Smoke-Test Checklist

Содержание

Audion DevOps Tools Specific

После изменений в этом проекте дополнительно проверить:

- корневой список содержит отдельные секции `Network Cleaner`, `Подключение и адаптеры` и `Обслуживание и очистка`;
- корневые секции не подсвечиваются красным из-за `kind: dangerous`;
- в шапке есть selector темы, default `Code Темная`, выбор темы не обрезает заголовок и controls;

- `runtime\python.exe system_core\doctor.py` показывает `[GUI themes]` без ошибок;
- `WSL Toolkit / Базовые и установка` показывает не пустой экран, а команды WSL status/features/update/list online/install/install from image/list/status/shutdown;
- в `WSL Toolkit` нет обязательного выбора `Audion_WSL_Block_E/S`; установка принимает выбранную папку и локальный `.wsl` файл;
- `Виртуализация` стоит отдельной секцией после `WSL Toolkit`; `Статус виртуализации` read-only печатает hypervisorlaunchtype, optional features и VBS/Core Isolation; `Статус оптимизации` read-only печатает Core Isolation/VBS, power plan, Defender exclusions, `.wslconfig` и WSL VHDX placement;
- destructive-команды `Виртуализация` показывают confirm, делают BCD backup в `backup\virtualization` и явно пишут, что нужен reboot;
- `Подключение и адаптеры / Wi-Fi профили` показывает import XML рядом с export profiles, с file/folder import и scope `current/all`;
- `Подключение и адаптеры / SMB вход в сеть` показывает кэш computer/user, ручные поля, save/delete/clear cache, checkbox открытия Explorer; пароль вводится только во внешней `net use` консоли и не появляется в GUI/log;
- `Hosts and Bitrix / Detect current endpoint` не меняет hosts, принимает только local/private DNS IP и возвращает `field_updates` для `ip_address` / `bitrix_ports`;
- `Hosts and Bitrix` default: `portal.itpgrad.ru -> 192.168.0.130`, `bitrix_ports=443`;
- `Hosts and Bitrix / Status / DNS / ports` показывает hosts entries, effective resolution, DNS without hosts, configured DNS servers, авто-скан портов и итоговую TCP-проверку custom/detected ports;
- `Hosts and Bitrix / Disable override` восстанавливает `hosts` побитово из `backup=hosts_prepatch_....bak` в managed-комментарии, а не пересобирает файл построчно;
- `docs\BITRIX_HOSTS_RU.md` описывает рабочий сценарий detect -> status -> enable -> disable и bitwise depatch;
- `Default Apps Guard` показывает команды списком с компактным описанием справа и полным tooltip: status, snapshot, rescan/update, import XML, apply/repair policy, remove policy, cleanup backups и open folders;
- `Default Apps Guard / Status` можно запускать read-only; он показывает current/profile/policy comparison и не пытается править `UserChoice`;
- `Default Apps Guard / Проверить защиту defaults` показывает Windows edition и поддержку `DefaultAssociationsConfiguration`; на Home/Core `Включить / починить защиту defaults` по умолчанию должен отказать, кроме expert override;

- `docs\DEFAULT_APPS_GUARD_RU.md` содержит короткий сценарий для свежей Windows: status -> rescan -> remove Suggested -> apply -> sign out/reboot -> status, плюс gotchas;
- `Runtime and shell` содержит только preflight/runtime status, Windows DEV settings и portable PowerShell install; Nuke-утилит там быть не должно;
- `Обслуживание и очистка` содержит отдельные русские блоки `Очистка Codex` и `Очистка Python` с Audit/DryRun/destructive leaf-командами плюс `Очистить workspace`; висячих cleanup-команд в корне быть не должно;
- `Hardware` содержит `Накопители / дисковые процедуры` с disk inventory, selected disk details, SSD/NVMe wizard и WinRE; отдельного корневого `Storage` быть не должно;
- `Ключи, шрифты и настройки Windows` — отдельный раздел верхнего уровня с шестью паками: OpenSSH KeyKit, Сертификаты, Шрифты пользователя, Среда оболочки, Доступы из конфигурации, Переезд на новую машину; VSCode kits в проекте больше нет, ими занимается отдельный проект `Audion Setup for VS Code`;
- `Сертификаты (экспорт/импорт)` стоит рядом с `OpenSSH KeyKit`; status read-only, PFX export имеет `risk_level: secret_export`, import PFX/CA меняют certificate store;
- `Utilities / Documentation PDF` содержит dry-run plan, generate и open folder; output root должен быть `docs\PDF`, не `output\docs_pdf` и не рядом с Markdown;
- `AI CLI Backup` показывает отдельные поля для export/import/merge: auth выключен по умолчанию, import/merge начинаются с Dry run, Full влияет и на экспорт, и на импорт; новый bundle содержит проверяемый `manifest.json`;
- длинные command buttons читаются в две строки без грубого обрезания первого слова/смысла;
- видимые descriptions у длинных операций остаются короткими, а полный текст доступен в tooltip по кнопке или описанию;
- парные workflow-команды визуально связаны скруглёнными рамками: backup/restore, export/import, block/unblock, capture/apply;
- мягкие цветовые тона кнопок видны только там, где соседние сценарии различаются силой, и не превращают UI в тревожную раскраску;
- большие checkbox-группы, например Default Apps Guard `Отслеживать`, лежат в ровной сетке внутри Advanced-блока и имеют быстрые пресеты `Все`, `Снять`, `По умолчанию`, `IMAGE`, `VIDEO`, `AUDIO`, `OFFICE`;
- в Workbench-панели строго показаны `Источник`, `Добавить файл...`, `Назначение`, `Сбросить`, `Удалить`, `Список`; `Сбросить` возвращает проектные пути, а `Удалить` очищает выбранные пути после подтверждения;

- в правой панели возле журнала есть `ROOT`, `INPUT`, `OUTPUT`, `WORK`, `Logs`, `Report`, `Config`;
- нижняя command bar терминала показывает shell selector, command history/pin controls, cwd picker и file picker;
- command bar различает `История` и `Кэш`: `История` очищает незакреплённые команды, `Кэш` сбрасывает history/pinned/last command, но сохраняет shell и CWD;
- выбор файла в command bar добавляет путь файла в команду и ставит cwd в папку файла;
- `wsl_online_distro_options()` возвращает чистые distro ids вроде `Ubuntu`, `Ubuntu-26.04`, `Ubuntu-24.04`, `Debian`, а не строки с кракозябрами.
- `wsl.exe --list --online` в GUI-терминале декодируется как читаемый русский/английский текст, даже когда WSL ещё не установлен и команда выходит с кодом 1.

Быстрая проверка WSL provider:

```
runtime\python.exe -c "from pathlib import Path; import sys; sys.path.insert(0,
str(Path.cwd())); import system_core.services.devops_tools as d;
print(d.wsl_online_distro_options(Path.cwd())[:8])"
```

Runtime Imports

```
runtime\python.exe -c "import nicegui, webview, yaml, rich; print('OK GUI imports')"
```

Если проекта ещё нет в portable runtime, используйте системный Python 3.12.

Pytest

```
runtime\python.exe -m pytest -q
```

Ожидается: `36 passed`, без skipped при включённом Windows Developer Mode. Два symlink-теста проверяют, что cleaner не следует по ссылке за пределы workspace и отказывается чистить workspace, который сам является ссылкой. На Windows без права создания symlink только эти проверки пропускаются с явным `WinError 1314`.

Maintenance Scripts

```
init_folders.cmd
cleanup_project.cmd /DRYRUN /Y
cleanup_project.cmd /BACKUP /DRYRUN
```

Ожидается: `init_folders` выходит с кодом 0, cleaner в dry-run показывает очистку generated/downloaded зон проекта (`runtime`, `wheelhouse`, `release`, `install\download`, `system_core\powershell`, `fzf.exe`, logs/report/workspace/output/input/data/backup). Папки и `.gitkeep` восстанавливаются. `/BACKUP /DRYRUN` показывает более узкий план очистки только backup-снимков и требует отдельный `Y/N/Q` при реальном запуске.

Module Launchers

Короткая проверка меню без запуска тяжёлых операций:

```
"0" | cmd /d /c launcher_project.cmd
"0" | cmd /d /c launcher_project_ru.cmd
"0" | cmd /d /c cli\launcher_wsl.cmd
"0" | cmd /d /c cli\launcher_bitrix.cmd
"0" | cmd /d /c cli\launcher_default_apps.cmd
"0" | cmd /d /c cli\launcher_association_defense.cmd
"0" | cmd /d /c cli\launcher_hardware.cmd
"0" | cmd /d /c cli\launcher_docs_pdf.cmd --dry-run
```

Ожидается: меню открывается, кириллица читаемая, выход по `0` без ошибки. `cli\launcher_wsl.cmd`, `cli\launcher_bitrix.cmd` и `cli\launcher_default_apps.cmd` должны вызывать manifest-операции через `system_core\cli_operation.py`, а не копировать сервисную логику. `cli\launcher_docs_pdf.cmd --dry-run` должен показать план PDF без записи файлов.

Отдельно проверить руками, потому что эти wrappers передают управление `Nuke.cmd` с UAC elevation:

```
cli\launcher_codex_nuke.cmd
cli\launcher_python_nuke.cmd
```

Ожидается: открывается меню встроенного `tools\...\Nuke.cmd`; destructive modes всё ещё требуют typed confirmation.

Syntax

```
runtime\python.exe -m py_compile system_core\ui_nicegui\app.py
system_core\ui_nicegui\window.py
```

CMD Encoding

Все `.cmd` должны быть:

```
UTF-8 without BOM + CRLF
```

ВАЖНО: ВСЕ `.CMD` ФАЙЛЫ ОБЯЗАТЕЛЬНО UTF-8 БЕЗ BOM И СТРОГО CRLF.

LF-only `.cmd` перед релизом исправить. После любого patch/edit проверять `BOM=False` и `LoneLF=0`.

Штатная проверка/ремонт для проекта:

```
install\Check-CmdEncoding.cmd -Fix
```

Эта же проверка встроена в portable build, offline install, verify и release archive gate.

NiceGUI Smoke

```
runtime\python.exe system_core\ui_nicegui\app.py --smoke
```

Ожидается строка `OK nicegui shell`.

Эта проверка также вызывается из:

```
install\verify_portable_env.cmd
```

если `system_core\ui_nicegui\app.py` существует.

Server Check

Запустите на свободном тестовом порту:

```
runtime\python.exe system_core\ui_nicegui\app.py --host 127.0.0.1 --port 8099 --no-browser
```

Проверьте `http://127.0.0.1:8099/`.

Window Check

```
launcher_gui.cmd
```

Ожидается UAC-запрос, затем отдельное desktop-окно pywebview уже в elevated-контексте. Браузер не должен открываться сам.

Read-only/debug запуск без UAC:

```
set AUDION_GUI_NO_ELEVATE=1
launcher_gui.cmd
```

Ожидаемый стартовый размер окна: `1600x900`. Минимальный размер около `1180x720`.

Picker Check

Проверьте:

- `Add files...` открывает Windows file picker;
- можно выбрать несколько файлов;
- файлы копируются в `input`;
- `Add folder...` копирует папку в `input`;
- повторяющиеся имена получают уникальные suffix;
- попытка добавить сам `input` не проходит.

Layout Check

На WUXGA `1920x1200` с Windows scale 150%:

- окно можно сжать до ноутбучного логического профиля без развала двух колонок;
- команды остаются слева, статус и терминал справа;
- терминал не уезжает под список команд;
- нет раннего перехода в вертикальную ленту.

На FullHD/4K:

- кнопки в одну строку;
- терминал справа занимает большую часть высоты;
- status/progress не раздувают layout;
- `Cancel` виден только во время операции;
- `Logs` рядом с терминалом.
- под терминалом есть постоянный итоговый индикатор: серый в ожидании, синий во время выполнения, зеленый после успешного завершения, красный после ошибки.
- левая колонка не растягивается бессмысленно;
- терминал остается читаемым;
- нет пустых карточек ради декора.

Visual Smoke Screenshots

После заметных GUI/layout-правок сохраняйте smoke-скриншоты в проектной зоне отчётов, например:

```
report\gui_smoke_screenshots\
```

Минимальный набор:

- корневое меню с рабочими папками и терминалом;
- один главный экран команды с основными полями;
- тот же или похожий экран с раскрытым `Дополнительно`, если менялись редкие поля;

- экран TASK/длинной формы, если проект использует nested `operation_groups`;
- терминал и нижний статус после короткого успешного запуска.

Цель не в красивом альбоме, а в том, чтобы портирование ловило реальные UI-регрессии: не помещающиеся dropdown, слишком яркие рамки, дублирующие controls, ранний перенос в одну колонку, лишний scroll и невидимый финальный статус.

Nested Menu And Fields Check

Если проект использует `operation_groups` и `fields`:

- переходы по вложенному меню меняют только левую область команд;
- правый терминал, статус, прогресс и кнопки папок остаются на месте;
- leaf-команда сначала показывает финальный экран `Запустить` / `Назад`;
- `text`, `number`, `select`, `checkbox` и `checkboxes` отображаются корректно;
- `radio`, `profile_buttons` / `preset_buttons` отображаются корректно, если используются;
- `options_source` загружает динамические варианты, кэшируется и обновляется кнопкой `Обновить список`;
- `checkboxes` переносится на несколько строк и не ломает ширину окна;
- `min_selected: 1` блокирует запуск, если пользователь снял все флажки;
- выбранные значения видны в `context.operation.parameters` или в итоговой CLI-команде;
- пустые строки сохраняются, если они означают auto/default.
- схожие выборы на форме стоят рядом и читаются как один блок;
- маленький фиксированный single-choice не спрятан в dropdown без причины;
- нет двух почти одинаковых кнопок запуска для одного пользовательского результата;
- нет дублирующих controls избранного для одного и того же списка;
- нет оторванных команд без объекта: `В избранное`, `Сохранить`, `Проверить`, `Удалить` должны быть переименованы или визуально привязаны к единственному полю;
- второстепенные параметры не отодвигают основной запуск ниже видимой области.
- редкие параметры можно свернуть в `Дополнительно`, а состояние блока запоминается, если проект это поддерживает.

PowerShell / CLI Window Check

Если проект вызывает `pwsh.exe`, `powershell.exe`, Office COM helpers, ffmpeg или другой дочерний CLI:

- запуск через GUI не должен создавать всплывающие консольные окна на каждый файл;
- `-WindowStyle Hidden` не считается достаточной защитой;
- проверьте, что subprocess использует `run_process()` или `STARTUPINFO/SW_HIDE` и `CREATE_NO_WINDOW`;
- вывод дочерней команды должен идти в правый GUI-терминал или лог, а не в отдельное пользовательское CLI-окно;
- после завершения операции зеленый индикатор под терминалом остается видимым, даже если окно было неактивно.

NiceGUI ProcessPool Fallback

В закрытых portable/sandbox окружениях NiceGUI может не создать multiprocessing process pool. GUI должен стартовать всё равно, если проект использует только обычные GUI-задачи и `run.io_bound`.

Проверка считается успешной, если `runtime\python.exe system_core\ui_nicegui\app.py --smoke` проходит, а серверный запуск не падает на `PermissionError` / `WinError 5`.

Для native picker dialogs PowerShell ищется в таком порядке:

1. `system_core\powershell\pwsh.exe`
2. `pwsh.exe` из `PATH`
3. встроенный `powershell.exe`

Наличие хотя бы одного варианта видно в секции `[GUI portability]` команды:

```
runtime\python.exe system_core\doctor.py
```