Audion DocFlow (Portable)

Содержание

Портативный офлайн-набор для предсказуемой очистки, контроля и процессинга документов **DOCX**, **XLSX**, **CSV** и **Markdown**.

Проект приведён к стабильной portable-схеме:

- пользовательские launcher-файлы лежат в корне;
- service/build слой отделён от пользовательского меню;
- обработанные файлы пишутся в `output\`;
- отчёты пишутся в `report\`;
- временные файлы launcher-логики лежат в `._runtime\`.

Текущий фокус проекта - **не OCR**. OCR-возможности вынесены из основного каркаса и не должны возвращаться в этот набор случайно: для OCR лучше держать отдельный специализированный проект.

Что делает проект

DOCX

- проверка качества DOCX для поиска не-чёрного цвета шрифта, подсветки, заливок, зачёркивания, комментариев и правок из режима рецензирования
- строгая проверка DOCX с ненулевым кодом для автоматических сценариев
- одношаговая финализация с итоговой проверкой
- удаление зачёркнутого текста в финальной очистке без удаления абзацев, таблиц, разрывов страниц и секций
- удаление комментариев
- попытка принять правки из режима рецензирования без ручного открытия документа
- проверка и исправление пробелов и пунктуации
- `--dry-run` для текстовой гигиены, чтобы увидеть план удаления временных Office-файлов и записи в `output\` без изменений
- удаление непечатаемого мусора DOCX в консервативном режиме: только мягкие переносы (`U+00AD` и `w:softHyphen`); пробелы, табы, разрывы, видимые дефисы и управляющие переносами символы не трогаются
- раздел `Найти и заменить`: отчёт по подозрительному верхнему регистру после запятых только внутри ячеек таблиц DOCX и опциональное понижение регистра обычных слов с сохранением форматирования фрагментов Word; отчёт включает каждое срабатывание с фрагментом текста после смоделированной правки
- морфологический поиск и замена в DOCX/XLSX через `pymorphy3`: поиск по леммам, режимы `Заменить` и `Добавить рядом`, отчёт пробного запуска, опциональное согласование падежа заменяющей/добавляемой фразы; отчёт включает каждую замену полным предложением от точки до точки
- команды раздела `Найти и заменить`, которые могут менять текст, пишут гибридные отчёты: общий сводный файл в `report\` и пофайловые `.md/.json` в папке с тем же именем отчёта, где имена файлов совпадают с исходными документами
- стили: разведка документа перед переносом, перенос стилевой базы эталона (styles.xml, нумерация, тема, шрифты), разметка заголовков, подписей и перечней по карте заголовков,

сквозная нумерация подписей и чистка XML от мусора

- `DOCX anomaly inspector` в разделе `СТИЛИ И ГИГИЕНА`: проверка без изменения документов явных аномалий DOCX с человеческой локацией `страница / раздел / Таблица N / Рисунок N`; RO-слой проверяет таблицы, подписи таблиц/рисунков, секции/ориентацию, колонтитулы и нумерацию страниц, поля/оглавление/ссылки/закладки, списки/нумерацию, пустоты и заголовки; отдельная корректировка безопасных правок пишет копии в `output\docx_anomaly_fixed` и умеет убирать лишние пустые абзацы, снимать точную высоту строк таблиц, включать перенос текста в ячейках и опционально унифицировать границы/поля ячеек
- `Глубокая гигиена DOCX` в разделе `СТИЛИ И ГИГИЕНА`: общий двухпроходный запуск, где `Гигиена текста` и правила аудита идут первым проходом, а `Аномалии документа` могут подключаться вторым проходом; единый отчёт `docx_deep_hygiene.md/.docx/.json` показывает pass-records, `domain_boundaries`, контракт владения классами ошибок, `target_index`, `unified_findings`, отпечатки целей/ошибок и подавленные точные дубли; будущим остаётся только отдельный морфологический слой для адресов и топонимов
- аудит правил: детерминированные корпоративные правила для `кв. м`, `куб. м`, `№ 1`, процентов, градусов, дат без `г.`, опечаток в названиях таблиц и режима только отчёта для `РФ`; операция может создавать `*__annotated.docx` с видимыми якорями аудита и затем снимать эти якоря после одобрения правок
- поиск похожих DOCX
- точный hash-дедупликатор внутри отчёта похожести DOCX
- diff двух DOCX без порога: по умолчанию сопоставляет разделы по похожести, затем сравнивает текст и таблицы внутри найденных пар; доступен режим сравнения по порядку документа; оглавление, media и внутренняя структура пакета не анализируются
- нативное сравнение двух DOCX через Microsoft Word COM с созданием отдельного DOCX с правками Word
- склейка DOCX через Microsoft Word COM, близко к ручной вставке документа после документа
- сшивка разрезанных таблиц Word
- `Document tables unifyer`: одношаговая унификация всех таблиц внутри существующего DOCX без пересборки текста документа; автоопределение шрифта основного текста, два размера по плотности таблиц, единые границы, поля ячеек по умолчанию `0,2 см`, пропуск первых двух страниц, балансировка колонок и вписывание в поля разделов
- объединение совместимых DOCX-таблиц: только таблицы или таблицы с заголовками/разделами как объединёнными строками; Unifier аккуратнее сохраняет найденные

многострочные и объединённые шапки, поддерживает параметр `Строки перед шапкой`, выбор A4/A3, ориентации и полей в миллиметрах

- оптимизация ширин DOCX-таблиц, адаптация таблиц к ориентации страницы и вписывание таблиц в текущие поля документа или заданный формат; параметр `Не вписывать в поля по числу столбцов` по умолчанию не трогает таблицы с 3 столбцами или меньше
- извлечение всех таблиц DOCX в XLSX с индексом
- извлечение всех медиафайлов/изображений DOCX/PPTX с индексом
- уменьшение полей ячеек таблиц DOCX до `0,1 см`; для XLSX сбрасывается доступный текстовый отступ, потому что формат XLSX не хранит настоящие внутренние поля ячеек как Word

XLSX, CSV и Markdown

- сравнение двух XLSX по вычисленным значениям
- сверка DOCX/XLSX/CSV таблиц в четыре списка
- автоподбор ключевой колонки для сверки таблиц через `--auto-key`
- экспорт таблиц Markdown в XLSX
- очистка незначащих строк Excel: создание XLSX-копий без полностью пустых строк и строк только из пробелов

GUI

- GUI-оболочка `launcher_gui.cmd` поверх существующих CLI/FZF-команд
- канонический Workbench передаёт выбранные внешнюю папку или один файл `Источник` и папку `Назначение` непосредственно backend, без копирования в локальный `input\`
- цветовые темы в шапке GUI: выбранная тема хранится в `config\gui_settings.yaml`, палитры и CSS-токены - в `config\ui_colors.yaml`; по умолчанию используется `Code Темная` (`code_dark`)
- отдельный раздел `Найти и заменить` для команд регистровой и морфологической автоматизации
- отдельный раздел `АУДИТ ПРАВИЛ`: аудит DOCX с якорями, безопасными правками и пробным запуском, а также снятие якорей с одного файла или пакета; кнопка `ПРАВИЛА` открывает папку `config\rules\`
- отдельный раздел `СТИЛИ И ГИГИЕНА` с рабочей панелью и переключателями `ГИГИЕНА ТЕКСТА`, `АНОМАЛИИ ДОКУМЕНТА`, `СТИЛИ ДОКУМЕНТА`; режимы `Проверка` и `Корректировка`

подсвечены разными кантами, чекбоксы классов проверок выстроены адаптивной сеткой, а кнопка запуска находится в верхней строке справа на уровне `НАЗАД`

- отдельный раздел `ТАБЛИЦЫ WORD/EXCEL` для DOCX-таблиц: корневая команда `Унифицировать таблицы в документе`, обычная сшивка и отдельное восстановление повторяемой шапки, безопасное и width-only объединение, оптимизация ширин, адаптация ориентации, вписывание и извлечение таблиц
- leaf-команды без параметров запускаются компактными тёмно-янтарными action buttons: кнопка содержит только название, а описание расположено отдельно; параметризованные команды сохраняют формы в светлых скруглённых блоках с тёмными полями и checkbox/radio-чипами
- подсказки в GUI доступны при наведении на названия подразделов, команды и поля параметров; длинные пояснения не занимают постоянное место в панели
- XLSX-сравнение и сверка таблиц находятся в разделе `Сравнение и сверка`; Markdown-таблицы в XLSX и уменьшение полей ячеек находятся в `Технические операции`
- служебные и релизные команды остаются в `launcher_tools.cmd` / `builder_main.cmd` и не показываются на первом экране GUI

Launcher-слой

Пользовательские launcher-файлы

- `launcher_project.cmd` - английский launcher
- `launcher_project_ru.cmd` - русский launcher

Оба launcher-файла используют одну и ту же внутреннюю логику:

- единая схема `FZF + CMD fallback`;
- первый экран разделён по смыслу: `АУДИТ ПРАВИЛ`, `СТИЛИ И ГИГИЕНА`, `Найти и заменить`, `DOCX контроль и очистка`, `Сравнение и сверка`, `ТАБЛИЦЫ WORD/EXCEL`, `Технические операции`, `Проверки`;
- правила аудита отделены от стилей и текстовой гигиены;
- раздельные temp-файлы в `._runtime\`:
  - EN -> `project_menu_en.txt`, `project_menu_en_res.txt`
  - RU -> `project_menu_ru.txt`, `project_menu_ru_res.txt`

Service layer

- `builder_main.cmd` - launcher для задач сборки и релиза
- `launcher_tools.cmd` - launcher для служебных задач

Этот слой теперь опирается на:

- `install\init_folders.cmd`
- `install\make_release_archive.cmd`
- `system_core\license\`
- `licenses\`

Рекомендуемый сценарий работы

1. Положите исходные файлы в `input\`.
2. Запустите `launcher_project.cmd` или `launcher_project_ru.cmd`.
3. Выберите нужный инструмент.
4. Заберите обработанные файлы из `output\`, отчёты - из `report\`.

Для окружения и релизных задач:

- используйте `builder_main.cmd`;
- используйте `launcher_tools.cmd` для service/licensing-операций.

Для GUI:

```
launcher_gui.cmd
```

Отчёты и язык интерфейса

- Пользовательские отчёты Markdown/DOCX формируются на русском языке.
- В разделе `Найти и заменить` Markdown-отчёты предназначены для человека, а JSON-отчёты являются машинным контрактом для внешнего LLM-pipeline. Сам проект LLM не вызывает: он только детерминированно собирает кандидаты, контекст и смоделированный результат правки.
- Для автоматизации инструменты проверки и отчётности поддерживают опциональный `--json-out PATH`; JSON пишется UTF-8 с русскими строками без escaping.

- Технические статусы и идентификаторы (`OK`, `PASS`, `FAIL`, `diff`, `SHA-256`, `DOCX A/B`, ключи JSON/YAML, CLI-параметры, расширения файлов, имена папок, `COM`, `PowerShell`, `runtime`, `wheelhouse`, `launcher`, `build`, `release`, style id Word вроде `a7`) сохраняются как технические значения.
- Описания сценариев переводятся по смыслу: человеческое объяснение пишется по-русски, но фактические имена папок (`input`, `output`, `report`) и точные технические термины можно оставлять рядом в скобках только там, где это имя команды, файла или поля отчёта.
- В русской UI табличные термины переведены явно: `pre-header` - это `Строки перед шапкой`, а повторяемая шапка - это строки шапки таблицы, которые Word повторяет при переносе.
- Вторичные диагностические GUI-кнопки могут оставаться сухими техническими метками: `LOGS`, `REPORT`, `CONFIG`, `TOOLS`.
- Старые файлы в `report\`, созданные до обновления, не переписываются сами: для русского отчёта нужно запустить соответствующую операцию заново.

Smoke-тесты

Быстрый smoke:

```
& '.\runtime\python.exe' '.\tests\smoke.py' --quick
```

Полный smoke для набора scripted-инструментов:

```
& '.\runtime\python.exe' '.\tests\smoke.py' --full
```

Структура проекта

- `launcher_project.cmd`, `launcher_project_ru.cmd` - пользовательские точки входа
- `launcher_gui.cmd` - GUI-оболочка проекта
- `builder_main.cmd`, `launcher_tools.cmd` - service/build точки входа
- `input\` - исходные файлы
- `output\` - обработанные результаты
- `report\` - отчёты
- `logs\` - логи

- `install\` - скрипты сборки portable-среды и релиза
- `system_core\` - Python-инструменты и внутренние helper-скрипты
- `system_core\ui_nicegui\` - GUI-shell из portable-шаблона
- `system_core\services\` - адаптеры GUI к существующим CLI-командам
- `system_core\word_com\` - PowerShell/Word COM helper-скрипты
- `system_core\license\` - release licensing helper-слой
- `licenses\` - собранные release notices
- `config\` - зарезервированная зона конфигов проекта
- `config\tool_manifest.yaml` - дерево команд GUI
- `config\gui_settings.yaml` - язык, тема GUI и GUI-only настройки
- `config\ui_colors.yaml` - палитры и CSS-токены цветовых тем GUI
- `config\rules\rules.yaml` - машинная карта правил аудита проекта
- `config\rules\rules.md` - разделение правил аудита на безопасные для проекта правила, AI-остаток и маршрут с якорями
- `runtime\` - встроенный Python runtime
- `wheelhouse\` - офлайн-колёса
- `release\` - собранные релизы
- `GitHub\` - публикационная документация
- `._runtime\` - temp-файлы launcher-слоя

Важные замечания

- Исходники не переписываются на месте; результаты пишутся в `output\`.
- Гигиена DOCX удаляет временные Office-файлы `~$*` внутри выбранного `input\`.
- Вся работа локальная и офлайн.
- Сравнение DOCX через Word COM и склейка DOCX через Word COM требуют установленный Microsoft Word и доступный PowerShell (`system_core\powershell\pwsh.exe`, `pwsh.exe` или Windows PowerShell).
- Папка `system_core\powershell\` зарезервирована под portable PowerShell runtime; рабочие скрипты туда не складываются.
- Сложные истории правок и нестандартные DOCX-структуры всё ещё требуют визуальной проверки в Word.

Troubleshooting

Если runtime отсутствует, используйте:

```
builder_main.cmd
```

или:

```
install\Build_Portable_Env_Build.cmd
```

Если launcher ведёт себя странно:

- сначала подтвердите `UTF-8 without BOM` и `CRLF` у `.cmd` через `install\Check-CmdEncoding.cmd`;
- затем смотрите `._runtime\` и launcher-специфичные temp-файлы, перечисленные выше.

Лицензия

См. основной license-файл проекта, если он присутствует, и `licenses\THIRD_PARTY_NOTICES.md` в релизно-ориентированных сборках.

Канонические названия Workbench

Workbench использует единый публичный словарь Audion Image Tools во всех проектах. Кнопки всегда расположены и называются одинаково: **Источник**, **Добавить файл...**, **Назначение**, **Сбросить**, **Удалить**, **Список**.

`Сбросить` возвращает проектные `input/output` и не удаляет файлы; `Удалить` очищает текущие `Источник` и `Назначение` только после подтверждения. В английском интерфейсе точные названия: **Source**, **Add file...**, **Target**, **Reset**, **Delete**, **List**. Варианты `Цель`, `Очистить`, `Destination` и `Clear` для этих элементов Workbench не используются.