

Интеграции

- [Отчёты по встречам Контур.Толка: contur_transcribe](#)
- [YouGile и GitHub: yogile_connector](#)

Отчёты по встречам

Контур.Толка:

contur_transcribe

Для разработчиков. Состояние на 28.09.2026. Переменные, песочница и выкатка по шагам — [README](#) репозитория `Buharestskie/contur_transcribe`. На сервере проект развёрнут под именем `donu1a-kontur-reporter`.

Коротко

`contur_transcribe` — демон без интерфейса и без входящих запросов. Раз в пять минут он забирает новые записи встреч из пространства Контур.Толка, получает расшифровку, делает управленческий разбор языковой моделью и раскладывает результат: короткий отчёт и файл расшифровки — в закрытую группу Telegram, структурированные поля — в рабочую таблицу Google Sheets отдела продаж. Задача — чтобы итоги тет-а-тетов и общих встреч РОПов доходили до таблицы без ручной работы.

Как устроен один проход

1. **Список записей.** Один ключ пространства (`KTALK_SPACE_API_KEY`) читает записи всех РОПов за последние `LOOKBACK_DAYS` дней (по умолчанию 7). Уже опубликованные и тестовые (по шаблонам названий) пропускаются.
2. **Расшифровка.** Сначала берётся готовая расшифровка Толка. Если текста в ней нет, запись скачивается в качестве 240p, из неё вытаскивается звук (ffmpeg) и отправляется в сервис распознавания а5х; задача опрашивается до готовности (до часа). Если у Толка запись ещё не готова, встреча откладывается до следующего прохода — это не ошибка.
3. **Разбор.** Диалог уходит в языковую модель а5х (`A5X_MODEL`). Она возвращает короткий управленческий радар: план и факт отдельно по каждому сотруднику, одна строка по каждому участнику, до трёх главных проблем, все поставленные задачи с ответственными и сроками (до 12), где нужен руководитель, оценка полезности встречи 0–10. Отсутствующее помечается «Не зафиксировано». Длинные встречи разбираются через конспекты частей. Если модель недоступна,

разбор собирается из протокола Толка.

4. **Маршрутизация.** Если участников два-три, однозначно опознан один сотрудник и на встрече был его РОП — это тет-а-тет: заполняется строка сотрудника на листе этого РОПа. Всё остальное — на лист «Общие встречи». Полная расшифровка в любом случае пишется на скрытый лист «Расшифровки Контур.Толк».
5. **Telegram.** Отчёт одним сообщением (длинный — разбивается на части) и расшифровка отдельным `.txt`.
6. **Метрики.** В конце прохода счётчики `contur_transcribe_cycle_*` (найдено, опубликовано, отложено, упало, пропущено) уходят в Pushgateway мониторинга.

Каждый выполненный шаг сразу отмечается в `data/state.json`. Сбой посередине не теряет встречу и не публикует её дважды: следующий проход доделает только оставшееся.

Кто есть кто

Сотрудники и РОПы описаны в двух справочниках вне кода: `config/rops.json` (РОП и название его листа) и `config/employees.json` (сотрудник, его РОП и признаки опознания: `emails`, `konturUserKeys`, `aliases`, `roomNames`, `titlePatterns`). Образцы — `*.example.json`. Заполнять руками не нужно: `python -m tools.sync_identities` показывает, что изменится, с `--write` — дописывает каждому найденному человеку ключ, почту и форму имени из справочника пространства.

Мониторинг

Проба снаружи сервису бессмысленна: у него нет адреса. Вместо неё два правила в `prometheus/rules/contur_transcribe.yml` репозитория `infra`:

Алерт	Когда	Обычная причина
<code>ConturTranscribeStalled</code>	30 минут подряд: встречи находятся, часть падает, опубликовано ноль	отказ распознавания или модели а5х
<code>ConturTranscribeSilent</code>	проходов не было больше 30 минут	контейнер упал, завис или не поднялся

Метрики уходят, только если задан `METRICS_PUSH_URL` (с ним — `METRICS_PUSH_USER`, `METRICS_PUSH_PASSWORD`, `METRICS_INSTANCE`); без адреса отправка молча выключена. Здоровое состояние без новых встреч выглядит как «найдено 59, пропущено 59, опубликовано 0» — пропущенные в основном уже опубликованы раньше.

Где код и как запустить

- Репозиторий `Buharestskie/contur_transcribe`, Python 3.12, httpx, pydantic. Команды: `python -m app.cli check` — проверить все доступы, ничего не публикуя; `once` — один боевой проход (есть `--limit` и `--only`); `daemon` — бесконечный цикл, так работает контейнер.
- **Песочница:** `tools.create_sandbox` создаёт тестовую книгу, `tools.find_chat` находит чат тестового бота. Значения кладутся в `.env.sandbox` со своим `STATE_FILE`, команды запускаются с `--env-file .env.sandbox` — рабочая таблица и группа отдела не задеваются.
- Тесты — `pytest`, внешние сервисы замканы.
- **Выкатка:** на сервер доставляются `app/`, `pyproject.toml`, `Dockerfile`, `docker-compose.yml`. Серверные `.env`, `secrets/`, `data/` и справочники не перезаписываются. Порядок: резервная копия `data/state.json` → `docker compose build reporter` → `check` и `once` разовым контейнером → `docker compose up -d reporter` → логи.

На что наступают

- `data/state.json` **нельзя удалять или обнулять**. Пустой файл — повторная рассылка всех встреч за `LOOKBACK_DAYS` дней в группу и в таблицу. Перед любыми работами — копия.
- **Не запускайте `once` поверх работающего демона**. Второй процесс упадёт с «Предыдущий запуск ещё работает»: состояние защищено блокировкой.
- **Имена сотрудников в Толке и в справочнике расходятся**. Большинство заходит гостями и вписывает имя как придётся, иногда латиницей. Одно слово имени опознаёт человека, только если оно в отделе у одного. При двух Аннах «анна» бесполезна, и встреча уходит в общие. После любого изменения состава отдела прогоняйте `tools.sync_identities`.
- `titlePatterns` **пишите по основе фамилии**, а не по полной форме: шаблон с «никитина» не совпадёт с «Никитиной».
- **Листы РОПов сервис не создаёт**. Нет листа из `rops.json` — `check` сообщит об этом. «Общие встречи» и лист расшифровок создаются сами. Лист «Контроль РОПа+» считается формулами книги: после новых строк и месяцев проверьте, что диапазоны формул их охватывают.
- **Файл для распознавания ограничен 110 МБ** (`A5X_MAX_UPLOAD_MB`, сервис a5x отказывает около 128 МБ). Без ffmpeg запись уходит видеофайлом и может не пройти.
- **Сервер общий с другими боевыми контейнерами**, лимиты памяти и CPU в `docker-compose.yml` занижены намеренно.
- **Три имени у одного проекта:** репозиторий `contur_transcribe`, на сервере `donula-kontur-reporter`, локальная папка часто `kontur-talk-reporter`. Ищите по всем трём.

YouGile и GitHub: yogile_connector

Для разработчиков. Состояние на 28.09.2026. Переменные и все ручки — README
репозитория `Buharestskie/yogile_connector`.

Коротко

`yogile_connector` связывает задачи YouGile с ветками GitHub и двигает задачу по доске вслед за кодом: ветки нет — задача ждёт ветку; ветка появилась — задача в работе; ветка слита в основную — задача закрыта; после слияния пришли новые коммиты — задача снова в работе. Внутри YouGile скрипт-витрина показывает состояние бейджем на карточке и блоком «GitHub» в открытой задаче. Сервис доступен по адресу `https://dev-tasks.donula26.ru`, настройки — в админ-панели `/admin`.

Как задача находит свои ветки

- **По ID задачи в имени ветки.** Номер задачи из YouGile плюс префикс проекта дают ID вида `PRI-18`. Любая ветка, в имени которой есть этот токен (`feat/PRI-18-cart`, `pri_18`), считается веткой задачи — в каком бы репозитории организации она ни появилась.
- **Веток может быть несколько**, по одной в разных репозиториях. Каждая ведётся отдельно, общее состояние собирается из них: пока не слита хоть одна — задача в работе, слиты все — закрыта.
- **«Слито»** — у ветки нет коммитов вне основной (`compare`, `ahead_by == 0`), и при этом собственные коммиты на ней хоть раз были. Основная ветка у каждого репозитория своя: у части проектов это `dev`, а не `main`.
- **Стикер «Репозиторий»** на задаче выбирает один репозиторий — тот, где ветка создаётся автоматически. Задачи без стикера и без привязанных репозитория сервис не двигает (состояние «репозиторий не выбран»). Стикер сервис заводит и поддерживает сам: вешает на включённые доски, дописывает новые репозитории организации, убирает исчезнувшие.

Что сервис делает сам

Событие	Что происходит
Задачу перенесли в рабочую колонку	Создаётся ветка в репозитории из стикера (если на доске включено автосоздание)
Для задачи появилась ветка	Задача переезжает в рабочую колонку
Все ветки слиты	Задача уезжает в «Готово» и отмечается выполненной
После слияния пришли новые коммиты	Задача возвращается в работу
Задача ушла из рабочей колонки	Пустая ветка (без своих коммитов) удаляется — кроме колонок, где ветки велено сохранять
Задача закрыта	Слитая ветка удаляется. Неслитая не удаляется никогда — в чат задачи один раз приходит сообщение с её именем и числом неслитых коммитов

Автосоздание ветки на доске по умолчанию выключено, удаление пустых и слитых веток — включено; всё это переключается по доскам в админ-панели. Каждый переход пишется в чат задачи, если задан `YOUGILE_BOT_USER_ID`. Цвет карточки меняется под состояние: серый — нет репозитория или ветки, зелёный — слито.

Откуда события

- **GitHub** → `POST /webhooks/github`: события `push`, `create`, `delete`, `pull_request`. Подпись проверяется по `GITHUB_WEBHOOK_SECRET`, повторные доставки отбрасываются. Push в основную ветку пересчитывает все задачи досок этого репозитория.
- **YouGile** → `POST /webhooks/yougile`: события `task-created` и `task-moved`.
- **Плановая сверка** всей доски раз в `RECONCILE_INTERVAL_MINUTES` (по умолчанию 15) — страховка на случай потерянного вебхука.
- **Витрина** (`yougile-script/github-widget.js`) раз в 30 секунд читает `GET /api/board/state` и ничего не решает сама. Ответ кешируется на 30 секунд.

Админ-панель

Настройки живут в базе (SQLite в `./data` по умолчанию) и правятся на `/admin`: репозиторий и ключи по умолчанию, параметры YouGile, таблица досок — какие синхронизировать, какой репозиторий у каждой, какие колонки рабочая и завершающая, создавать ли ветку автоматически. При первом запуске значения переносятся из окружения, дальше окружение нужно только для базы и токенов входа. Кнопки панели ходят в `/admin/api/*` с заголовком `X-Admin-Token`:

Ручка	Что делает
<code>POST /admin/api/reconcile</code>	Полная сверка досок
<code>POST /admin/api/webhooks/yougile</code>	Создать подписки YouGile, тело <code>{"public_url": "https://dev-tasks.donula26.ru"}</code>
<code>POST /admin/api/webhooks/github</code>	Завести вебхук сервиса во всех репозиториях (или в переданном списке <code>repos</code>)
<code>POST /admin/api/reload</code>	Перечитать настройки и структуру досок
<code>GET /admin/api/status</code>	Сводка состояния

Проверка живости — `GET /healthz`, отвечает `{"status": "ok"}`.

Где код и как выкатывается

- Репозиторий `Buharestskie/yogile_connector`: FastAPI, SQLAlchemy, httpx. Локально — `uvicorn app.main:app --reload --port 8000` с заполненным `.env`. Вебхуки на ноутбук не придут — пересчёт вызывайте через `POST /admin/api/reconcile`.
- На сервере — git-клон: `git pull`, затем `docker compose -f docker-compose.yml -f docker-compose.prod.yml up -d --build`. Прод-файл — надстройка: убирает публикацию порта и подключает контейнер к сети внешнего прокси. Собранный без неё контейнер опубликует порт наружу и потеряет прокси.
- У сервиса есть внешняя проба «Связь с YouGile» (на статус-странице скрыта) — выкатка в окне обслуживания.
- Витрина ставится внутри YouGile** отдельно: конфигуратор скриптов (`Ctrl + ~`, только у администратора компании) → «добавить скрипт». Проще всего взять готовый текст с `/widget/script.js?token=...` — адрес сервиса и токен в нём уже подставлены.

На что наступают

- Кириллица в скрипте YouGile ломается** при передаче в песочницу: регулярка с `[а-яё]` валит скрипт целиком, строки превращаются в мусор. Код витрины пишется только латиницей, все русские подписи отдаёт сервис вместе с состоянием.
- Витрина работает в песочнице без `window`** и не видит ID задачи вида `PRI-18` — поэтому сопоставлением занимается сервис.
- App.httpGet из API конфигуратора не годится для GitHub**: нет `User-Agent`, GitHub отвечает `403`. Работает обычный `fetch`. `UI.safeHtml` вырезает SVG — плашки рисуются через `UI.rawHtml` с ручным экранированием.
- Стикер к доске привязывается только своим**. Если отправить в `PUT /boards/{id}` весь набор стикеров, YouGile отвергнет запрос из-за текстовых стикеров.

- **Первое появление задачи ничего не двигает** — сервис только запоминает состояние. Переходы работают со второго наблюдения, иначе первый запуск перетасовал бы всю доску.
- **Токену GitHub нужна запись.** Для создания и удаления веток — `Contents: Read and write`; с токеном только на чтение GitHub отвечает `403`, сервис пишет об этом в чат задачи. Без токена лимит — 60 запросов в час, полная сверка съедает его быстро.
- **События на одну задачу приходят пачкой** (`create` и `push` почти одновременно), поэтому обработка задачи идёт под мьютексом. Не убирайте его — задача получала по три одинаковых сообщения.
- **Зеркальную колонку нельзя делать рабочей:** задача туда переезжает, но пропадает из виду своей доски.
- **Задача, унесённая на ненастроенную доску,** больше не двигается, но её состояние в базе остаётся.
- **YouGile периодически отвечает `502` / `503`.** Событие, упавшее на такой ошибке, доделает плановая сверка — не выключайте её (`RECONCILE_INTERVAL_MINUTES=0`).