Audion Hub Manager

English · Руководство · Решения · История

Содержание

Портативная мастерская для проектов: видеть, что в проекте лежит, вести его историю в Git и держать чистое зеркало — без того, чтобы таскать за собой рантаймы, кэши, логи и локальные ключи.

Зачем это сделано

У живого проекта два несовместимых свойства. Он должен быть полным — со всеми временными файлами, сборочным мусором, настройками конкретной машины, иначе на нём нельзя работать. И он должен быть обозримым — чтобы историю можно было читать, изменения сравнивать, а копию восстанавливать, не разбирая, что здесь исходник, а что след вчерашней сборки.

Совместить это в одной папке не выходит. Поэтому проект живёт в трёх слоях:

```
Исходник     полный проект, единственный источник правды
Зеркало      отфильтрованная техническая копия со своим Git
Документы    человекочитаемый слой: заметки, описания, указатели
```

Зеркало можно удалить и собрать заново. Документы можно синхронизировать чем угодно — редактором, файловым менеджером, Obsidian, облаком или ничем. Исходник не меняется никогда.

Короткая формула проекта:

> Исходник остаётся целым. Зеркало остаётся обозримым. Документы остаются читаемыми. Git перестаёт быть страшным.

Принципы

Исходник — источник правды, и зеркало в него не пишет. Никогда, ни при каких настройках. Зеркало вправе удалять и пересобирать свои файлы, но со стороны исходника оно умеет только читать.

Настоящая запись требует явного намерения. По умолчанию зеркало показывает, что собирается сделать, и на этом останавливается. Чтобы оно действительно писало, нужно сказать это отдельным словом.

Фильтр не может молча стать полной копией. Если профиль требует масок, а масок нет — сборка не начинается, а падает с объяснением. Иначе однажды «отфильтрованное зеркало» окажется полным дублем проекта, и узнается это на диске, который кончился.

Ошибка копирования запрещает удаление. Безусловно, независимо от настроек. Это ровно тот случай, когда зеркало способно потерять файлы, существующие только в нём.

Это не хранилище секретов. Токены, пароли и ключи живут там, где им место: в диспетчере учётных данных, в ssh-agent, во внешних утилитах входа. Программа выполняет обычные команды `git` и показывает их вывод целиком — если вы уже вошли где-то ещё, всё работает и без её ведома.

История зеркала собирается по тому же списку, что и само зеркало. Никаких `git add .`: в коммит попадает только то, что пропустил бы профиль.

Как это выглядит в работе

Список проектов вверху переключает сразу весь комплект: исходник, зеркало, папку документов, профиль фильтрации и ветку по умолчанию. Переключение — одним движением, а не поиском по дереву.

Есть общая папка с двумя десятками проектов? Её не нужно делать одним проектом: сканирование найдёт настоящие корни внутри, включая вложенные вида `Проект/Проект`, и заведёт каждый отдельной записью.

Три кнопки слоя показывают, что именно вы сейчас смотрите — живой проект, зеркало или документы. Значок рядом с заголовком называет активный слой и его путь, чтобы не перепутать.

Правая половина окна — вкладки по видам работы: частые команды, ветки, редактор, сравнение, хранилище, удалённые репозитории, корзина коммита, чтение, история и подробности.

Что она умеет

Реестр проектов — вести список, сканировать общую папку, чистить записи с исчезнувшими путями и дубли, показывать состояние всех проектов разом: сколько чистых, сколько с изменениями, где ошибки.

Зеркало — предпросмотр и применение по профилю, проверка совпадения исходника и зеркала по контрольным суммам, восстановление проекта из зеркала, если исходник потерян.

Git — почти весь повседневный оборот: начать, посмотреть состояние, сравнить, подготовить и записать изменения, метки, ветки и переключение, отложенные правки, история и граф, восстановление, обслуживание, синхронизация с удалёнными. Редкое и опасное остаётся явными заготовками, которые надо осознанно выполнить.

Удалённые репозитории — сборка адреса кнопками (GitHub, GitLab, Forgejo, Gitea, свой сервер), отправка и приём, работа с несколькими удалёнными сразу, проверка входа без хранения токенов.

Редактор — чтение и правка Markdown и текста прямо в окне, с выходом в VS Code одним нажатием.

Дальше

- Руководство — работа по шагам: проекты, зеркало, Git, вход.
- Решения — почему три слоя, почему зеркало устроено так и почему программа не хранит пароли.
- История — что менялось от версии к версии.
- Git-вики — подробный разбор работы с Git.

Техническая часть

Запуск

```
launcher_gui.cmd
```

Проверка работоспособности без окна:

```
launcher_cli.cmd --mirror-preview demo_local --json
launcher_cli.cmd --mirror-apply demo_local --json
launcher_cli.cmd --mirror-apply demo_local --apply --json
```

Нет портативного рантайма — собрать:

```
install\Build_Portable_Env.cmd
```

Описание проекта

`config/projects.json`:

```
{
  "id": "audion_hub_manager",
  "title": "Audion Hub Manager",
  "source_path": ".",
  "projection_path": "S:/Audion/Hub Data/Audion Hub Manager",
  "docs_path": "S:/Audion/Docs/Projects/Audion Hub Manager",
  "profile": "audion_python_project_projection",
  "default_branch": "main"
}
```

поле	что
`source_path`	живой проект; относительный путь считается от корня программы
`projection_path`	зеркало, может иметь свой `.git` и пересобираться
`docs_path`	необязательный слой документов для чтения
`profile`	правила фильтрации из `config/projection_profiles.json`
`default_branch`	ветка по умолчанию для команд Git

Испорченный файл реестра больше не мешает запуску: программа откатывается к умолчаниям, сохраняет уцелевшие записи и пишет о проблеме в окно вывода.

Что где лежит

```
config/projects.json              реестр проектов
config/projection_profiles.json   профили фильтрации
config/forgejo_hosts.json         известные сервера Forgejo и Gitea, без токенов
logs/<project_id>/                отчёты сверки, датированные JSON
```

Слепки контрольных сумм строятся в памяти; ни исходник, ни зеркало, ни документы служебных файлов не получают.

Проверка

```
python -m compileall -q system_core
python -m pytest -q tests
python system_core\ui_nicegui\app.py --smoke
```

Опорное значение: `121 passed`.

Восстановление из зеркала

```
robocopy "<зеркало>\Audion Hub Manager" "<корень>\Audion Hub Manager" ^
  /E /COPY:DAT /DCOPY:DAT /R:1 /W:1 /XJ
```

После этого пересобрать рантайм или релизные сборки, если они нужны.

Правила, которые нельзя нарушать

1. Исходник — источник правды.
2. Зеркало производно: его можно удалить и собрать заново.
3. Зеркало не изменяет исходник.
4. `.git/**` защищён от сканирования, удаления и обслуживания.
5. Настоящая запись требует явного намерения.
6. Коммит идёт по тому же списку, что и зеркало.
7. Токены, пароли, приватные ключи и локальные пути не попадают в общие настройки.
8. Машинное держится в `*.local.json`, `.env` и прочих исключённых файлах.
9. Отчёты сверки пишутся только в `logs/<project_id>/`.
10. Слой документов отдельно не хэшируется: его техническая копия уже в зеркале.