GUI И Workflow Audion Docs AI

Дата актуализации: 2026-07-16

Этот документ описывает текущий интерфейс `Audion Docs AI v3` и то, как пункты GUI связаны с реальными пайплайнами проекта. Source of truth для состава экранов - `config\tool_manifest.yaml`; бизнес-логика находится в `system_core\pipeline.py`, `system_core\doc_task_runner.py`, `system_core\document_normalizer.py` и сервисном слое `system_core\services\audion_docs_service.py`.

Назначение GUI

`launcher_gui.cmd` запускает NiceGUI/desktop shell поверх существующих CLI-движков. GUI не дублирует логику обработки документов: он собирает параметры, показывает статус и живой журнал, а затем вызывает service/CLI-слой.

Основные движки:

- `system_core\pipeline.py` - аудит DOCX/PPTX, отчёты и аннотации.
- `system_core\doc_task_runner.py` - произвольные document tasks по DOCX/PPTX/XLSX/PDF.
- `system_core\document_normalizer.py` - безопасная нормализация DOCX по готовым audit JSON.
- `system_core\document_normalizer.py` и `system_core\pipeline.py` не должны вызываться пользователем напрямую из GUI: интерфейс задаёт корректные аргументы и папки.

Layout

- Слева: Workspace-папки, дерево операций, параметры выбранного режима.
- Справа: статус, прогресс и терминальный журнал.
- Разделитель между левой рабочей зоной и журналом можно двигать мышью.

- Верхняя панель содержит тему и переключатель языка.
- Все интерактивные controls имеют tooltip с задержкой `1500 ms`.
- Цвет tooltip глобально задан как `rgb(23, 33, 43)`.
- Редкие параметры уходят в `Дополнительно`, чтобы основные сценарии оставались короткими.

Workbench I/O

Верхний блок слева — канонический Workbench, общий модуль приложений Audion (Office OCR AI, Build Licenses). Источник может быть папкой или отдельным DOCX/PPTX/XLSX/PDF-файлом; назначение выбирается как папка. Выбранные пути передаются прямо в pipeline, TASK и normalization backend без промежуточного копирования в `input\`. Проектные `input\` и `output\` остаются маршрутами по умолчанию.

Для источника и назначения доступны история путей, pin/unpin, удаление записи из истории и системный picker.

Канонические названия Workbench

Русские кнопки: `Источник`, `Добавить файл...`, `Назначение`, `Сбросить`, `Удалить`, `Список`.

Английские кнопки: `Source`, `Add file...`, `Target`, `Reset`, `Delete`, `List`.

`Сбросить` / `Reset` возвращает маршруты на проектные `input\` и `output\`, очищая только неприкреплённый кэш путей. Эта команда не удаляет документы. `Удалить` / `Delete` очищает выбранные Source и Target только после подтверждения; для внешнего источника интерфейс отдельно показывает предупреждение.

Команды без видимых параметров запускаются непосредственно тёмно-янтарными action-кнопками и не открывают пустое дочернее окно. Внутри кнопки находится только название команды, а описание остаётся отдельным текстом справа. Полноценные формы, select-поля и другие параметризованные команды в action-кнопки не превращаются. Radio и checkbox-контролы оформляются как чипы внутри читаемых секционных блоков.

Главное Меню

Текущий верхний уровень:

- `АУДИТ OPENAI`

- `АУДИТ GEMINI`
- `АУДИТ XAI`
- `АУДИТ CLAUDE`
- `ЗАДАЧА ДЛЯ OPENAI`
- `ЗАДАЧА ДЛЯ GEMINI`
- `ЗАДАЧА ДЛЯ XAI`
- `ЗАДАЧА ДЛЯ CLAUDE`
- `ИСПРАВЛЕНИЕ OPENAI`
- `ИСПРАВЛЕНИЕ GEMINI`
- `ИСПРАВЛЕНИЕ XAI`
- `ИСПРАВЛЕНИЕ CLAUDE`
- `Инструкция задачи`
- `Специализированные команды`
- `Правила аудита`
- `Подготовка`
- `Отчёты`

`Быстрая задача` и `Восстановить регистр после запятых` находятся внутри `Специализированные команды`, а не в корне меню.

Правило документации: названия разделов должны совпадать с интерфейсом. Если пункт переименован в manifest, документация обновляется в тот же заход.

Общие Controls Для Моделей И Ключей

Поля провайдера используют один паттерн для OpenAI, Gemini, xAI и Anthropic Claude:

- select ключа;
- select модели;
- иконка pin/unpin для избранного;
- `+` для добавления API-ключа;
- корзина для удаления API-ключа;
- refresh для обновления списка;
- reset для очистки кэша моделей у provider-селекта.

Добавление ключа:

1. Нажать `+` справа от select ключа.
2. Ввести метку, ключ и комментарий.
3. Поле ключа - textarea: можно вставить чистый ключ или строку `Метка | ключ | комментарий`.
4. После сохранения запись добавляется в файл провайдера:
   - `config\api_key_openai.txt`;
   - `config\api_key_gemini.txt`;
   - `config\api_key_xai.txt`.
5. Новый ключ выбирается в текущем select.

Удаление ключа всегда требует подтверждения. Кэш избранных ключей хранит только refs/метки в `config\gui_key_cache.json`, сами секреты остаются в `config\api_key_*.txt`.

Кэши И Fallback

GUI использует локальные кэши:

- `config\gui_model_cache.json` - модели, избранное, smoke-check статусы;
- `config\gui_key_cache.json` - избранные refs ключей;
- `config\gui_rules_cache.json` - избранные правила;
- `config\gui_doc_task_cache.json` - избранные TASK-инструкции;
- `config\gui_doc_task_quick_cache.json` - быстрые inline-инструкции.

Если модель не выбрана явно, fallback берётся так:

1. последняя модель со статусом `ok` в provider cache;
2. первая избранная модель;
3. если fallback невозможен, запуск просит выбрать модель.

Сырой provider model list не используется как auto-default, потому что он может содержать исторические или недоступные модели.

Пайплайн Аудита DOCX/PPTX

GUI-пункты `АУДИТ OPENAI`, `АУДИТ GEMINI`, `АУДИТ XAI`, `АУДИТ CLAUDE` вызывают `run_pipeline_operation` с режимом `audit` и нужным provider. Это основной пользовательский пайплайн.

Вход

Аудит читает рекурсивно поддерживаемые документы из `input\`:

- `.docx`;
- `.pptx`.

Office temp-файлы вроде `~$...` игнорируются. Для каждого документа строятся относительные пути, чтобы одинаковые имена в разных подпапках не конфликтовали.

Шаг 1. Scan

Режим `scan` доступен в `Подготовка -> Сканировать input`.

Что делает:

- проходит по `input\`;
- собирает список поддерживаемых документов;
- пишет `logs\scan.json`;
- не вызывает LLM;
- не создаёт отчёты.

Когда использовать:

- перед большим запуском, чтобы проверить состав входа;
- если надо убедиться, что вложенные папки видны;
- если пользователь не уверен, попали ли файлы в managed input.

Шаг 2. Render Map

Режим `render` доступен в `Подготовка -> Карта рендера COM`.

Что делает для каждого DOCX/PPTX:

- строит block map;
- создаёт marked OOXML-копию;

- экспортирует документ в PDF через Microsoft Office COM;
- извлекает текст страниц PDF;
- строит render map, связывающую `block_id` с человеческой локацией;
- пишет render logs.

Ключевые артефакты:

```
logs\<relative>\*__block_map.json
logs\<relative>\*__render.json
logs\<relative>\*__render_map.json
logs\<relative>\*__render.log
work\marked_ooxml\<relative>\*__marked.docx|pptx
work\rendered_pdf\<relative>\*.pdf
work\extracted_pdf_text\<relative>\*__pages.json
```

Если Office COM не смог построить PDF, pipeline пишет fallback render map. В обычном audit это не всегда фатально, но при строгом `require_render_map` документ получает статус `failed_render_map`.

Шаг 3. LLM Audit

Режим `audit` делает весь практический прогон:

1. Берёт существующий render map, если он подходит исходному документу.
2. Если render map нет, строит его.
3. Загружает активные или выбранные правила аудита.
4. Делит block map на chunks.
5. Вызывает выбранного provider:
   - OpenAI;
   - Gemini;
   - xAI/Grok;
   - Anthropic Claude.
6. Пишет progress cache по chunks, если включён resume.
7. Нормализует сырые ответы модели.
8. Для русского запуска находит полностью английские или преимущественно английские `problem` и `recommendation`.

9. Отправляет только забракованные поля на дешёвый language-repair: сначала тому же provider, при неуспехе — одному резервному provider.
10. Привязывает проблемы к `block_id`, `render_map`, странице и human location.
11. Только после успешных проверок пишет JSON, XLSX, DOCX и annotated-копию.

Ключевые артефакты:

```
cache\<relative>\*__issues__<provider>_<model>_<reasoning>.json
logs\<relative>\*__audit.json
logs\<relative>\*__language_repair.json
logs\<relative>\*__annotation.json
output\<relative>\*__audit_table.xlsx
output\<relative>\*__audit_report.docx
output\<relative>\*__annotated.docx|pptx
```

LLM cache подписывается не только именем файла. В подпись входят:

- sha256 исходного документа;
- provider;
- model;
- reasoning;
- правила;
- chunk/overlap/min_chunks;
- параметры вывода.

Поле «Многозадачность» (`--workers`) задаёт, сколько частей документа обрабатывается одновременно: части независимы, перекрытие уже вшито в текст каждой. Значение 1 даёт прежнюю последовательную работу, 4 стоит по умолчанию, максимум 32. Ускорение прямое: на 71 части переход с 1 на 8 сокращает прогон примерно с часа до восьми минут, а расход токенов не меняется. Ограничение сверху - лимиты провайдера (запросов и токенов в минуту); при их превышении provider возвращает 429 и отрабатывает штатный retry, поэтому ставить больше, чем позволяет тариф, смысла нет. Результаты собираются по номеру части, так что порядок находок и нумерация не зависят от того, какая часть закончила первой. Уже посчитанные части берутся из cache и при параллельной работе.

Если документ изменился, старый cache считается stale и пересчитывается.

Provider-Особенности

OpenAI:

- использует OpenAI provider;
- штатная модель приложения - `gpt-5.6-luna`;
- штатная глубина аудита - `high`; GUI предлагает `low`, `medium`, `high` и передаёт реальный reasoning effort;
- подходит для основного аудита и TASK.

Gemini:

- использует Gemini provider;
- штатная модель приложения - `gemini-3.6-flash`;
- штатная глубина - `medium`;
- для Gemini 3 уровни `minimal`, `low`, `medium`, `high` передаются как настоящий `thinking_level`; старые модели получают запрос без этого параметра;
- список моделей и проверка доступа идут через provider API;
- при нестабильной сети возможны retry на уровне provider.

xAI/Grok:

- provider id: `xai`;
- ключ хранится в `config\api_key_xai.txt`;
- модели обновляются через xAI Models API/cache;
- штатная модель приложения - `grok-4.3`, штатная глубина - `high`;
- GUI передаёт выбранный `none`, `low`, `medium` или `high` как настоящий `reasoning_effort`;
- provider использует streaming-first запрос, retry/backoff, обработку transient network ошибок и fallback для structured JSON;
- ответы декодируются как UTF-8, а mojibake блокируется до создания отчётов.

Anthropic Claude:

- provider id: `anthropic`;
- ключ хранится в `config□pi_key_anthropic.txt`;
- модели обновляются через Anthropic Models API/cache;
- штатная модель приложения - `claude-sonnet-5`, штатная глубина - `medium`;

- GUI передаёт выбранный `low`, `medium`, `high`, `xhigh` или `max` как настоящий `effort`; thinking остаётся adaptive;
- каждый вызов идёт streaming-запросом, поэтому длинные чанки не упираются в HTTP-таймаут;
- блок инструкций отправляется как кэшируемый system-префикс, поэтому со второго чанка входные токены читаются из кэша;
- отказ модели приходит как успешный ответ со `stop_reason: refusal` и превращается в явную ошибку до записи отчётов.

Язык И Публикация Отчёта

Русские audit-команды передают `report_lang=ru`. В этом режиме переводятся заголовки и технические enum-значения, а prompt требует русский язык в человекочитаемых полях. Полностью или преимущественно английский текст в `problem` или `recommendation` считается браком ответа модели. Pipeline сначала делает короткий запрос к тому же provider только с дефектными полями. Если запрос упал, пропустил поле или снова вернул английский текст, выполняется ровно один cross-provider fallback. Готовые чанки повторно не отправляются.

Каскад фиксирован: OpenAI и xAI переключаются на `gemini-3.6-flash` с `minimal`; Gemini и Anthropic переключаются на `gpt-5.6-luna` с `low`. Первая попытка OpenAI/xAI использует `low`, Gemini — `minimal`. Если ключ или модель fallback недоступны, это считается второй неудачей.

Исходные и исправленные значения, обе попытки, выбранные provider/model, расход токенов и статус сохраняются в `logs\<stem>__language_repair.json`. Повторный запуск с тем же точным набором полей использует этот cache без нового API-вызова. Если обе попытки не прошли проверку, pipeline показывает ошибку и не публикует новый audit JSON/XLSX/DOCX.

Точные цитаты из исходного документа, идентификаторы моделей, API-поля и технические значения без пользовательского перевода остаются неизменными.

Живой Журнал Чанков

Python запускается без буферизации. Сразу после старта журнал показывает `provider`, `model`, `reasoning` и `report_lang`, затем для каждого чанка — `start`, время завершения, токены, ETA и число найденных строк. Если строки появляются только после завершения всей операции, нарушен unbuffered-запуск или чтение stdout.

Шаг 4. Report

Режим `report` доступен в `Отчёты -> Отчёт из журнала`.

Он не вызывает LLM. Он читает `logs\**\*__audit.json` и заново пишет:

```
output\<relative>\*__audit_table.xlsx
output\<relative>\*__audit_report.docx
```

Использовать, если:

- изменился формат отчёта;
- нужно пересобрать DOCX/XLSX после ручной правки audit JSON;
- LLM уже запускалась и повторять её нельзя.

Шаг 5. Annotate

Режим `annotate` доступен в `Отчёты -> Аннотация`.

Он читает audit logs и block map, затем создаёт annotated-копии с якорями ошибок:

```
output\<relative>\*__annotated.docx|pptx
logs\<relative>\*__annotation.json
```

Шаг 6. Report + Annotate

`Отчёты -> Отчёт + аннотация` выполняет `report`, затем `annotate`. LLM не вызывается.

Шаг 7. Strip Anchors

`Отчёты -> Снять якоря` удаляет якоря ошибок из annotated DOCX/PPTX и пишет чистую копию:

```
output\<relative>\*__unanchored.docx|pptx
```

Использовать после ревью, если нужен документ без служебных маркеров.

Правила Аудита

Правила лежат в:

```
config\audit_rules\
```

Активный default:

```
config\audit_rules\active_audit_rules.md
```

Раздел `Правила аудита` позволяет:

- выбрать файл правил;
- сделать его активным;
- добавить правила в избранное;
- импортировать внешний Markdown/TXT в managed config.

Пустой выбор правил в audit-экране берёт активный файл.

Пайплайн Document TASK

TASK-пункты:

- `ЗАДАЧА ДЛЯ OPENAI`;
- `ЗАДАЧА ДЛЯ GEMINI`;
- `ЗАДАЧА ДЛЯ XAI`;
- `ЗАДАЧА ДЛЯ CLAUDE`;
- `Инструкция задачи -> Запустить через ...`;
- `Быстрая задача -> Запустить TASK ...`.

TASK runner читает рекурсивно:

- `.docx`;
- `.pptx`;
- `.xlsx`;
- `.pdf`.

PDF используется как read-only источник текста через PyMuPDF. Поле `Страниц PDF` ограничивает чтение первых страниц, по умолчанию - `5`.

TASK-Инструкция

Managed инструкции лежат в:

```
config\doc_tasks\
```

Активная инструкция:

```
config\doc_tasks\active_doc_task.md
```

В GUI можно:

- выбрать инструкцию;
- сделать её активной;
- добавить в избранное;
- импортировать внешний Markdown/TXT.

TASK Scope

`Охват задачи`:

- `Авто` - один corpus run для нескольких файлов, но точные замены остаются пофайловыми;
- `Весь корпус input` - модель получает общий корпус;
- `Каждый файл отдельно` - отдельный запуск на каждый файл.

TASK Run

Пайплайн TASK:

1. Читает выбранную или активную инструкцию.
2. Добавляет уточнение из поля `Уточнение к TASK`.
3. Строит block map для DOCX/PPTX/XLSX/PDF.
4. Делит данные на chunks.
5. Вызывает OpenAI/Gemini/xAI/Claude.
6. Собирает rows, findings и replacements.
7. Пишет aggregate report.
8. При включённой опции замен применяет безопасные DOCX replacements.

Выход:

```
output\_doc_tasks\<timestamp>__doc_task.json
output\_doc_tasks\<timestamp>__doc_task.xlsx
output\_doc_tasks\<timestamp>__doc_task.docx
output\_doc_tasks\<timestamp>__doc_task.md
```

Для пофайловых задач также используются per-document cache-файлы:

```
cache\<relative>\*__doc_task__<provider>_<model>.json
```

Для corpus-задач:

```
cache\_doc_tasks\corpus__<cache_id>__<provider>_<model>.json
```

TASK cache подписывается source hash, инструкцией, уточнением, provider/model, chunk-параметрами, лимитом PDF-страниц и системным prompt.

Точные Замены DOCX

Опция `Найти и заменить текст в документах DOCX` применяет только пары `old_text -> new_text`, которые вернула модель. Замены выполняются безопасно:

- только в DOCX;
- только там, где старый текст найден ожидаемо;
- результат пишется в `output\`, исходник не правится.

Чистая Таблица По Шаблону

Опция `Чистая таблица по шаблону` пишет дополнительный чистый экспорт по DOCX/XLSX-шаблону из `input\`.

Правила:

- если шаблон один, пустой выбор берёт его автоматически;
- если кандидатов несколько, шаблон нужно выбрать явно;
- первая строка таблицы или листа задаёт точные названия колонок;
- следующая непустая строка используется как подсказка формата;
- лишние ключи из `values` не попадают в чистую таблицу.

Выход:

```
output\_doc_tasks\<timestamp>__doc_task_clean.json
output\_doc_tasks\<timestamp>__doc_task_clean.xlsx
output\_doc_tasks\<timestamp>__doc_task_clean.docx
```

DOCX clean output пишется только если шаблон был DOCX.

Быстрая Задача

`Быстрая задача` нужна для inline-инструкций без создания Markdown-файла.

Можно:

- написать инструкцию прямо в GUI;
- сохранить её в историю;
- добавить сохранённую инструкцию в избранное;
- запустить через OpenAI/Gemini/xAI.

История и избранное живут в `config\gui_doc_task_quick_cache.json`.

Нормализация Документов

Пункты:

- `ИСПРАВЛЕНИЕ OPENAI`;
- `ИСПРАВЛЕНИЕ GEMINI`;
- `ИСПРАВЛЕНИЕ XAI`.

Нормализация не вызывает LLM. Она читает готовые audit logs:

```
logs\**\*__audit.json
```

Фильтр provider берётся из `meta.provider`.

Применяются только безопасные исправления:

- `fix_mode` входит в safe replace режимы;
- confidence высокий;
- есть `block_id`;
- есть `old_text` и `new_text`;

- `old_text` найден в нужном блоке ровно один раз.

Выход:

```
output\_normalization\<timestamp>__normalization_<provider>.json
output\_normalization\<timestamp>__normalization_<provider>.xlsx
output\_normalization\<timestamp>__normalization_<provider>.docx
output\_normalization\<timestamp>__normalization_<provider>.md
report\document_normalization\<timestamp>__normalization_patch_plan_<provider>.json
report\document_normalization\latest_normalization_patch_plan_<provider>.json
```

Сомнительные случаи попадают в `Unresolved Items`; скрипт не угадывает замену сам.

Восстановить Регистр После Запятых

Раздел находится в `Специализированные команды`.

Он работает с готовыми артефактами DocFlow и не запускает DocFlow повторно. Подробности в `docs\COMMA_LOWERCASE_RESTORE_RU.md`.

Режимы источника решений:

- готовый JSON/карта;
- OpenAI;
- Gemini;
- xAI, если сценарий подключён через общие provider-поля.

Локальный скрипт только применяет решение `restore` или `keep` по координатам. Смысловое решение принимает LLM или человек.

Отчёты И Якоря

Раздел `Отчёты` работает только с уже созданными logs:

- `Отчёт из журнала` - пересобрать XLSX/DOCX;
- `Аннотация` - пересобрать annotated DOCX/PPTX;
- `Отчёт + аннотация` - оба шага подряд;
- `Пересобрать якоря` - повторная аннотация из logs;
- `Снять якоря` - создать чистую копию без error anchors.

LLM на этих шагах не вызывается.

Подготовка

Раздел `Подготовка`:

- `Проверить input` - список входных DOCX/PPTX/XLSX/PDF без provider-вызовов;
- `Сканировать input` - pipeline scan и `logs\scan.json`;
- `Карта рендера COM` - render map без LLM.

Использовать перед большим аудитом, если нужно заранее поймать проблемы Office COM, пустой input или неверную структуру папок.

Проверка Моделей

`ПРОВЕРИТЬ МОДЕЛЬ` отправляет маленький provider-запрос и сохраняет результат в `config\gui_model_cache.json`.

Статусы:

- `ok` - модель отвечает;
- `error` - запрос завершился ошибкой;
- `no_access` - модель видна или указана, но аккаунт не имеет доступа.

Проверка модели не заменяет refresh списка. Refresh отвечает на вопрос "что provider сейчас перечисляет", smoke-check отвечает на вопрос "эта модель реально запускается с этим ключом".

Проверки После GUI-Изменений

Минимум:

```
runtime\python.exe -m compileall system_core tests
runtime\python.exe -m unittest discover -s tests
```

Визуальный smoke:

- корневое меню;
- `АУДИТ OPENAI/GEMINI/XAI`;
- `ЗАДАЧА ДЛЯ OPENAI/GEMINI/XAI`;

- `Быстрая задача`;
- `ИСПРАВЛЕНИЕ OPENAI/GEMINI/XAI`;
- popup добавления ключа;
- delete-confirm ключа;
- tooltips с задержкой `1200 ms`;
- закрытый и открытый блок `Дополнительно`;
- терминал после короткого запуска.

Cleanup И Init Folders

- `cleanup_project.cmd` чистит managed/generated зоны проекта.
- `install\init_folders.cmd` создаёт ожидаемую структуру.
- `.gitkeep` в payload-папках `runtime`, `wheelhouse`, `system_core\powershell` необязателен.
- Каталоги `backup_before_*` нельзя создавать в корне проекта. Для отката используется внешний release-архив, контроль версий или backup вне `Audion Docs AI`.
- Временные QA-render каталоги, stale PID, `__pycache__` исходников/tests и `.pytest_cache` не являются артефактами проекта и удаляются после проверки.
- После любых правок `.cmd` запускать `install\Check-CmdEncoding.cmd`.