `goal` — какую зону/навык развивает.
- `difficulty` — `easy` | `medium` | `hard`.

⚙️ Параметры

Поле	Тип / Формат	Описание
`user_id`	string	Идентификатор пользователя

Поле	Тип / Формат	Описание
`entries`	array	Список всех записей дневника (≥ 1)
`date`	YYYY-MM-DD	Дата события
`date_title`	string	Заголовок записи
`journal_text`	string	Текст заметки (до нескольких абзацев)
`voice_note_url`	string (opt)	URL аудио (пока заглушка-подпись)
`emotions`	array	Субъективные эмоции
`feedback_to_match`	object (opt)	Блок фидбэка: `liked`, `not_liked`, `feedback_style`

```
import os, re, json
from datetime import datetime, timedelta, UTC
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]:
    """Добавляем дефолты 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_negatives(entries: List[Dict]) -> Dict:
    """Возвращаем наиболее частые негатив-эмоции и темы."""
    negative_pool = {"тревога", "неуверенность", "разочарование", "усталость"
                     "напряжение", "сожаление", "неловкость"}
    emo_counter = {}
    topic_set   = set()
```

```
    for e in entries:
        for emo in e.get("emotions", []):
            if emo in negative_pool:
                emo_counter[emo] = emo_counter.get(emo, 0) + 1
        txt = (e["journal_text"] + e["voice_text"]).lower()
        topic_set.update(re.findall(r"[а-яё]{5,}", txt))

    top_neg = sorted(emo_counter, key=emo_counter.get, reverse=True)[:3]
    return {"neg_emotions": ", ".join(top_neg) or "—",
            "topics": ", ".join(sorted(topic_set)) or "—"}


# ── prompt builder ──────────────────────────────────────────────────
def build_challenge_prompt(user_id: str, entries: List[Dict]) → str:
    stats = calc_negatives(entries)
    return f"""
Ты – relationship-coach-бот. Проанализируй дневник пользователя и предло›
SMART-челленджей, которые помогут прокачать слабые стороны.

Наблюдаемые частые негативные эмоции: {stats['neg_emotions']}
Популярные темы дневника: {stats['topics']}

Сформируй JSON-массив объектов:
[
  {{
    "title": "короткое название",
    "description": "что именно сделать (1-2 предложения)",
    "goal": "какую навык/зону улучшает",
    "difficulty": "easy|medium|hard"
  }},
  ...
]
Без дополнительного текста.
""".strip()
```

```
# ── GPT call ──────────────────────────────────────────────────────────────
def call_gpt(prompt: str) → List[Dict]:
    chat = [{"role": "user", "text": prompt}]
    res  = sdk.models.completions("yandexgpt").configure(temperature=0.6).run(
    txt  = res[0].text.strip()

    # убираем ```json ... ``` если модель вернёт в блоке
    if txt.startswith("```"):
        txt = re.sub(r"^```[a-z]*", "", txt, flags=re.I).strip()
        txt = re.sub(r"```$", "", txt).strip()

    return json.loads(txt)
```