ARSENY NN · AUTOMATION & AI SYSTEMS

Строю системы автоматизации для бизнеса

CRM-процессы, Telegram-боты, LLM-обработка текста, автогенерация документов, парсеры, API-интеграции и операционная аналитика.

Арсений Николаевич Р. / @arsenynn

Telegram: @arsenynnn

Email: arsenynickolaevich@yandex.ru

portfolio.arsenynn.com/it-ai

РЕЗУЛЬТАТЫ АВТОМАТИЗАЦИИ

Цифры, которые понятны бизнесу без чтения кода

КОММЕРЧЕСКИЕ ПРЕДЛОЖЕНИЯ

2–4 часа → 5 минут

Автогенерация КП из брифа, CRM-данных и шаблонов.

G20

ДОКУМЕНТЫ

1–2 дня → 5 минут

Договоры, счета и акты через Google Docs/PDF.

G20

ОТВЕТ НА ЛИД

до 10 минут

SLA-уведомления и единая обработка заявок.

G20

ЗАЯВКИ ЧЕРЕЗ БОТ/САЙТ

55%

Входящие заявки проходят через управляемый digital-flow.

G20

КОНТРОЛЬ ДУБЛЕЙ

100%

Каждая вакансия обрабатывается один раз по ID.

JOB AUTOMATION

ЧТО ДЕЛАЮ

Собираю рабочие системы, которые убирают ручную операционку

Автоматизация лидов, CRM, документов, отчётов, Telegram-ботов, LLM-обработки, парсеров, API/Webhook-интеграций и внутренних инструментов.

Подход: сначала бизнес-задача и измеримый результат, затем стек, архитектура и технические детали.

ПРОЕКТЫ

Кейсы автоматизации: от заявки и CRM до LLM-обработки и аналитики

01 · CRM + EVENT PRODUCTION

G20 — автоматизация техпродакшна мероприятий

Спроектировал и реализовал операционную систему для event-production на базе Google Workspace, Apps Script, Python и Telegram: CRM, калькулятор КП, генерация документов и счетов, календарное планирование, архив, клиентская база, интеграции Avito/SiteChat/EventRadar, Telegram-уведомления, мониторинг и AI-подбор оборудования.

ЗАДАЧА

Коммерческие предложения готовились вручную 2–4 часа, документы занимали 1–2 дня, часть данных копировалась вручную, а заявки могли теряться между сайтом, чатами и таблицами.

РЕШЕНИЕ

CRM-ядро на Google Sheets, Apps Script-автоматизация, Python Telegram-бот, генерация КП/счетов/документов, календарное планирование, архив, клиентская база, интеграции Avito/SiteChat/EventRadar, мониторинг и AI-подбор оборудования.

КАК РАБОТАЕТ

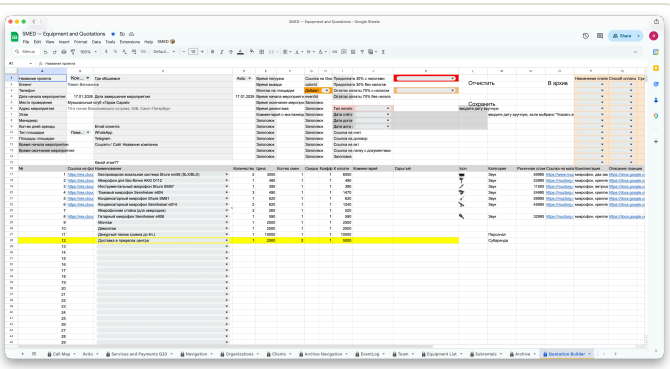
Заявка с сайта, Avito, бота или EventRadar попадает в Telegram/CRM → менеджер собирает смету в Quotation Builder → система считает стоимость, прибыль и расходы → генерирует КП/счёт/PDF → создаёт календарные события → отправляет уведомления → сохраняет архив и аналитику.

РОЛЬ

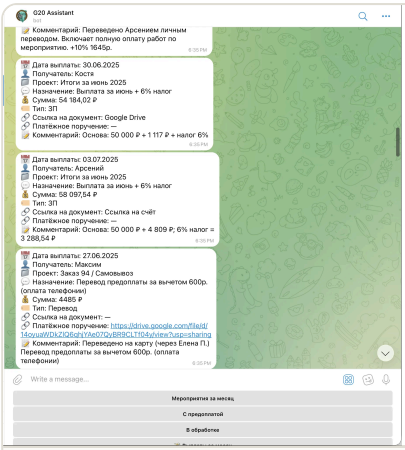
Архитектура системы, CRM, Telegram-бот, Apps Script-автоматизация, генерация документов, аналитика и интеграции.

РЕЗУЛЬТАТЫ

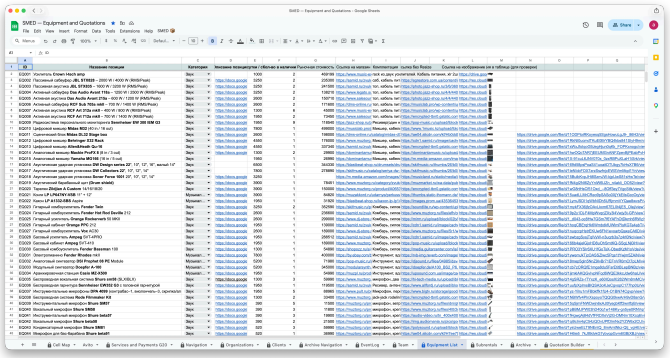
- Коммерческие предложения: 2–4 часа → 5 минут
- Документы: 1–2 дня → 5 минут
- Ответ на лид: до 10 минут
- Заявки через бот/сайт: 55%
- 20+ скриптов Apps Script для CRM, документов и уведомлений
- Меньше ошибок за счёт автозаполнения и валидаций



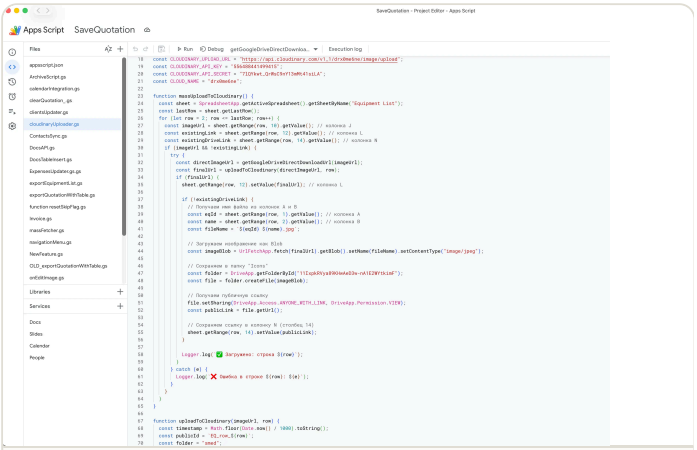
Конструктор КП: позиции, смены, коэффициенты, скидки и итоговая стоимость.



Telegram-бот для уведомлений, выплат, статусов проектов и операционного контроля.



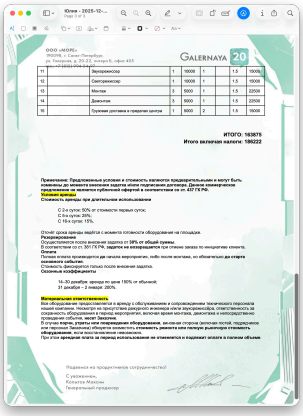
Справочник оборудования: категории, цены, рыночная стоимость, изображения и ссылки.



20+ Apps Script скриптов



PDF/Docs-коммерческое предложение с таблицей оборудования и автозаполненными данными.



Итоги, налоги, условия аренды и юридический блок в финальном документе.

ЧТО ВНУТРИ СИСТЕМЫ

Шесть модулей, которые превращают G20 в операционную систему

CRM & Project Core

- CRM-ядро на Google Sheets: проекты, клиенты, оборудование, навигация, архив, команда и расходы.
- Создание нового проекта из шаблона КП с автоматической нумерацией, формулами, стилями и служебными полями.
- Navigation-панель по активным проектам: статус, клиент, дата, площадка, суммы, прибыль и ссылка на проект.
- Архивирование проектов: перенос в Archive Navigation, скрытие листа и сохранение архивной копии.
- Синхронизация клиентов: client ID, поиск дублей по телефону, история проектов клиента.

Quotation Builder

- Конструктор коммерческих предложений в Google Sheets.
- Автозаполнение количества, смен, коэффициентов, скидок, стоимости и итогов.
- Расчёт по формуле: количество × цена × смены × коэффициент × скидка.
- Логика коэффициентов по количеству смен для оборудования и персонала.
- Equipment List: категории, цены, рыночная стоимость, описания, ссылки и изображения.
- Автодобавление новых позиций оборудования в справочник.

Finance, Documents & Payments

- Расчёт итогов, налогов, расходов, чистой прибыли, 25% и 50% от прибыли.
- Генерация коммерческого предложения в Google Docs по шаблону.
- Замена плейсхолдеров клиента, телефона, типа события, даты, площадки и адреса.
- Вставка таблицы оборудования и изображений в Google Docs через API.
- Добавление юридического блока из отдельного Google Docs-документа.
- Генерация счёта, PDF-экспорт и QR-код для оплаты.

Calendar, Archive & Operations

- Интеграция с Google Calendar: события создаются и обновляются из данных проекта.
- Два календаря: общий календарь событий и календарь подтверждённых/оплаченных проектов.
- Статус предоплаты управляет созданием события в Confirmed & Paid.
- Event ID сохраняется в таблицу, чтобы не создавать дубли.
- Telegram-отчёт по активным КП без предоплаты.
- Кастомное меню Google Sheets для быстрых действий и навигации.

Telegram Bot & Integrations

- Telegram-бот G20 Assistant на Python с модульным рантаймом g20bot.
- Интеграции Avito Messenger, site chat и EventRadar через webhook.
- Site chat валидирует payload, CORS и токен авторизации, затем отправляет лиды в Telegram.
- EventRadar-лиды валидируются, форматируются и отправляются в отдельный чат.
- Дедупликация лидов по chat/message ID.
- SQLite outbox: если Telegram/API недоступны, сообщения сохраняются и отправляются позже.

Reliability, AI & Monitoring

- Централизованная конфигурация через .env и JSON-логирование с маскированием секретов.
- Retry, timeout, rate limiting и circuit breaker для внешних API.
- Idempotency-хранилище, чтобы не обрабатывать Telegram update повторно.
- Prometheus /metrics, /healthz, heartbeat и Telegram-алерты администраторам.
- AI-помощник smed_assistant подбирает комплект оборудования по описанию мероприятия.
- Экспорт базы оборудования в JSON и отправка AI-подбора обратно в Google Sheets.

▼ Показать полный technical scope

- CRM-ядро на Google Sheets с листами для проектов, клиентов, оборудования, навигации, архива, команды и расходов.
- Финансовые расчёты: итог, налоги, расходы, чистая прибыль, 25% и 50% от чистой прибыли.
- Архивный экспорт КП в отдельный Google Sheets-файл и сохранение архивной копии на Google Drive.
- Генерация КП в Google Docs по шаблону с плейсхолдерами, таблицей оборудования, изображениями и юридическим блоком.
- Интеграция с Google Calendar: общий календарь событий и календарь Confirmed & Paid без дублей по Event ID.
- Конструктор КП с автозаполнением количества, смен, коэффициентов, скидок, стоимости и итогов.
- Создание проекта из шаблона с копированием диапазонов, формул, стилей, ширин колонок и служебных полей.
- Синхронизация клиентов, генерация client ID, поиск дублей по телефону и история проектов клиента.
- Генерация счёта, QR-код оплаты и PDF-экспорт.
- Кастомное меню Google Sheets, защита важных зон таблицы и очистка шаблонов без удаления формул.

- Google People API для подрядчиков, выпадающие списки и добавление новых подрядчиков в Google Contacts.
- Загрузка и обработка изображений оборудования: Pexels API, Drive, ресайз 300×300, Cloudinary и сохранение ссылок.
- Экспорт базы оборудования в JSON и AI-помощник smed_assistant на OpenAI API для подбора комплекта оборудования.
- Python-скрипты для создания листов проектов и отправки AI-подбора обратно в Quotation Builder.
- G20 Assistant на Python: модульный runtime, .env, JSON-логирование, retry, timeout, rate limiting и circuit breaker.
- SQLite outbox, idempotency-хранилище, нарезка длинных сообщений и fallback-доставка Telegram-уведомлений.
- Интеграции Avito Messenger, SiteChat и EventRadar через webhook с дедупликацией лидов.
- Prometheus /metrics, /healthz, heartbeat, latency, retries, circuit breaker state, outbox size и Telegram-алерты.
- Dockerfile и инструкции запуска через Docker, systemd и launchd.
- Легаси-бот: меню Telegram, Google Sheets, касса, выплаты, напоминания, Avito-опрос, дневные summary через Groq, OCR/ASR для аудио, PDF, DOCX и изображений.

Google Sheets | Apps Script | Python | Telegram Bot API | Google Docs API | Drive API | Calendar API | OpenAI API | Avito API | SQLite | Prometheus | REST/Webhooks/JSON

Portfolio repo: https://github.com/workflowarsenynn/event-crm-gworkspace-automation-portfolio

02 · PYTHON + AI

Автоматизация откликов на вакансии HeadHunter

Browser-first платформа на Python + Playwright для поиска вакансий, LLM-анализа JD, генерации персональных писем, заполнения анкет и controlled auto-submit с Telegram approval.

ЗАДАЧА

Ручной поиск, копирование описаний, выбор резюме и отклики занимали время, создавали дубли и плохо масштабировались. После ограничений API нужно было перенести автоматизацию в браузерный контур.

РЕШЕНИЕ

Система управляет Chromium через Playwright: ищет вакансии, извлекает описание, анализирует его через GPT, выбирает резюме, генерирует письмо, готовит отклик и отправляет его только через безопасный workflow. Простые вакансии отправляются автоматически, анкеты и нестандартные кейсы уходят в Telegram approval/handoff.

КАК РАБОТАЕТ

Search → fetch details → LLM classification → cover letter → prepare apply → safety check → auto-submit или Telegram approval → логирование → retry/status tracking.

РОЛЬ

Архитектура browser-first runtime, Playwright flow, LLM-промпты, safety layer, anti-duplicates, Telegram approval, retry queue, dashboard и regression smoke.

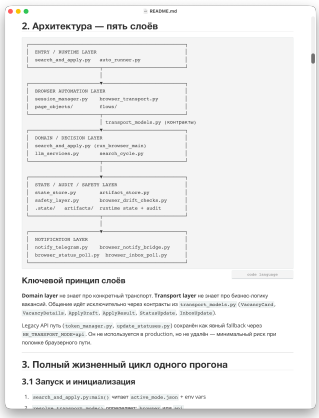
РЕЗУЛЬТАТЫ

- 100% контроль дублей по ID вакансии
- Controlled auto-submit для простых вакансий
- Telegram handoff для анкет и сложных кейсов
- Questionnaire engine для text, radio, checkbox, select
- Retry queue между сессиями и safety layer
- Trace artifacts, dashboard и regression smoke

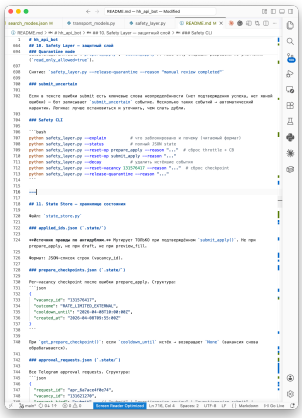
ПОДРОБНЫЙ РАЗБОР

Как и почему я строил browser-first автоотклики: архитектура, safety layer, handoff и ограничения.

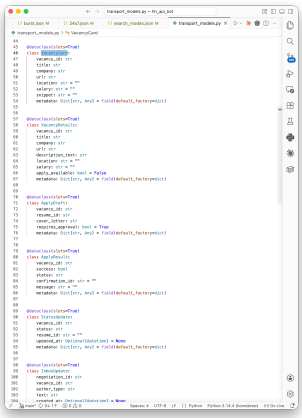
https://t.me/annsflow/136



Пятислойная архитектура: runtime, browser automation, decision layer, state/audit/safety и notification layer.

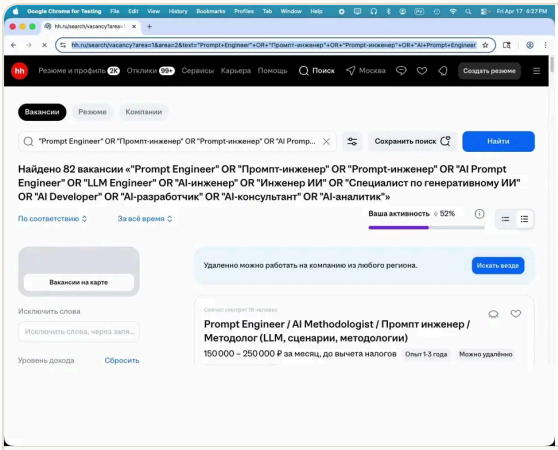


Safety Layer и CLI-восстановление: throttling, quarantine, submit_uncertain, reset операций и checkpoint recovery.



Типизированные transport contracts: единые dataclass-модели для вакансий, деталей, черновиков отклика, результатов submit, статусов и inbox-сообщений.

ВИДЕО PROOF



Демо browser-first процесса: система работает в браузере HeadHunter, готовит отклики и показывает controlled automation flow.

ЧТО ВНУТРИ СИСТЕМЫ

Ключевые модули и инженерные слои проекта

Browser Runtime

- Перевёл проект с API-first на browser-first архитектуру через Playwright и Chromium.
- Реализовал управление реальной HH-сессией через persistent Chromium profile.
- Добавил health-check авторизации, bootstrap-login и single-writer lock для browser profile.
- Построил browser transport adapter вместо прямого HH API transport.
- Сохранил доменную логику поиска, анализа, выбора резюме, генерации письма и логирования.

Apply Flow & Questionnaires

- Реализовал ApplyFlow как state machine: открыть страницу, выбрать резюме, вставить письмо, подготовить черновик, отправить отклик.
- Добавил классификацию страниц отклика: simple, questionnaire, already applied, fixed resume, unknown UI, auth-required.
- Сделал controlled auto-submit для простых вакансий без анкеты.
- Построил questionnaire engine для textarea, input, radio, checkbox, select и mixed-вопросов.
- Добавил LLM-ответы на анкеты с honest fallback logic и final approval.

Safety & Human-in-the-loop

- Финальный submit невозможен без approval_token=APPROVED.
- Режим auto_with_handoff отправляет простые вакансии автоматически, а сложные передаёт в Telegram approval.
- Telegram-кнопки Approve / Reject / Skip дают ручной контроль чувствительных действий.
- Safety layer включает throttling, circuit breakers, quarantine mode и cooldown.
- Система различает SKIP / NEEDS_HUMAN и реальные инфраструктурные ошибки.

State, Retry & Observability

- Антидубли между запусками через state-файлы и SQLite audit log.
- Retry queue сохраняет незавершённые вакансии между сессиями.
- Trace, screenshots и HTML artifacts сохраняются при ошибках browser runtime.
- Dashboard pending approvals показывает ручные кейсы в браузере.
- Ежедневная статистика и Telegram-отчёты строятся из SQLite application log.

LLM & Prompts

- LLM-классификация вакансий: APPLY / SKIP / MANUAL_REVIEW / DELAY.
- Генерация персонализированного cover letter под конкретный JD.
- Structured output для классификации и questionnaire drafts.
- Prompt injection guard через явные content boundaries для текста вакансии.
- Salary parsing hardening и fallback-логика для вопросов о зарплате.

Scheduler & Regression

- Auto runner работает как планировщик запусков через macOS LaunchAgents.
- Operational profiles: обычный 24x7 профиль и burst profile.
- Квоты по отправкам за запуск и за час защищают runtime от перегруза.
- Regression corpus покрывает apply-page variants и scheduler behavior.
- Preflight-проверки валидируют runtime, session health, safety state и active defaults.

▼ Показать полный technical scope

- Persistent Chromium profile, health-check авторизации, bootstrap-login и profile lock.
- Browser-based поиск вакансий, извлечение карточек и full vacancy details через DOM-локаторы.

- Domain/transport contracts: VacancyCard, VacancyDetails, ApplyDraft, ApplyResult, StatusUpdate, InboxUpdate.
- Controlled auto-submit, Telegram approval workflow, pending approvals resume и dedup approval requests.
- Retry queue, startup resume, attempt limits и escalation в needs_human.
- Artifacts: trace, screenshots, HTML, run summary, daily stats и Telegram reports.
- Operational profiles, runtime_profile_ctl, scheduler guard, quotas и burst mode.
- Atomic JSON-state writes, corrupt JSON backup, stale temp cleanup и synthetic hygiene.
- Production invariants: не обходить CAPTCHA, не делать submit без approval token, не мутировать applied IDs до подтверждённого submit.
- ApplyPage abstraction, PageVariant taxonomy и fallback-сценарии для нестабильного HH UI.
- Questionnaire engine: text, textarea, email, phone, number, radio, checkbox, select, mixed option + textarea.
- Safety layer: throttling, circuit breakers, quarantine mode, cooldown, bounded navigation и wall-clock guard.
- Browser pollers для статусов откликов, переписок и negotiation messages.
- OpenAI SDK v2, structured output, prompt injection guard, salary parsing hardening и fallback policies.
- Regression levels: preflight, draft smoke, approval/notify smoke, controlled submit regression.

Python Playwright Chromium OpenAI API SQLite Telegram Bot API pytest LaunchAgents JSON state

Portfolio repo: https://github.com/workflowarsenynn/job-application-automation-portfolio

03 · LLM + INTAKE

Avito Legal Bot для юридической компании

Система первичной обработки юридических заявок из Avito Messenger: сценарный диалог, LLM-анализ, FSM-контроль, SQLite-состояние и запись структурированного лида в Google Sheets.

ЗАДАЧА

Клиенты приходили из Avito сырыми сообщениями: данные были неполными, юристы тратили время на первичные вопросы, бот не должен был давать консультации или обещать результат.

РЕШЕНИЕ

Построен гибрид FSM + LLM: сценарий контролирует порядок и безопасность диалога, LLM анализирует сообщения, извлекает факты и формулирует живые ответы внутри guardrails. Результат сохраняется в SQLite и Google Sheets как структурированная карточка клиента.

КАК РАБОТАЕТ

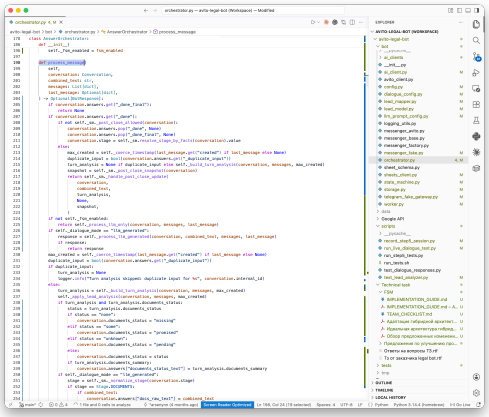
Avito message → batching/dedup → Turn Analysis через LLM → обновление фактов клиента → Stage Router/FSM → безопасный ответ → Lead Analysis → Google Sheets CRM → replay/tests для проверки поведения.

РОЛЬ

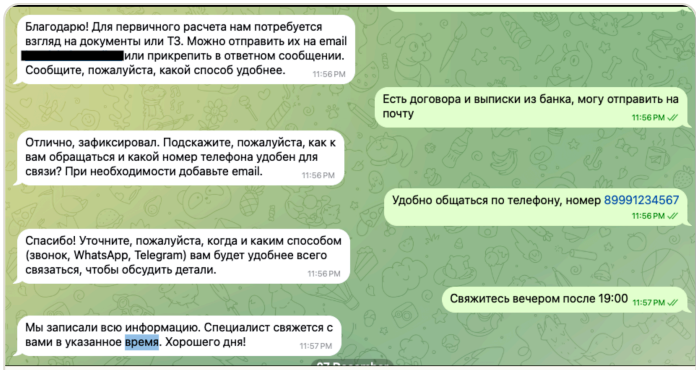
Архитектура диалога, FSM, LLM-orchestration, guardrails, storage, Avito/Sheets интеграции, тесты и документация.

РЕЗУЛЬТАТЫ

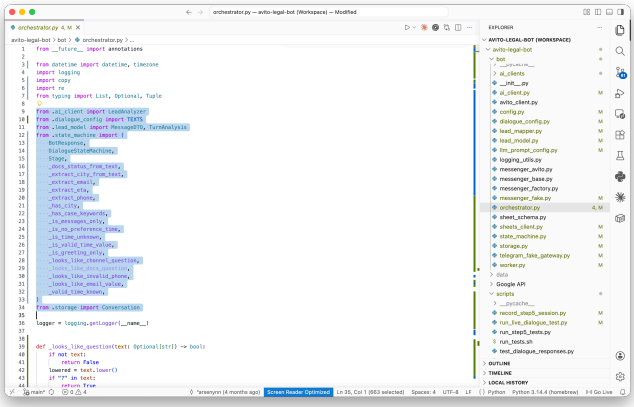
- Структурированная карточка лида вместо сырой переписки
- FSM не даёт боту пропускать обязательные этапы
- LLM отвечает живо, но не нарушает юридические guardrails
- SQLite сохраняет состояние и историю диалога
- Google Sheets получает один диалог как одну CRM-строку
- Replay, BDD и live-smoke тесты проверяют поведение



Runtime-flow обработки сообщения: состояние диалога, LLM-анализ, дедупликация, обновление фактов и переход по FSM.



Telegram fake gateway позволяет тестировать Avito Legal Bot в реальном времени без боевого Avito-чата.



orchestrator.py связывает LLM-анализ, FSM, storage и обработку фактов в единый pipeline.

ЧТО ВНУТРИ СИСТЕМЫ

Ключевые модули и инженерные слои проекта

Dialogue Flow

- Полный сценарий первичного юридического диалога: ситуация, город, документы, канал отправки, контакт и время связи.
- FSM жёстко контролирует порядок этапов и не даёт боту перескакивать через важные шаги.
- Stage Router определяет актуальную стадию по уже собранным фактам.
- Бот не спрашивает повторно телефон, документы, город или канал, если клиент уже сообщил данные.
- Есть отдельная ветка для нецелевых обращений: контакт сохраняется для передачи партнёрам.

LLM Orchestration

- FSM отвечает за правила и безопасность, LLM отвечает за живую формулировку сообщений.
- Turn Analysis извлекает город, документы, канал, контакт, время связи и намерение клиента.
- Lead Analysis формирует структурированную карточку лида для Google Sheets.
- Строгая JSON-схема делает LLM-ответы предсказуемыми.
- Fallback-политика регенерирует ответ или откатывается на безопасный шаблон.

Legal Guardrails

- Боту запрещено давать полноценные юридические консультации и прогнозы исхода дела.
- Ответы остаются краткими первичными пояснениями и возвращают клиента к сбору данных.
- Документы уточняются только после понимания ситуации и города.
- Телефон в приоритете, при отказе запрашивается альтернативный контакт.
- Служебные контакты компании не записываются как контакт клиента.

Storage & CRM

- SQLite хранит разговоры, сообщения, стадии, ответы, документы и технические флаги.
- Google Sheets работает как CRM-таблица юридических лидов.
- Append/update логика по internal ID не создаёт дубли строк.
- Отложенная запись в Sheets защищает диалог от сбоев таблицы.
- Сохраняются контакт, город, услуга, документы, канал отправки, время связи, комментарий и статус.

Integrations & Runtime

- Avito API получает чаты, сообщения и отправляет ответы клиентам.
- Telegram gateway имитирует клиентов и показывает ответы в реальном времени.
- Единый messenger-интерфейс позволяет менять Avito/fake-режим без изменения бизнес-логики.
- .env управляет Avito, Sheets, OpenAI, SQLite, Telegram fake-mode и режимами диалога.
- Worker polling-loop обрабатывает новые сообщения, отвечает и синхронизирует таблицу.

Tests & Diagnostics

- BDD-сценарии в YAML проверяют диалоги как пользовательские истории.
- Replay-механизм воспроизводит живые LLM-ответы без повторных вызовов OpenAI.
- Fake AI и heuristic AI дают детерминированные тесты.
- Acceptance-тесты проверяют соответствие исходным бизнес-требованиям.
- LLM-as-judge оценивает естественность и соответствие правилам.

▼ Показать полный technical scope

- Avito Messenger intake, FSM-сценарий, LLM-orchestrated и LLM-generated режимы.
- Turn Analysis, Lead Analysis, Stage Router, strict JSON schema и post-processing LLM-текста.
- Дедупликация входящих сообщений, дедупликация исходящих ответов и batching быстрых сообщений клиента.
- Google Sheets append/update по internal ID, маппинг в фиксированную схему колонок и delayed write при сбоях.
- FSM tests, JSON parsing tests, Orchestrator tests, dedup tests, restart resilience tests.
- Production-профиль thin: LLM пишет естественно, FSM контролирует этапы, запреты и обязательные условия.
- Обработка ранних данных, повторных вопросов, soft-must времени связи и post-close update.
- SQLite storage для разговоров, сообщений, стадий, ответов, статусов документов и технических флагов.
- Avito token refresh, fake Telegram mode, messenger factory, .env config и polling worker.
- Live-smoke, replay JSONL, test artifacts, stage trace summary, technical guide и implementation guide.

Python | Avito API | OpenAI API | SQLite | Google Sheets API | Telegram fake gateway | FSM | BDD/YAML tests

Portfolio repo: https://github.com/workflowarsenynn/legal-assistant-automation-portfolio

---

04 · PARSER + LEAD GEN

EventRadar — парсер Telegram-каналов

Отдельный сервис для автоматического поиска горячих заявок на техническое обеспечение мероприятий в Telegram-чатах. Система слушает профессиональные сообщества через Telethon, фильтрует шум, классифицирует лиды и отправляет чистые заявки в backend основного бота.

ЗАДАЧА

Менеджеры вручную мониторили десятки Telegram-чатов, пропускали горячие заявки, ловили дубли и тратили внимание на вакансии, рекламу и нерелевантный шум.

РЕШЕНИЕ

EventRadar собирает сообщения через Telethon, нормализует их в RawMessage, применяет pre-filter, основной фильтр и LLM-классификатор, формирует LeadCandidate и отправляет лиды в backend бота через защищённый webhook.

КАК РАБОТАЕТ

Telegram-чаты → Telethon listener → RawMessage → soft pre-filter → lead classifier / LLM → LeadCandidate JSON → backend /eventradar/lead → Telegram-уведомление продажам.

РОЛЬ

Архитектура сервиса, Telegram client, pre-filter, LLM-классификация, модели данных, backend contract, конфиги и диагностика.

РЕЗУЛЬТАТЫ

- Автоматизирован мониторинг профессиональных event-сообществ
- LLM + эвристики отделяют заявки от шума
- Лиды доставляются в тот же поток, что Avito и site chat
- Bearer-token защищает внутренний endpoint
- Score и reason объясняют, почему сообщение стало лидом
- Supervisor-cycle восстанавливает сервис после временных сбоев

ЧТО ВНУТРИ СИСТЕМЫ

Ключевые модули и инженерные слои проекта

Telegram Source Layer

- Сбор сообщений из выбранных Telegram-чатов через Telethon под рабочим аккаунтом.
- Прослушивание нескольких event-чатов в реальном времени или с минимальной задержкой.
- YAML-конфиг для добавления новых источников без переписывания кода.
- Тестовый Telegram-чат для безопасной проверки логики.
- Скрипт для получения chat_id и списка Telegram-диалогов.

Filtering Pipeline

- RawMessage сохраняет текст, источник, автора, чат, ссылку, reply, forward, media и сервисный статус.
- Soft pre-filter отсекает очевидный шум до дорогой обработки.
- Фильтруются сервисные сообщения, пустые медиа, бессодержательные reply/forward и короткие символьные ответы.
- Pre-filter безопасен: он не режет потенциальные заявки агрессивно.
- Режимы off, soft и зарезервированный hard вынесены в конфиг.

Lead Classification

- Основной фильтр определяет, похоже ли сообщение на запрос по техобеспечению мероприятия.
- Распознаются звук, свет, DJ с аппаратурой, экраны, проекторы, сцена, видео и другое оборудование.
- Категории оборудования: sound, dj_with_gear, lights, led_screen, plasma, projector, stage, video_production, other.
- LLM-классификатор принимает смысловое решение, эвристики страхуют при ошибках.
- Извлекаются город, дата, бюджет, тип события, reason, score и категории.

Backend Contract

- Лиды отправляются в backend основного Telegram-бота через HTTP POST.
- Авторизация между EventRadar и backend идёт через Bearer token.
- JSON-формат LeadCandidate унифицирует передачу лида.
- Endpoint /eventradar/lead валидирует токен, принимает JSON и пересылает уведомление.
- EventRadar подключён как отдельный источник, не ломая Avito и web-chat архитектуру.

Runtime & Recovery

- .env, .env.local и YAML управляют Telegram API, backend URL, token, фильтрами и LLM.
- Telethon session-файл позволяет работать без повторной авторизации.
- Supervisor-cycle перезапускает сборщик при сетевых сбоях, ошибках LLM или недоступности backend.
- Exponential backoff увеличивает задержку до 5 минут при повторяющихся ошибках.
- KeyboardInterrupt корректно останавливает сервис как daemon или Launch Agent.

Product Logic

- Система помогает находить заявки раньше конкурентов сразу после появления в чате.
- Шум превращается в поток потенциальных лидов с объяснимой классификацией.
- Структура уведомлений готова к маршрутизации по оборудованию, городу, бюджету и типу мероприятия.
- Score и reason позволяют анализировать ложные срабатывания и улучшать prompt.
- Проект является частью единой системы входящих обращений из Avito, сайта и Telegram-сообществ.

▼ Показать полный technical scope

- Telethon client, saved session, YAML chat config, test chat и chat_id discovery script.
- Pre-filter modes: off, soft, reserved hard, minimum text length и DEBUG reasons.
- OpenAI-compatible LLM config: API base, model, key, timeout и safe disable при неполной конфигурации.
- HTTP POST backend integration, Bearer token, /eventradar/lead contract и separate EventRadar group routing.
- Supervisor loop, exponential backoff, graceful shutdown, daemon/Launch Agent readiness.
- RawMessage и LeadCandidate модели для унифицированного lifecycle сообщения.
- Lead classifier с категориями оборудования и video_production routing.
- Hybrid LLM + heuristics, permissive mode и legacy stricter heuristic mode.
- Centralized .env/.env.local/YAML, validation, safe defaults и warnings по неизвестным режимам.
- README, release checklist, compileall, unit-testable pre-filter/LLM/filter boundaries.

Python | Telethon | OpenAI-compatible LLM | YAML config | HTTP webhook | Bearer token

Telegram Bot backend

GitHub: в подготовке

05 · RAG + KNOWLEDGE SYSTEMS

Obsidian RAG Core

Backend-first read-only RAG-система поверх Obsidian Vault. Проект индексирует Markdown, Canvas-графы, вложения и OCR-текст, строит retrieval-индекс в SQLite + Qdrant и отвечает через API с sources, evidence snippets и retrieval metadata.

ЗАДАЧА

Обычный поиск по заметкам плохо работает с большой базой знаний: он находит похожие куски текста, но не понимает контекст проекта, связи между идеями, структуру Canvas и границы области поиска.

РЕШЕНИЕ

Система читает Obsidian Vault в read-only режиме, строит собственный индекс, превращает Canvas в semantic graph, поддерживает scope-aware retrieval и отдаёт ответы через FastAPI с источниками, evidence snippets и retrieval diagnostics.

КАК РАБОТАЕТ

Vault discovery → Markdown/Canvas/OCR ingestion → SQLite metadata → Qdrant vector store → scope-aware retrieval → Canvas graph expansion → LLM answer → sources/evidence/retrieval metadata.

РОЛЬ

Архитектура RAG-системы, ingestion pipeline, Canvas IR parser, metadata model, retrieval policies, model adapters, API layer и roadmap retrieval quality / latency.

РЕЗУЛЬТАТЫ

- Read-only backend: vault остаётся source of truth
- Canvas-aware retrieval превращает .canvas в semantic graph
- Scope-aware search по всем проектам, одному проекту или выбранным scopes
- Evidence-first answers с sources и snippets
- OpenAI и локальные Ollama adapters
- Debug search, retrieval timing и semantic traces

ЧТО ВНУТРИ СИСТЕМЫ

Ключевые модули и инженерные слои проекта

Vault Ingestion

- Read-only backend читает Obsidian Vault без изменения исходных заметок.
- Project discovery определяет folder note, _MOC.md, Canvas, Markdown, attachments и legacy media.
- Markdown разбирается на retrieval-документы с metadata и chunking.
- Attachment ingestion поддерживает PDF text layer и optional OCR.
- Incremental indexing обновляет только изменившиеся источники.

Canvas Understanding

- .canvas файлы превращаются в Canvas IR, а не отправляются в модель как сырой JSON.
- Извлекаются nodes, edges, groups, labels, colors, directions, file references и subpaths.
- Canvas semantic layer переводит визуальные связи в retrieval-сигналы.
- Graph expansion учитывает соседние элементы и ограничивает глубину traversal.
- Semantic trace объясняет, почему Canvas был прочитан именно так.

Retrieval Core

- SQLite хранит metadata layer, источники, hash-значения и lookup-данные.
- Qdrant используется как vector store для dense retrieval.
- Scope-aware retrieval не подмешивает unrelated projects без явного разрешения.
- Retrieval profiles explore и fact_evidence используют разные policies.
- Evidence-first answer assembly возвращает sources и snippets.

Model Backends

- OpenAI adapters для embeddings и generation.
- Ollama adapters для локальных embeddings и generation.
- Model abstraction layer позволяет менять backend без переписывания retrieval core.
- Fallback paths деградируют при недоступности внешних backend'ов.
- Локальный запуск подтверждён на Mac Studio с Ollama, Qdrant и FastAPI.

API & Debuggability

- FastAPI endpoints для full reindex, project reindex, changed-only indexing, query и debug search.
- /query возвращает answer + sources.
- /search/debug показывает scope-aware hits без генерации.
- Retrieval metadata показывает backend, model, timings и параметры retrieval.
- Indexing summary logs помогают контролировать процесс индексации.

Roadmap & Evaluation

- Зафиксирован roadmap перехода с 256-dim embeddings на 512/1024-dim embeddings.
- Запланирован настоящий BM25 вместо упрощённого sparse scoring.
- Scope и metadata filtering должны перейти в Qdrant payload filters.
- Планируется async /query и streaming generation.
- Regression retrieval set и benchmarks нужны для оценки retrieval quality.

▼ Показать полный technical scope

- Read-only RAG backend, Projects indexing, project boundary model и strict scope-aware retrieval.
- Canvas IR parser: nodes, file nodes, link nodes, groups, edges, edge labels, directions, subpaths, colors.
- Attachment ingestion: PDF text layer, OCR via tesseract, OCR retrieval-documents.
- Dense retrieval, initial hybrid fusion, profile-aware ranking, per-source cap и source deduplication.
- OpenAI/Ollama adapters, local model stack, fallback paths и vendor-agnostic model abstraction.
- Debug endpoints, retrieval timings, semantic traces, regression retrieval set и evaluation layer.
- Markdown ingestion: folder notes, _MOC.md, frontmatter, wiki-links, headings и project-relative paths.
- Canvas semantic layer: precedence rules, alias resolution, semantic trace, graph expansion и partial reindex.
- SQLite metadata layer, Qdrant vector store, stable document IDs, qdrant-client query_points compatibility.
- Sensitivity-aware metadata, max_sensitivity, exclude_sensitive, access_context и clearance-ready filters.
- Full indexing, incremental indexing, symlink-aware ingestion, canonical source identity и hash-based detection.
- Roadmap: stronger embeddings, BM25, Qdrant payload filters, async flow, streaming generation, query embedding cache.

Python FastAPI SQLite Qdrant Obsidian Canvas Markdown Ollama OpenAI API Docker OCR

Private / in progress

КОНТАКТ

Открыт к AI / automation задачам и техническим ролям

Telegram: @arsenynnn · Email: arsenynickolaevich@yandex.ru · GitHub: workflowarsenynn · Telegram-канал: https://t.me/annsflow