Отлично, тогда добавим пункт про локальный просмотр сайта и уберём всякую информацию о скачиваемых файлах.

Вот обновлённая шпаргалка для твоего Word-файла:

**Шпаргалка: как пушить и деплоить TЗ на MkDocs + GitHub Pages**

**1️⃣ Сохраняем изменения локально**

* Редактируем Markdown-файлы, добавляем схемы, картинки, конспекты.
* Проверяем, что все изменения готовы.

**2️⃣ Проверяем сайт локально (без деплоя)**

mkdocs serve

**Что происходит:**

* MkDocs запускает локальный веб-сервер (по умолчанию [http://127.0.0.1:8000](http://127.0.0.1:8000/))
* Можно посмотреть, как будет выглядеть сайт, прежде чем деплоить.

**3️⃣ Пушим в main (репозиторий с исходниками)**

git add .

git commit -m "Краткое описание изменений"

git push

**Что происходит:**

* Все файлы (Markdown, схемы, конспекты) сохраняются в ветке main на GitHub.
* История коммитов обновляется.
* Можно откатиться к любой версии исходников, если что-то сломается.

**4️⃣ Деплой сайта (gh-pages)**

mkdocs gh-deploy

**Что происходит:**

* MkDocs собирает HTML/CSS/JS из docs/.
* Обновляет ветку gh-pages.
* Сайт на GitHub Pages отражает все изменения.

**5️⃣ Рекомендованная последовательность**

1. Редактируем и сохраняем файлы локально.
2. Проверяем локально: mkdocs serve
3. git add → git commit → git push (ветка main)
4. mkdocs gh-deploy (ветка gh-pages)

🔹 Совет: сначала пушим в main, потом деплой — безопасно для истории и исходников.

Если хочешь, я могу прямо сделать **мини-таблицу с командами и назначением**, чтобы Word-шпаргалка выглядела как компактная «кнопочная» памятка. Хочешь, чтобы я так сделал?