|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
|  |  |  |  |
|  |  |  | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
| GOSTDOWN+ | | | |
| Система документирования на Markdown | | | |
|  | | | |
|  |  |  | |
| Листов 49 | | | |

Реферат

Отчёт 49 с., 17 рис., 1 табл., 3 источн., 0 прил.

документирование, документация как код

Объект исследования – способы разработки документации как кода в соответствии с требованиями Заказчика, ГОСТов (в трактовке Заказчика) и необходимости предоставления результатов в формате docx.

Цели и задачи: разработать фреймворк для разработки и доставки документации в форматы docx и веб с возможностью хранения в git.

Содержание

[Введение 7](#_Toc117580029)

[1 Описание тулчейна 8](#_Toc117580030)

[1.1 Что вообще происходит 8](#_Toc117580031)

[1.2 Установка тулчейна 9](#_Toc117580032)

[1.2.1 Pandoc 10](#_Toc117580033)

[1.2.2 Python 10](#_Toc117580034)

[1.2.3 Gostdown 11](#_Toc117580035)

[1.2.4 (Optional) Mkdocs 13](#_Toc117580036)

[1.3 Все вместе 13](#_Toc117580037)

[2 Сборка docx из markdown 15](#_Toc117580038)

[2.1 Структура файлов и папок 15](#_Toc117580039)

[2.2 Пишем markdown с gostdown 17](#_Toc117580040)

[2.2.1 Настройка обработки 17](#_Toc117580041)

[2.2.2 Нумерация заголовков 18](#_Toc117580042)

[2.2.3 Рисунки 18](#_Toc117580043)

[2.2.4 Таблицы 19](#_Toc117580044)

[2.2.5 Источники 20](#_Toc117580045)

[2.3 Сборка документа 21](#_Toc117580046)

[2.3.1 Запуск скрипта 21](#_Toc117580047)

[2.3.2 Предобработка 22](#_Toc117580048)

[2.3.3 Постобработка 23](#_Toc117580049)

[3 Собираем веб-доку 29](#_Toc117580050)

[3.1 Превью и html 29](#_Toc117580051)

[3.2 Удаляем теги gostdown 31](#_Toc117580052)

[3.3 Полезные плагины и расширения 32](#_Toc117580053)

[3.3.1 Аббревиатуры 33](#_Toc117580054)

[3.3.2 Источники 33](#_Toc117580055)

[3.3.3 Сноски 34](#_Toc117580056)

[3.3.4 Спойлер 35](#_Toc117580057)

[3.3.5 Сниппеты 35](#_Toc117580058)

[3.3.6 Критик 36](#_Toc117580059)

[3.3.7 Формулы 37](#_Toc117580060)

[3.4 Генерация PDF 38](#_Toc117580061)

[3.4.1 С плагином mkdocs-with-pdf 38](#_Toc117580062)

[3.4.2 MkDocs PDF with JS Plugin 38](#_Toc117580063)

[4 VScode to rule them all 40](#_Toc117580064)

[4.1 Расширения 40](#_Toc117580065)

[4.2 Сниппеты 43](#_Toc117580066)

[4.3 Tasks 46](#_Toc117580067)

[4.4 Запускаем сборку документации 46](#_Toc117580068)

[4.5 Запускаем стриппер 47](#_Toc117580069)

[Заключение 48](#_Toc117580070)

[Список использованных источников 49](#_Toc117580071)

Обозначения и сокращения

|  |  |  |
| --- | --- | --- |
| НИР | — | Научно-исследовательская работа |
| СНС | — | Свёрточная нейронная сеть |
| CGP | — | (англ. *Class general perception*) |
| CPR | — | (англ. *Combined perception refinement*) группирует сообщения электронной почты в два класса: класс спама и класс обычных сообщений. Традиционный подход к фильтрации спама |
| ESP | — | (англ. *E-mail specific perception*) каждое электронное письмо рассматривается как отдельный класс, в отличие от модели CGP, где есть только два общих класса. Цель состоит нахождении сходства входящего электронного письма с отдельными сообщениями в наборе данных |
| PCA | — | (англ. *Principal component analysis*, Метод главных компонент) один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации |

Термины и определения

|  |  |  |
| --- | --- | --- |
| стемминг | — | процесс нахождения основы слова для заданного исходного слова |
| лемматизация | — | процесс приведения словоформы к лемме — её нормальной (словарной) форме |
| accuracy | — | доля верных ответов - отношение количества верных прогнозов о классе объекта к общему количеству прогнозов |
| MSER | — | (англ. Maximally Stable Extremal Region, Максимально стабильные экстремальные области) используется как метод обнаружения blob в изображениях. Этот метод был предложен для нахождения соответствия между элементами изображения из двух изображений с разными точками обзора |

Введение

Этот документ собран в качестве демонстрации возможностей фреймворка сборки документации на Gostdown (https://gitlab.iaaras.ru/iaaras/gostdown), костылях из скриптов Python, макросов Microsoft Word и VSCode. Фреймворк прошел успешные испытания на 7 проектах (4 НИРах, 2 НТО и 1 ПО). Возможно, кому-то этот фреймворк или отдельные его части пригодятся :).

# Описание тулчейна

Стильный модный молодежный подход DocOps – разработка документации как кода. Используем несколько инструментов (Gostdown, mkdocs, набор скриптов, макросов и задач VSCode) для автоматизации всего, чего можно.

## Что вообще происходит

При условии, что все установлено и настроено (не волнуйтесь, ниже я все расскажу!), происходит вот что:

1. Отдельные разделы и куски документов разрабатываем в 🌐[markdown](https://www.markdownguide.org/basic-syntax/). Альтернативы, о которых следует знать: asciidoctor, reStructeredText: более продвинутые из коробки и поддерживают single-source.
2. В директорию с файлами .md помещаем скрипт 🌐[gostdown](https://gitlab.iaaras.ru/iaaras/gostdown) и шаблон .docx документа. В шаблоне docx можно настроить рамки, титул, лист регистрации и основные надписи (например, нумерация страниц вверху листа по ГОСТ 19). В репозитории 🌐[gostdown](https://gitlab.iaaras.ru/iaaras/gostdown) есть подробное описание как это работает. Кроме того, там есть демо-файлы с примерами.
3. В файл .bat скрипта gostdown добавляем наши файлы .md в том порядке, в котором хотим видеть содержимое в финальном документе.
4. Запускаем скрипт. Скрипт все собирает, обрабатывает и вы получаете - docx и (опционально) pdf.

Все. По замыслу разработчиков скрипта gostdown никаких доработок в документе не требуется. Однако, есть ряд проблем: таблицы и рисунки нумеруются не автоматической нумерацией, ссылки вставлены гиперссылками, а не перекрестными ссылками. Некоторое время мне этого хватало, но потом появились заказчики, которым страшно нужна была автоматическая нумерация. К счастью, эта задача относительно просто решается VBA-макросом.

Также можно собрать веб-документацию 🌐[mkdocsом](https://squidfunk.github.io/mkdocs-material/getting-started/).

1. В корень с нашими markdown файлами помещаем конфиг mkdocs.yml.
2. Выполняем команду mkdocs server в папке с mkdocs.yml и нашими файлами. Красивый сайт будет доступен по адресу, указанному в командной строке.
3. Выполняем команду mkdocs build – генерируется html из доки, html можно передавать заказчику и выкладывать на веб-сервер.

Вот примерно такой workflow. Получаем на выходе и доку по ГОСТ, и онлайновую доку. Кроме того, markdown файлы можно спокойно хранить в репозитории гит и показывать разработчикам :) А разработчики могут туда даже что-то писать. Это проще, чем просить отредактировать Word, не говоря уже о плюсах контроля версий и изменений: всегда можно посмотреть, когда что поменялось, и откатиться назад.

Приступим к установке и настройке всей этой красоты.

## Установка тулчейна

Итак, чтобы вся эта красота заработала нам нужно установить:

* pandoc и плагин к нему pandoc-crossref;
* python;
* gostdown;
* mkdocs;
* скачать и установить шрифты [PT Serif, PT Sans и PT Mono](https://www.paratype.ru/public/). Можно и без них, если в шаблоне gostdown измените шрифт на тот, что нравится.

В репозитории вот здесь представлен демо-проект с текстом вот этой статьи, на котором можно ставить эксперименты и проверять, что и как работает (или не работает :)). Также для автоматической установки pandoc и pandoc-crossref нужных версий мой коллега @[nicky1038](https://github.com/nicky1038) написал скрипт install-toolchain.ps1 (также лежит в репозитории).

### Pandoc

[Pandoc](https://pandoc.org/) – это универсальная утилита для работы с текстовыми форматами. Основная сфера применения – форматирование математических и технических текстов, говорит нам Википедия. Pandoc удобно использовать для конвертирования между форматами, например, docx в markdown и наоборот.

1. Установить 📥[pandoc](https://github.com/jgm/pandoc/releases/). На момент написания статьи последняя версия – 2.18 (релиз 04.04.2022), именно эту версию я использую.
2. Скачать 📥[pandoc-crossref](https://github.com/lierdakil/pandoc-crossref/releases). Я использую версию 📥[0.3.13.0](https://github.com/lierdakil/pandoc-crossref/releases/tag/v0.3.13.0) для pandoc 2.18. Нужно иметь в виду, что последняя версия gostdown была выпущена год назад, так что с новыми версиями pandoc и pandoc-crossref могут быть ошибки. Стабильная версия: pandoc [2.17.1](https://github.com/jgm/pandoc/releases/tag/2.17.1) и pandoc-crossref [v0.3.12.1a](https://github.com/lierdakil/pandoc-crossref/releases/tag/v0.3.12.1a). Короче говоря, можете ставить версии pandoc 2.18 и pandoc-crossref 0.3.13.0, а если возникнут ошибки – в первую очередь попробовать старую версию. У меня пока ошибок не было.
3. Распаковать pandoc-crossref в папку pandoc: %localappdata%\Pandoc (положить туда exeшник).
4. Проверить версию pandoс (и что он правильно прописался в PATH). Пуск > Поиск > powershell > запустить Powershell. В powershell ввести команду:

pandoc --version  
  
pandoc.exe 2.18  
Compiled with pandoc-types 1.22.2, texmath 0.12.5, skylighting 0.12.3,  
citeproc 0.7, ipynb 0.2, hslua 2.2.0  
Scripting engine: Lua 5.4  
User data directory: C:\Users\Anna\AppData\Roaming\pandoc  
Copyright (C) 2006-2022 John MacFarlane. Web: https://pandoc.org  
This is free software; see the source for copying conditions. There is no  
warranty, not even for merchantability or fitness for a particular purpose.

### Python

Python нам понадобится, чтобы использовать mkdocs (который написан на Python и js) и вспомогательные скрипты. Ну и вообще, Python везде пригодится. Все в этой статье написано для Python 3.10. Ставится Python очень просто: загрузите 📥[инсталлятор](https://www.python.org/downloads/) с сайта и запустите. Не снимайте галочку с опции про прописать путь в PATH.

Проверьте, что все установилось:

python --version

Чтобы не городить огород из библиотек, предлагаю сразу использовать виртуальную среду. Виртуальная среда для Python позволяется вам устанавливать все нужные пакеты в отдельную папку, и из нее запускать обособленный экземпляр Python. Это еще удобно тем, что если что-то пошло не так, то не нужно переставлять весь питон, искать способы переустановки библиотек, а можно просто снести папку и сделать все заново.

Все команды выполняем в PowerShell консоли:

1. Создать папку для нашего mkdocs.
2. Открыть созданную папку, кликнуть Shift + ПКМ по пустой папке и выбрать «Открыть окно PowerShell» здесь.
3. Выполнить команду: python -m venv .venv. В папке будет создана директория .venv со средой исполнения Python.
4. Чтобы выполнять команды в этой среде, устанавливать сюда пакеты и вообще с ней работать, нужно её активировать. Не забывайте про это! Выполните: .\.venv\Scripts\activate.

Должно получиться вот так. В скобках указана наша виртуальная среда, все команды теперь выполняем тут.

PS D:\virtualenv\mkdocs> .\.venv\Scripts\activate  
(.venv) PS D:\virtualenv\mkdocs>

### Gostdown

Gostdown не нужно устанавливать, достаточно скачать код из 📥[официального репозитория](https://gitlab.iaaras.ru/iaaras/gostdown). Вы получите кучу файлов и папок, давайте разберемся, что где и зачем (Таблица 1).

Таблица  – Описание файлов

|  |  |
| --- | --- |
| Что | Зачем |
| demo-report-beginning.md | это пример начала отчета, здесь есть пример списка исполнителей, список терминов, определений, реферат, введение |
| demo-report-end.md | это пример заключительного файла с заключением, списком источников и приложениями |
| demo-template-espd.docx | **обязательный файл** – это файл-шаблон, хоты бы один такой файл должен быть в проекте. Скрипт использует этот файл, чтобы к тексту применять стили. Не меняйте названия стилей. Вы можете отредактировать этот файл (например, добавить титул, добавить рамки, поменять лист регистрации, добавить что угодно до или после %MAINTEXT% < именно сюда будет вставлен текст |
| demo-template-report.docx | аналогично – файл-шаблон, но для отчета, а то был для ГОСТ 19 |
| drracket-screenshot.png | просто картинка пример |
| gost-r-7-0-5-2008-numeric-iaa.csl | это файл стилей для списка источников. Если в проекте есть список источников, должен быть в директории проекта |
| iaa-logo.emf | еще один пример картинки |
| linebreaks.lua | обязательный файл – разрывы в коде, должен быть в проекте |
| lunokhod-expo.jpg | еще одна картинка, пример |
| oc-plot.emf | еще одна картинка, пример |
| build.ps1 | обязательный файл – это и есть основной скрипт, который творит всю магию |
| build-demo-espd.bat | это командный файл, который запускает ps1 с аргументами. В аргументах указывается какой шаблон использовать (template.docx), нужно включать шрифты в файл, нужно ли делать pdf и, самое главное, – список md файлов в том порядке, котором хотим их видеть в результирующем docx |
| build-demo-report.bat | просто еще один пример командного файла |
| build-docs.bat | хотя бы один такой файл должен быть обязательно в проекте (ну, если вы не хотите вручную команды вводить |
| demo.bib | это список источников в формате BibTex. Обязательно должен быть в проекте, если хотите на что-то ссылаться |
| demo-espd-beginning.md | просто пример markdown начала для ГОСТ 19 |
| demo-espd-end.md | просто окончания markdown начала для ГОСТ 19 |
| demo-main.md | это основной текст. Хотя бы один md должен быть в проекте |
| demo-report.pdf | пример собранного отчета. docx получается большой, если включена в bat опция embedfont |

**Обязательно загляните в demo-report.pdf и demo-main.md!** Там есть все инструкции, как писать какие-то штуки в markdown (например, таблицы, ссылки на рисунки, формулы) и как оно будет выглядеть в результирующем файле.

### (Optional) Mkdocs

Mkdocs библиотеки:

* 📥[mkdocs-material](https://squidfunk.github.io/mkdocs-material/getting-started/)
* 📥[mkdocs-bibtex](https://pypi.org/project/mkdocs-bibtex/)

В нашей виртуальной среде выполняем команды:

pip install mkdocs-material  
pip install mkdocs-bibtex

🌐[Настройки](https://squidfunk.github.io/mkdocs-material/creating-your-site/#configuration) проекта хранятся в файле mkdocs.yml.

## Все вместе

Полный воркфлоу на примере тестового репозитория.

1. Запустить файл \docs\_gost\build-docs.bat. Убедиться, что в .bat файле указаны правильные пути к вашей установленной среде Python. Скрипт автоматически скопирует файлы картинок, чтобы в документе все правильно отобразилось.
2. Убедиться, что в директории \docs\_gost появились файлы report.pdf, report.docx.
3. Открыть .docx файл и запустить последовательно [макросы](docs_gost\02_md_to_docx\03_build_doc_macro_final.md): FigCapAutoNum, FigReferenceAutoInsert, TblCapAutoNum, TblReferenceAutoInsert. Эти макросы должны быть предварительно сохранены в шаблон Normal.dot.
4. Убедиться, что в документе теперь есть автоматическая нумерация.

Теперь делаем веб-доку:

1. Запустить скрипт stripper.py с командного файла \scripts\_and\_macros\run\_stripper.bat. Убедиться, что в .bat файле указаны правильные пути к вашей установленной среде Python.
2. Убедиться, что в корне demo\_gostdown создана директория docs.
3. Из корня demo\_gostdown запустить mkdocs\_serve.bat. Предварительно, убедитесь, что в activate.bat в переменной VIRTUAL\_ENV указан путь к виртуальной среде с установленным mkdocs и нужными плагинами (например, set VIRTUAL\_ENV=D:\virtualenv\mkdocs\.venv).
4. В консоли отобразится процесс запуска и в конце будет выведен адрес для подключения (http://127.0.0.1:8000/). Открыть адрес в браузере и полюбоваться на документацию.
5. Завершить исполнение командного файла (Ctrl + C).
6. Запустить mkdocs\_build.bat. В корневой директории появится собранная веб-документация в папке /site.

# Сборка docx из markdown

Все подготовлено, теперь можно писать текст и собирать красивую документацию. Сначала рассмотрим возможную структуру папок, потом несколько примеров, как что делать в gostdown и, наконец, как все собрать и обработать.

## Структура файлов и папок

Структура папок по большому счету не имеет большого значения, однако нужно принимать во внимание особенности скрипта gostdown:

* пути к файлам в .bat должны быть указаны относительно ps1/bat скрипта;
* пути к изображениям также должны быть указаны относительно скрипта, пути относительно самого файла .md скрипт не понимает.

Все .md файлы лежат в директории /docs или /docs\_gost. В /docs на каждый раздел документа. Когда я точно знаю, что веб-документация не нужна, я не создаю папку docs, она нужна только для работы mkdocs.

В корне проекта лежат:

* build.ps1, linebreaks.lua – сам скрипт и вспомогательный файл;
* build-docs.bat – это командный файл. Если вам нужно добавить еще один файл md в сборку, вам нужно его отредактировать (да, там длинный список, внимательно!);
* sources.bib – здесь перечислены источники (литература), а файл gost-r-7-0-5-2008-numeric-iaa.csl – отвечает за правильное отображение списка литературы. Он просто должен лежать тут;
* template-report.docx – это шаблон. Если вам нужно отредактировать титул, надписи (например рамки вставить), добавить список исполнителей – все это делаем здесь;
* папка \_img – у gostdown есть недостаток – он плохо обрабатывает относительные пути от файлов. Поэтому, все картинки должны размещаться по пути, расположенному от файла build-docs.bat.

Пример структуры папок одного из моих отчетов:

.  
├── docs  
├─── docs\01\_recognition  
├─── docs\03\_geoposition  
├─── docs\04\_behaviour  
├─── docs\05\_spam  
├─── docs\06\_growth\_trends  
├─── docs\07\_hardware  
├─── docs\08\_alg  
├─── docs\appendix  
├── docs\report-beginning.md  
├── docs\report-end.md  
├── \_img  
├── build-docs.bat  
├── build.ps1  
├── gost-r-7-0-5-2008-numeric-iaa.csl  
├── linebreaks.lua  
├── photon-docs-nir22.code-workspace  
├── report.docx  
├── sources.bib  
└── template-report.docx

Пример структуры с файлами (первые две директории):

.  
└── docs  
 ├── docs\01\_recognition  
 │ ├── docs\01\_recognition\img  
 │ ├── docs\01\_recognition\01\_rcg\_intro.md  
 │ ├── docs\01\_recognition\02\_rcg\_objects.md  
 │ ├── docs\01\_recognition\03\_impl.md  
 │ └── docs\01\_recognition\rcg\_summary.md  
 └── docs\03\_geoposition  
 ├── docs\03\_geoposition\img  
 ├── docs\03\_geoposition\01\_geo\_intro.md  
 ├── docs\03\_geoposition\02\_geo\_ner.md  
 ├── docs\03\_geoposition\03\_geo\_image.md  
 ├── docs\03\_geoposition\04\_geo\_impl.md  
 └── docs\03\_geoposition\05\_geo\_osm.md

Кстати, так как мне часто приходится создавать типовые документы с типовой структурой, то создание такой структуры директорий также можно автоматизировать. Для примера в репозитории в папке /scripts\_and\_macros лежит скрипт md\_creator.py и конфигурационный файл к нему – md\_creator\_config.json. В конфигурационном файле указывается структура директорий, типовое содержимое разделов и количество повторений для файлов. Например, в разделе (folders, часто – раздел ТЗ), есть несколько файлов (top\_files, подпункты ТЗ), и каждый файл создается с нужным количеством префиксов. Получается структура вида:

- Раздел (/razdel)  
-- Подраздел (/podrazdel)  
----0100\_punkt\_tz.md  
----0101\_device1.md  
----0102\_device2.md  
... и т.д.

Запустить скрипт можно командой: python .\md\_creator.py. По умолчанию будет создана директорий из конфига md\_creator\_config.json с родительской папкой docs\_gost. Чтобы указать свои параметры, запустить скрипт командой python .\md\_creator.py -cf "md\_creator\_config.json" -fn "docs".

## Пишем markdown с gostdown

Все возможные примеры, которые вам могут понадобиться есть в репозитории Gostdown в файле [демке](https://gitlab.iaaras.ru/iaaras/gostdown/-/blob/master/demo-main.md), и файле [pdf](https://gitlab.iaaras.ru/iaaras/gostdown/-/jobs/14344/artifacts/file/demo-report.pdf) (как это все отображается).

Ниже я описала операции, которые мне нужны чаще всего.

### Настройка обработки

В самом первом обрабатываемом файле .md (в данном случае это report-beginning.md) должен быть блок с настройками обработки файлов. Вот как он выглядит:

---  
chapters: false  
figureTitle: Рисунок  
tableTitle: Таблица  
tableEqns: true  
titleDelim: "&nbsp;–"   
link-citations: true  
linkReferences: true  
csl: gost-r-7-0-5-2008-numeric-iaa.csl  
bibliography: sources.bib  
...

Это стандартный блок для любых отчетов, можете смело его использовать без изменений. Из важных настроек тут:

* csl – название файла со стилем списка литературы (должен быть в корне);
* bibliography – название файла со списком литературы.

Если у нас не отчет, а просто дока (гост 19, 34), то шапка будет вот такой:

---  
chapters: false  
figureTitle: Рисунок  
tableTitle: Таблица  
tableEqns: true  
titleDelim: "&nbsp;–"   
link-citations: true  
linkReferences: true  
...

### Нумерация заголовков

Ничего особенного для этого делать не нужно, заголовки первого уровня (#) становятся заголовками Заголовок 1. Это нужно помнить, когда какой-то раздел разбиваете на несколько файлов.

### Рисунки

Вот как выглядит вставка рисунка в обычном markdown:

![Пример полносвязной нейронной сети](\_img/rcg\_fcnn.png)

А вот так нужно сделать, чтобы получился gostdown. Обратите внимание, что пробел перед {#fig:fully\_connected\_neural\_network} отсутствует (если будет пробел – то скрипт будет ругаться, что на нашел куда ссылаться).

![Пример полносвязной нейронной сети](\_img/rcg\_fcnn.png){#fig:fully\_connected\_neural\_network}

А вот ссылка на рисунок выше в тексте:

Изначально использовались полносвязные нейронные сети (англ. fully connected),   
то есть сети, в который каждый нейрон с одного слоя связан со всеми нейронами   
следующего слоя (см. рисунок [-@fig:fully\_connected\_neural\_network]).  
  
![Пример полносвязной нейронной сети](\_img/rcg\_fcnn.png){#fig:fully\_connected\_neural\_network}

Чтобы Gostdown понимал, как обращаться с рисунками, необходимо чтобы все ссылки на рисунки были указаны относительно .bat и .ps1 (командного файла, который собирает доку).

Рассмотрим структуру:

/docs  
-- /\_img  
---- some\_image.png  
-- example.md  
build.bat  
build.ps1

В example.md присутствует ссылка на рисунок:

Вот какой-то красивый рисунок [-@fig:some\_image\_id]`.  
  
![Красивый рисунок](\_img/some\_image.png){@fig:some\_image\_id}

В этом случае build.ps1 не найдет картинку, так как будет искать папку \_img относительно себя любимого, а не относительно файла example.md. Чтобы картинка отобразилась, нужно сделать вот так:

/docs  
-- example.md  
/\_img  
--some\_image.png  
build.bat  
build.ps1

Вот, теперь скрипт сможет найти картинку. Если хочется, чтобы картинке отображались и в редакторе, то можно папки с картинками дублировать вот так:

/docs  
-- /\_img  
---- some\_image.png  
-- example.md  
/\_img  
---some\_image.png  
build.bat  
build.ps1

Теперь скрипт сможет все собрать, и картинки отобразятся в редакторе.

### Таблицы

Название таблицы пишем через строку после нее вот так вот:

| Заголовок | Заголовок | Заголовок |  
|--------------|--------------|--------------|  
|Ячейка 1|Ячейка 2|Ячейка3|  
  
Table: Название таблицы {#tbl:table\_id}

Обратите внимание на пробел после названия. Ссылаемся также, как и для рисунков.

Информация в таблице [-@tbl:table\_id].  
  
| Заголовок | Заголовок | Заголовок |  
|--------------|--------------|--------------|  
|Ячейка 1|Ячейка 2|Ячейка3|  
  
Table: Название таблицы {#tbl:table\_id}

### Источники

Все источники записываются в файл .bib в корне проекта. Я чаще всего использую два типа BibTex: article и online. Пример файла bibtex приложен репозитории.

# онлайн источник. Обратите внимание на "note = russian", обязательно добавляйте его в онлайн источники.  
@online{rcg\_ntw\_intel,  
 author = {{Корпорация Intel}},  
 title = {{Что такое машинное зрение?}},  
 url = {https://www.intel.ru/content/www/ru/ru/manufacturing/what-is-machine-vision.html},  
 note = {russian},  
 urldate = {2022-02-18}  
}  
  
# Статья. Если "note = russian" не добавить, то все названия для этого источника в списке литературы будут на английском (например, Том = Volume, С. - Pages и т.ж.)  
# По ГОСТ по хорошему английские источники должны быть на английском (то есть "note = russian" добавлять не нужно), но иногда заказчики ругаются, так что можно и добавить.  
  
@article{beh\_survey2,  
 author = {Zamini, Mohamad and Hasheminejad, Seyed Mohammad Hossein},  
 year = {2019},  
 month = {04},  
 title = {A comprehensive survey of anomaly detection in banking, wireless sensor networks, social networks, and healthcare},  
 note = {russian},  
 journal = {Intelligent Decision Technologies}  
}

Чтобы сослаться на любой источник в тексте, нужно указать его вот так: [@beh\_survey2;@rcg\_ntw\_intel].

## Сборка документа

Время магии!

### Запуск скрипта

Документ собирается командным файлом .bat. Вот так он примерно выглядит внутри:

powershell.exe -command .\build.ps1 ^  
-md demo-report-beginning.md,demo-main.md,demo-report-end.md ^  
-template template-report.docx ^  
-pdf report.pdf ^  
-docx report.docx ^  
-embedfonts ^  
-counters  
pause

Параметры:

* -md – список файлов через запятую, содержимое которых будет записано в результирующий файл. Они будут обрабатываться именно в том порядке, в котором здесь указаны. Можно указывать относительный путь, например вот так: .\subfolder\examples.md;
* -template – файл docx, который будет использован в качестве шаблона
* -pdf – название результирующего pdf файла. Если pdf не нужен (он долго выводится достаточно), то просто уберите эту строчку
* -docx – название результирующего docx файла. Будет перезаписан, если существует
* -embedfonts – включать ли шрифты в файл. Если включать, то файл получиться большим (от 4 МБ), если включать не нужно - то удалите эту строчку
* -counters – счетчики - эта опция считает количество рисунков, таблиц, секций и нумерует все соответственно
* pause – это просто чтобы окошко не исчезало сразу, и можно было ошибки посмотреть. Опция -md может получиться очень длинной, так что внимательно.

Для сборки документов запустите отредактированный .bat файл, и рядом с .bat появятся (или перезапишутся) собранные готовые файлы.

### Предобработка

Как вы помните, в нашей структуре файлов в каждой папке с .md есть папка \_img, в которой хранятся картинки. В тексте .md ссылки на картинки - относительные, вида (\_img/file.png). Скрипту build.ps1 нужно, чтобы все картинки размещались относительно его самого, а не относительно файла .md. Простейший способ решить эту проблему – скопировать все картинки из дочерних \_img в корневую относительно build.ps1. Для этого написан простенький скрипт на python, который:

1. Получает все картинки .jpg, .png, .bmp (можете добавить в скрипт свои форматы и расширения).
2. Сохраняет список картинок в copied\_files.json. Если встретились картинки с одинаковыми названиями, перезаписывать не будет, так что все картинки должны быть с разными названиями.
3. Копирует картинки в корневую папку.
4. (Опционально) после сборки можно вызвать скрипт с флагом -r, чтобы удалить копии картинок из корневой \_img (чтобы не было дублирования).

Вызов скрипта можно добавить прямо в build-docs.bat:

@echo off  
"C:\Python\Python310\python.exe" %~dp0\..\scripts\_and\_macros\copy\_img.py -s "..\docs\_gost" -d "..\docs\_gost\\_img" -c  
@echo on  
  
powershell.exe -command .\build.ps1 ^  
-md .\report-beginning.md,.\01\_setup\01\_what\_is\_happening.md,.\01\_setup\02\_install.md,.\02\_md\_to\_docx\01\_folder\_structure.md,.\02\_md\_to\_docx\02\_gd\_examples.md,.\02\_md\_to\_docx\03\_build\_doc\_macro\_final.md,.\03\_md\_to\_web\01\_build\_web.md,.\03\_md\_to\_web\02\_remove\_gd\_tags.md,.\03\_md\_to\_web\03\_all\_together.md,.\03\_md\_to\_web\04\_useful\_plugins.md,.\03\_md\_to\_web\05\_pdf\_thoughts.md,.\04\_vscode\_settings\01\_setup\_workspace.md,.\04\_vscode\_settings\02\_tasks\_snippets.md,.\report-end.md ^  
-template template-report.docx ^  
-pdf report.pdf ^  
-docx report.docx ^  
-embedfonts ^  
-counters  
  
@echo off  
"C:\Python\Python310\python.exe" %~dp0\..\scripts\_and\_macros\copy\_img.py -r  
@echo on

### Постобработка

Хотя собранный документ по замыслу можно использовать прям как есть, все же требуется еще пара штрихов:

1. Автонумерация для рисунков и таблиц.
2. Перекрестные ссылки на нумерованные таблицы и рисунки.

Gostdown не использует в финальном файле автоматическую нумерацию, а вместо этого делает закладки и гиперссылки. Проблема тут в том, что если нужно отредактировать финальный файл или отдать его Заказчику для вставки в другой документ, то нумерация таблиц и рисунков поломается.

Для автоматической нумерации разработаны макросы, которые нужно добавить в шаблон Normal.dot.

Алгоритм макроса на автоматическую нумерацию:

1. Ищем в документе все Рисунок [0-9]{1;} и Таблица [0-9]{1;} (учитывается регистр), выделяем.
2. Заменяем на автоматическую нумерацию.

Аналогичного эффекта также можно добиться Поиском и заменой, если не хочется использовать макрос:

1. Вставить название рисунка или таблицы (с автонумерацией).
2. Правый клик по полю > Коды/значения полей (чтобы включить полное отображение поля).
3. Выделить «Название» с полем и Ctrl + C.
4. Вызывать диалог Find/Replace.
5. Отметить галку «Подстановочные знаки».
6. В поле «Найти» вставить: Рисунок [0-9]{1;} или (Рисунок ([0-9]{1;}) для группы), ну и Таблица для таблиц.
7. В поле «Заменить» Специальный > содержимое буфера обмена (или ^c).
8. Заменить все.
9. Обновить поля.

Код макроса:

Sub FigCapAutoNum()  
  
 Dim Rng As Range  
  
 Set Rng = ActiveDocument.Range  
  
 Do While Rng.Find.Execute(FindText:="Рисунок [0-9]{1;}", Forward:=True, Format:=False, Wrap:=wdFindStop, MatchWildcards:=True, MatchCase:=True) = True  
 Rng.MoveStart Unit:=wdCharacter, Count:=8  
 Rng.Fields.Add Range:=Rng, Type:=wdFieldEmpty, Text:="SEQ Рисунок \\* ARABIC", PreserveFormatting:=True  
 Loop  
  
 ActiveDocument.Fields.Update  
End Sub  
  
Sub TblCapAutoNum()  
  
 Dim Rng As Range  
  
 Set Rng = ActiveDocument.Range  
  
 Do While Rng.Find.Execute(FindText:="Таблица [0-9]{1;}", Forward:=True, Format:=False, Wrap:=wdFindStop, MatchWildcards:=True, MatchCase:=True) = True  
 Rng.MoveStart Unit:=wdCharacter, Count:=8  
 Rng.Fields.Add Range:=Rng, Type:=wdFieldEmpty, Text:="SEQ Таблица \\* ARABIC", PreserveFormatting:=True  
 Loop  
  
 ActiveDocument.Fields.Update  
End Sub

А вот с перекрестными ссылками так просто поиском и заменой уже не получится, поэтому только макрос. В макросе захардкожены ссылки на рисунки вида (рисунок N) с учетом регистра, т.е. в тексте это выглядит как на рисунке ниже (рисунок N). Алгоритм макроса:

1. Удаляем гиперссылки с текстом, содержащим fig: и tbl:.
2. Находим все (рисунок N)/(таблица N).
3. Обрезаем скобки.
4. Обрезаем номер рисунка/таблицы.
5. Заменяем текст на перекрестную ссылку функцией InsertCrossReference с номером, который выдрали на шаге 4.

Код макроса (убедитесь, что при вставке в Word VBA не побилась кодировка):

Sub FigReferenceAutoInsert()  
 Dim figReference As String  
   
 Dim oField As Field  
 For Each oField In ActiveDocument.Fields  
 If oField.Type = wdFieldHyperlink Then  
 If Mid(oField.Code, 16, 4) = "fig:" Then  
 oField.Unlink  
 End If  
 End If  
 Next  
 Set oField = Nothing  
   
 Dim Rng As Range  
 Set Rng = ActiveDocument.Range  
 Do While Rng.Find.Execute(FindText:="\(рисунок [0-9]{1;}\)", Forward:=True, Format:=False, Wrap:=wdFindStop, MatchWildcards:=True, MatchCase:=True) = True  
 figReference = Rng.Text  
 figReference = Mid(figReference, 10)  
 figReference = Replace(figReference, ")", "")  
 figInt = CInt(figReference)  
 figVar = CVar(figInt)  
 Rng.MoveStart Unit:=wdCharacter, Count:=1  
 Rng.MoveEnd Unit:=wdCharacter, Count:=-1  
 Rng.InsertCrossReference ReferenceType:="Рисунок", ReferenceKind:= \_  
 wdOnlyLabelAndNumber, ReferenceItem:=figVar, InsertAsHyperlink:=True, \_  
 IncludePosition:=False, SeparateNumbers:=False, SeparatorString:=" "  
 Loop  
 ActiveDocument.Fields.Update  
End Sub  
  
Sub TblReferenceAutoInsert()  
 Dim tblReference As String  
   
 Dim oField As Field  
 For Each oField In ActiveDocument.Fields  
 If oField.Type = wdFieldHyperlink Then  
 If Mid(oField.Code, 16, 4) = "tbl:" Then  
 oField.Unlink  
 End If  
 End If  
 Next  
 Set oField = Nothing  
   
 Dim Rng As Range  
 Set Rng = ActiveDocument.Range  
 Do While Rng.Find.Execute(FindText:="\(таблица [0-9]{1;}\)", Forward:=True, Format:=False, Wrap:=wdFindStop, MatchWildcards:=True, MatchCase:=True) = True  
 tblReference = Rng.Text  
 tblReference = Mid(tblReference, 10)  
 tblReference = Replace(tblReference, ")", "")  
 tblInt = CInt(tblReference)  
 tblVar = CVar(tblInt)  
 Rng.MoveStart Unit:=wdCharacter, Count:=1  
 Rng.MoveEnd Unit:=wdCharacter, Count:=-1  
 Rng.InsertCrossReference ReferenceType:="Таблица", ReferenceKind:= \_  
 wdOnlyLabelAndNumber, ReferenceItem:=tblVar, InsertAsHyperlink:=True, \_  
 IncludePosition:=False, SeparateNumbers:=False, SeparatorString:=" "  
 Loop  
 ActiveDocument.Fields.Update  
End Sub

Чтобы использовать этот код, необходимо в Microsoft Word активировать панель разработчика (Рисунок 1), вызвать лист VBA (Рисунок 2) и вставить код (Рисунок 3).

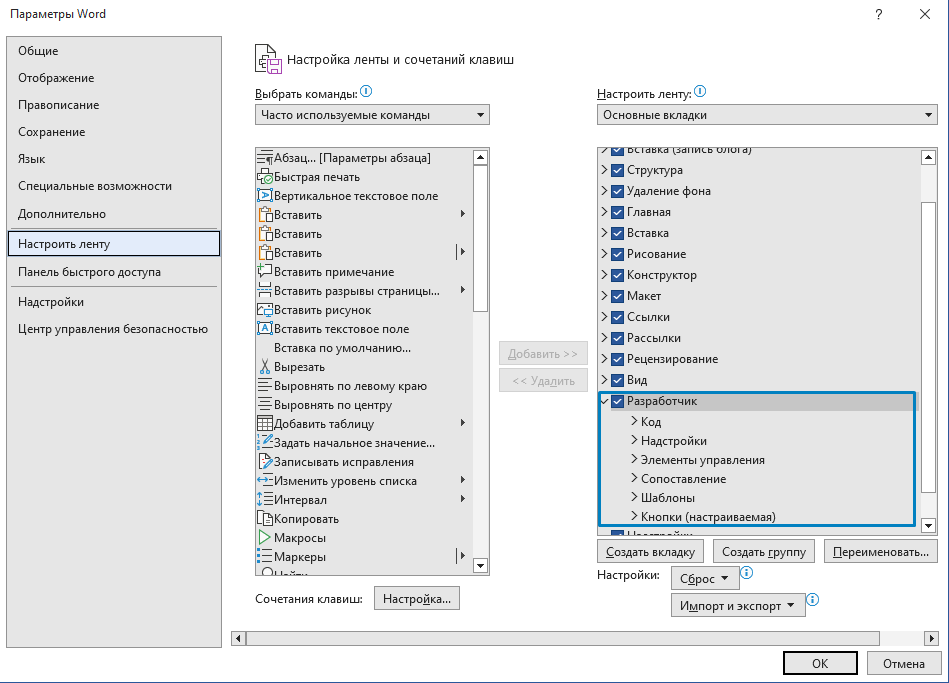


Рисунок  – Активация панели разработчика

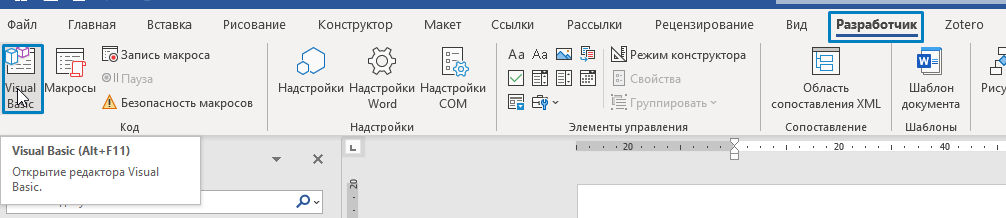


Рисунок  – Вызов листа с макросами

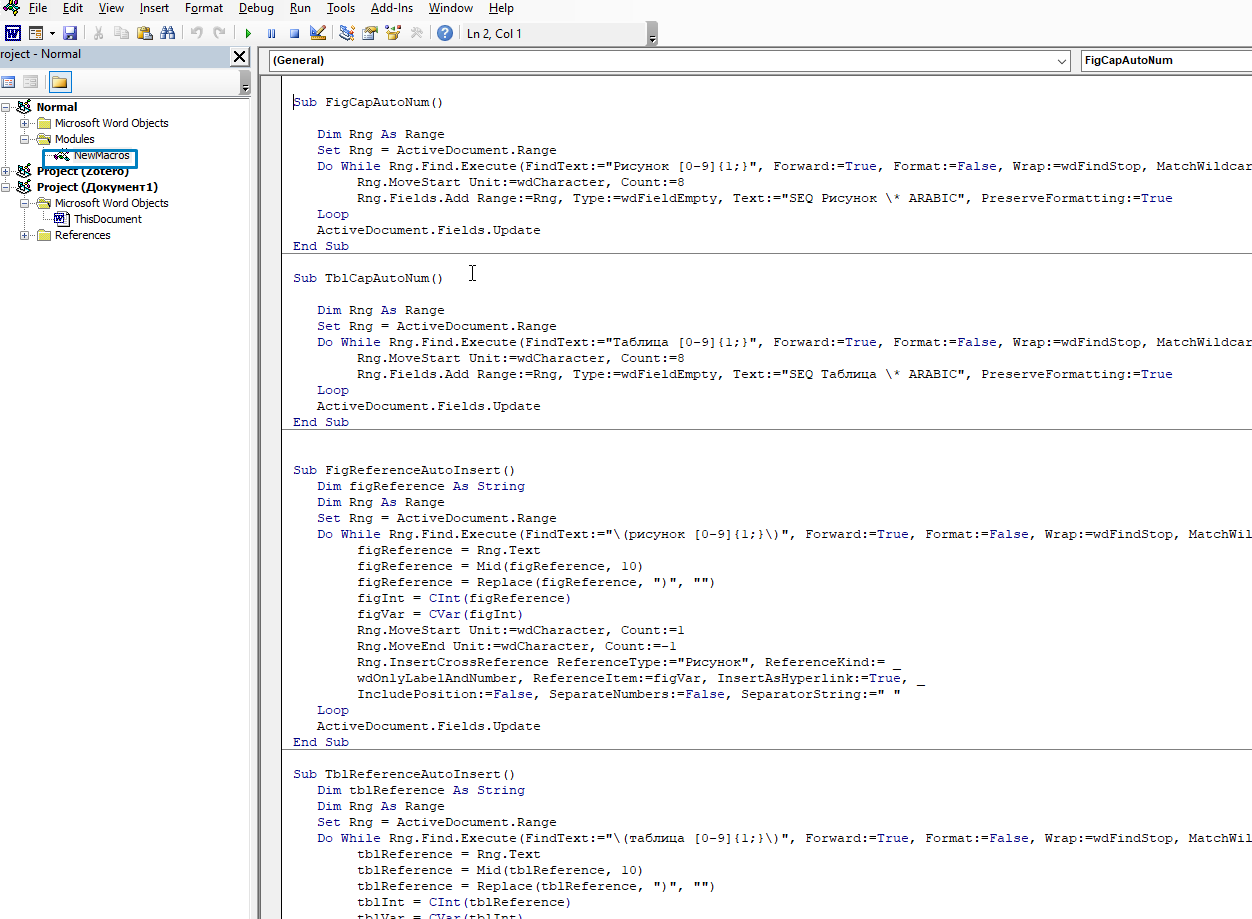


Рисунок  – Вставить код

А затем запустить макрос (Рисунок 4).

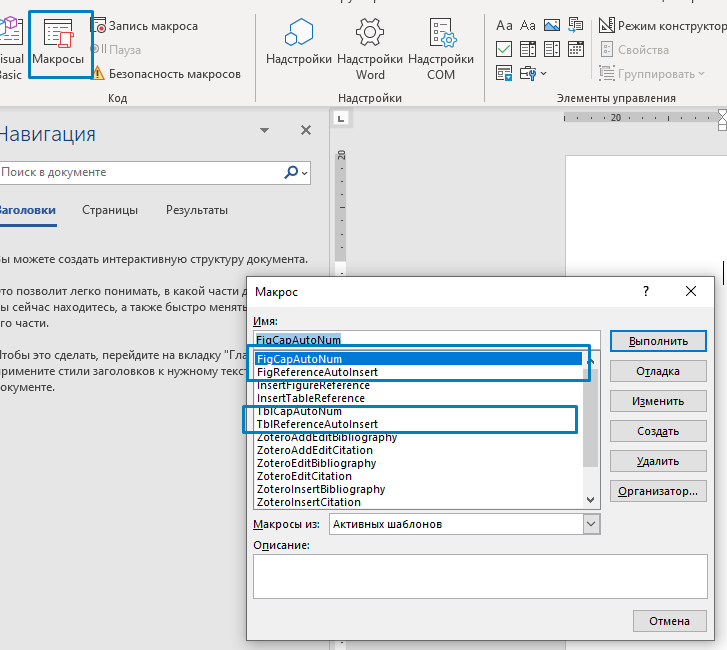


Рисунок  – Запуск макроса

Вот теперь у нас действительно классный docx.

# Собираем веб-доку

## Превью и html

Сборка выполняется 🌐[mkdocs-material](https://squidfunk.github.io/mkdocs-material/getting-started/), и по ссылке можно найти очень подробную документацию как сделать что угодно. Кроме того, mkdocs поддерживает огромное количество всяких плагинов, так что если что-то надо сделать необычное – скорее всего на это уже есть плагин. Кроме того, из mkdocs можно собирать PDF, хотя пока что мне это не пригодилось, поэтому инструкции не будет).

У Mkdocs есть требования к структуре (Рисунок 5):

* должна быть папка /docs, в которой все лежит;
* в папке /docs должен быть index.md файл;
* файлы .bib и .csl для источников также должны быть расположены относительно mkdocs.yml.

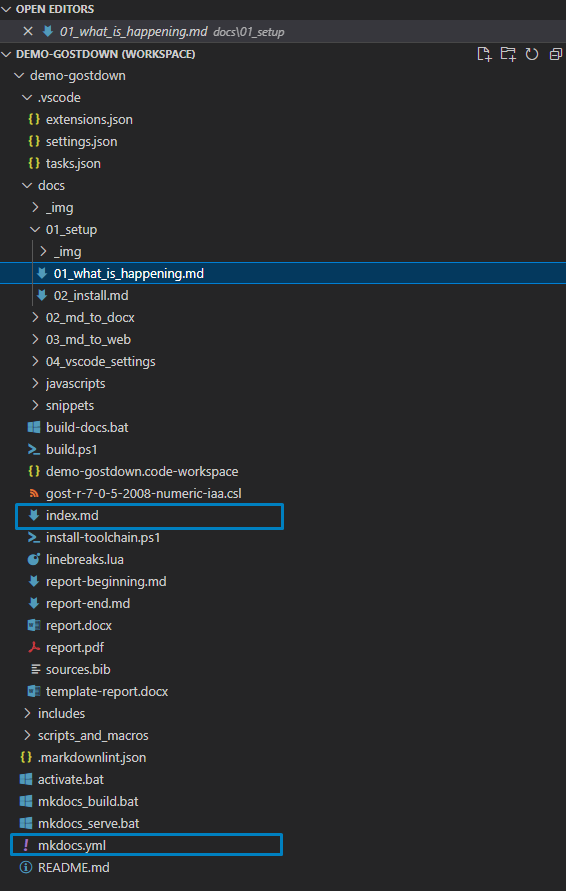


Рисунок  – Пример структуры в VSCode

В конфигурационном файле mkdocs.yml описана структура и плагины. В основном нужно редактировать четыре верхние строки (для нового проекта) и раздел nav. В разделе nav указаны пункты левого меню и файлы md, которые будут рендерится, когда перейдешь в это меню. Включите в своем редакторе отображение пробелов, так как yml очень капризен к отступам.

Собираем документ. В директории, где лежит mkdocs yml выполнить команду: mkdocs serve. Эта команда запустит маленький веб-сервер на 127.0.0.1:8000 и будет показывать наш сайт с докой. Если нужно собрать html из доки, нужно выполнить команду: mkdoc build.

## Удаляем теги gostdown

Чтобы использовать текст в формате Gostdown для веб-докуметации, нужно удалить некрасивые и ненужные теги для рисунков, таблиц, приложения и т.д. Один из моих коллег, эксперт в JavaScript предложил, что теги можно спрятать с помощью JavaScript. Однако я JavaScript знаю не так хорошо, как Python, а коллега свалил в Канаду, такие дела.

Поэтому вот вам скрипт на python, который берет все файлы .md из директории, за исключением проигнорированных, и удаляет из них паттерны Gostdown (названия и ссылки на рисунки ({#fig},[-@fig:]), таблицы (#tab},[-@tab:]), стили из заголовков custom-style}, {.unordered}, также удаляется брекеты [] из названий разделов. Не трогает: теги в названии Приложения (её руками нужно будет удалить), и ссылки на литературу ([\@litbib]), так как это универсальный формат, обрабатывается нормально mkdocs.

Аналогичные действия можно выполнить вручную через редактор (например Notepad++), инструмент Поиск и Замена по регулярным выражениям.

Сценарий использования: при разработке документов одновременно для docx (ЕСПД или ЕСКД) и веб-справки (mkdocs). Из файлов для gostdown делаем stripped версию скриптом – получаем веб-справку без лишних некрасивых тегов.

Список регулярных выражений и на что их заменять указан в коде, в переменной PATTERNS в виде списка. Если необходимо убирать какие-то еще паттерны, нужно просто добавить из в список по шаблону (pattern - регулярное выражение, replace - чем заменять). Пример списка:

PATTERNS = [  
 {"pattern": "{#(fig|tbl|sec):.+?(?=})}", "replace": "", "desc": "caption"},  
 {"pattern": "\[-@(fig|tbl|sec):.+?(?=])]",  
 "replace": "", "desc": "reference"},  
 {"pattern": "{custom-style.+?(?=})}", "replace": "",  
 "desc": "custom style named"},  
 {"pattern": "{\..+?(?=})}", "replace": "",  
 "desc": "custom style unnamed"},  
 {"pattern": "# \[([А-я].+?(?=\]))\]", "replace": r"# \1",  
 "desc": "headings brackets"}  
]

Запуск: из директории, в которой находится директория с файлами. Например:

img  
docs <- md файлы тут  
stripper.py  
build.bat  
..... any other files

Команда запуска для вывода хэлпа:

python .\stripper.py --help

Полная команда запуска в общем случае:

python .\stripper.py -s docs\_gost -d docs -i \*.png \*.bat \*.ps1 \*.lua end.md begin.md

* -s, --source – директория, которую копируем;
* -d, --destination – название директории, куда копируем;
* -i, --ignored – список игнорируемых файлов и расширения, передаются списком через пробел;
* -a, -abbreviations – если этот флаг указан, то дополнительно добавляет в каждый файл .md ссылку на файл с аббревиатурами.

## Полезные плагины и расширения

Все примеры ниже будут красиво отображаться при просмотре документации в mkdocs.

### Аббревиатуры

Расширение: [Abbreviations](https://squidfunk.github.io/mkdocs-material/setup/extensions/python-markdown/#abbreviations).

Установка: добавить в список markdown\_extensions ключ -abbr:

markdown\_extensions:  
 - abbr

Использование:

* Создать файл с аббревиатурами, например, includes/abbreviations.md вида:

\*[ДИВ]: деструктивное информационное воздействие  
\*[ПО]: программное обеспечение  
\*[СЗИ]: средство защиты информации  
\*[ДМЗ]: демилитаризованная зона

* В файле .md, к котором нужно использовать аббревиатуры, добавить в конце строку (без бэкслеша в начале строки):

\--8<-- "includes/abbreviations.md"

* В тексте использовать аббревиатуру как обычно, например, ПО (наведите мышку на аббревиатуру).

А на рисунке ниже (Рисунок 6) пример, как это выглядит в веб-документации.

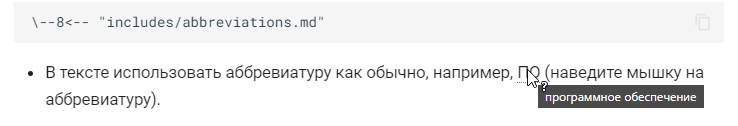


Рисунок  – Аббревиатуры в тексте

Кстати, если при запуске стриппера добавить аргумент -a, то он добавит в конец каждого файла ссылку на файл с аббревиатурами.

### Источники

Плагин: [mkdocs-bibtex](https://pypi.org/project/mkdocs-bibtex/).

Установка:

* Установить [pandoc](https://pandoc.org/installing.html).
* Выполнить команду:

pip install mkdocs-bibtex

Добавить плагин в список plugins:

yaml linenums="1" hl\_lines="3 4" plugins: - bibtex: bib\_file: "sources.bib" csl\_file: "gost-r-7-0-5-2008-numeric-iaa.csl"

где bibtex - название плагина, bib\_file - путь к файлу с источниками, csl\_file - ссылка на файл со стилями.

В тексте на источник ссылаться следующим образом: [ @test1] (без пробела), т.е. по его идентификатору. Например, бла-бла-бла согласно[[1](#ref-test1)] (электронный ресурс) или [[2](#ref-jia2014caffe)] (материалы конференции) или [[3](#ref-ren2015faster)] (статья).

Пример в тексте на рисунке ниже (Рисунок 7).

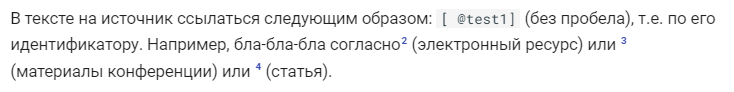


Рисунок  – Источники в тексте

### Сноски

Расширение: [Footnotes](https://squidfunk.github.io/mkdocs-material/setup/extensions/python-markdown/#footnotes)

Установка: добавить в список markdown\_extensions ключ -footnotes:

markdown\_extensions:  
 - footnotes

Использование: в тексте (например, в конце) указать сноску в формате:

[^f1]: Lorem ipsum dolor sit amet, consectetur adipiscing elit.

В тексте указать ссылку inline[[1]](#footnote-1). При использовании одновременно с bibtex и сноски, и источники будут отображены в порядке появления в тексте: сначала все сноски, потом все источники.

### Спойлер

Расширение: [Details](https://facelessuser.github.io/pymdown-extensions/extensions/details/).

Что делает: collapsable элементы.

Установка: добавить в список markdown\_extensions ключ -pymdownx.details.

Пример:

???+ note "Подробности"  
 Текст подробности

???+ note “Подробности” Текст подробности

Пример того, как это выглядит в тексте в веб-документации, приведен ниже (Рисунок 8).

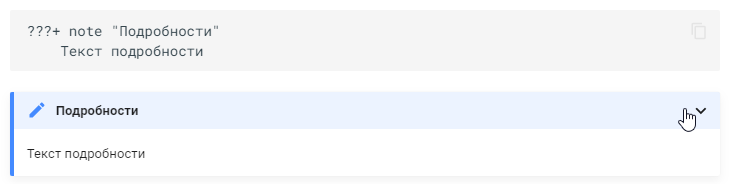


Рисунок  – Спойлер

### Сниппеты

Расширение: [Snippets](https://facelessuser.github.io/pymdown-extensions/extensions/snippets/).

Что делает: позволяет вставлять код из файла.

Установка: добавить в список markdown\_extensions ключ -pymdownx.snippets.

Использование:

yaml title="docs\snippets\example.js" --8<-- "docs\snippets\example.js"

Пример на рисунке ниже (Рисунок 9).



Рисунок  – Пример сниппета

### Критик

Расширение: [pymdownx.critic](https://facelessuser.github.io/pymdown-extensions/extensions/critic/).  
Что делает: позволяет в тексте проводить рецензирование.

Установка: добавить в список markdown\_extensions ключ -pymdownx.critic.

Использование: прямо в тексте в фигурных скобках указать {++то, что нужно вставить++}, {–то, что нужно удалить–} и то, что нужно {~изменить>заменить~~}. Также слова и фразы можно {==подсвечивать==}{>>А это комментарий<<}.

Вид можно настраивать ключом - mode для плагина. Доступные значения:

* view - подсветит все рецензии;
* accept - отобразит только добавленные и замененные, удалит удаленные изменения;
* reject - покажет текст без предложенных изменений.

Пример в тексте на рисунке ниже (Рисунок 10).

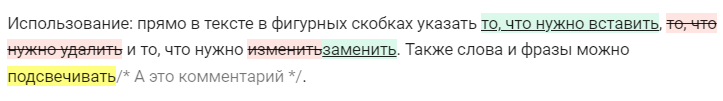


Рисунок  – Пример работы плагина Критик

### Формулы

Расширение: [Arithmatex](https://facelessuser.github.io/pymdown-extensions/extensions/arithmatex/). Много вариантов рендеринга формул. Здесь описан только один из возможных.

Установка: добавить в список markdown\_extensions ключ - pymdownx.arithmatex.

Формула в строку:

, (p(x|y) = ).

Блок:

[3 < 4]

Пример, как это выглядит в веб-документации, на рисунке ниже (Рисунок 11).

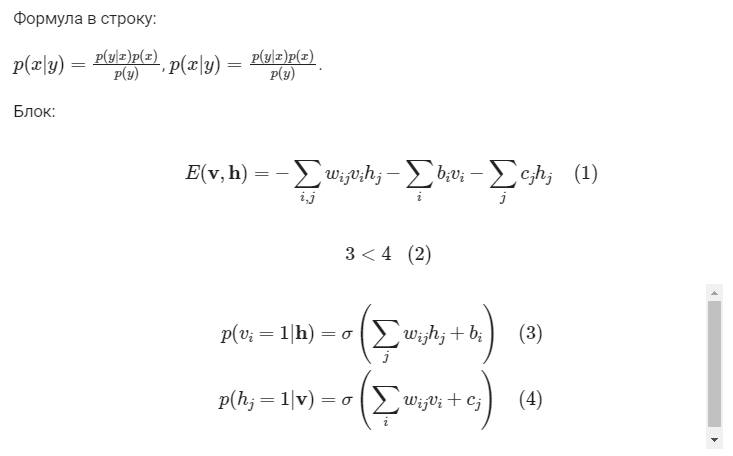


Рисунок  – Пример формул

## Генерация PDF

### С плагином mkdocs-with-pdf

Плагин: [mkdocs-with-pdf](https://github.com/orzih/mkdocs-with-pdf)

Установка:

* установить согласно инструкции [WeasyPring](https://doc.courtbouillon.org/weasyprint/stable/first_steps.html#windows) и зависимости;
* установить плагин pip install mkdocs-with-pdf.

*2022-05-12* Чтобы плагин не падал с ошибкой, если в доке есть диаграммы mermaid:

pip install beautifulsoup4==4.9.3  
pip install mkdocs-with-pdf

В конфигурационный файл mkdocs.yml добавить секцию:

plugins:  
 - with-pdf:  
 cover: true  
 cover\_title: Дока  
 toc\_title: Содержание  
 render\_js: true  
 headless\_chrome\_path: C:\Program Files\Google\Chrome\Application\chrome.exe  
 enabled\_if\_env: ENABLE\_PDF\_EXPORT

См. все ключи конфигурации в [README](https://github.com/orzih/mkdocs-with-pdf). С указанной конфигурации pdf не будет генерироваться во время mkdocs serve. Чтобы сгенерировать pdf во время mkdocs build, необходимо указать ENV:

$Env:ENABLE\_PDF\_EXPORT=1  
mkdocs serve

Все страницы будут склеены в один pdf в директории /site/pdf.

!!! warning Этот плагин не умеет генерировать диаграммы mermaid (они будут отображены блоком текста).

### MkDocs PDF with JS Plugin

Плагин: [mkdjcs-pdf-with-js-plugin](https://github.com/smaxtec/mkdocs-pdf-with-js-plugin)

В этом плагине не так много настроек, зато:

* он умеет отображать mermaid диаграммы;
* может генерировать pdf для конкретной страницы, а не все сразу.

Установка:

* установить браузер Chrome, [ChromeDriver](https://chromedriver.chromium.org/) нужной версии и указать путь к ChromeDriver в PATH;
* добавить в секцию plugins конфигурационного файла ключ - pdf-with-js. enable - true для активации плагина.

plugins:  
 ...  
 - pdf-with-js:  
 enable: true

# VScode to rule them all

Основной редактор, в котором я творю все вышеописанные безобразия, – VSCode. Интеграция с гитом из коробки, спеллчекинг (ну хоть какой-то), удобные линтер-плагины, рендеринг на лету и прочая красота.

Демо-репозиторий предоставляется с готовым файлов .workspace. Просто откройте его в VSCode (File > Open Workspace from File …) и часть настроек подхватится.

## Расширения

Расширения VSCode для работы с текстом:

* [Code Spell Checker](https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker) – проверяет орфографические ошибки. Я рекомендую добавлять неизвестные слова в Workspace Settings (ПКМ по слову и дальше разберетесь). Settings хранятся в корне директории в файле .code-workspace.
* [Russian Code Spell Checker](https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker-russian) – дополнение проверки с русским языком.
* [Live Preview](https://marketplace.visualstudio.com/items?itemName=ms-vscode.live-server)- рендерит markdown прямо в редакторе, для быстрого просмотра «печатного» вида файла.
* [Markdown all in one](https://marketplace.visualstudio.com/items?itemName=yzhang.markdown-all-in-one) – это набор инструментов для работы с markdown.
* [Markdown Table](https://marketplace.visualstudio.com/items?itemName=TakumiI.markdowntable) – использую отсюда одну единственную функцию – ПКМ по пустому месту в файле с таблицами > Format All Tables.
* [markdownlint](https://marketplace.visualstudio.com/items?itemName=DavidAnson.vscode-markdownlint) – это помощник по синтаксису markdown, чтобы все было красиво. Некоторые правила можно отключать. В корне проекта обычно лежит файл ..markdownlint.json, в котором перечислены правила, которые отключены. Чтобы узнать номер правила, наведите курсор на проблему – в появившемся тултипе будет написано.
* [Todo Tree](https://marketplace.visualstudio.com/items?itemName=Gruntfuggly.todo-tree) – удобный плагин для отслеживания #todo по всем файлам. В тексте просто пишете #TODO надо слепить снеговика и потом по всем файлам в панели Todo можно их просматривать. Режим отображения todo можно переключать (все todo списком, по файлам…). Пример на рисунке ниже (Рисунок 12).

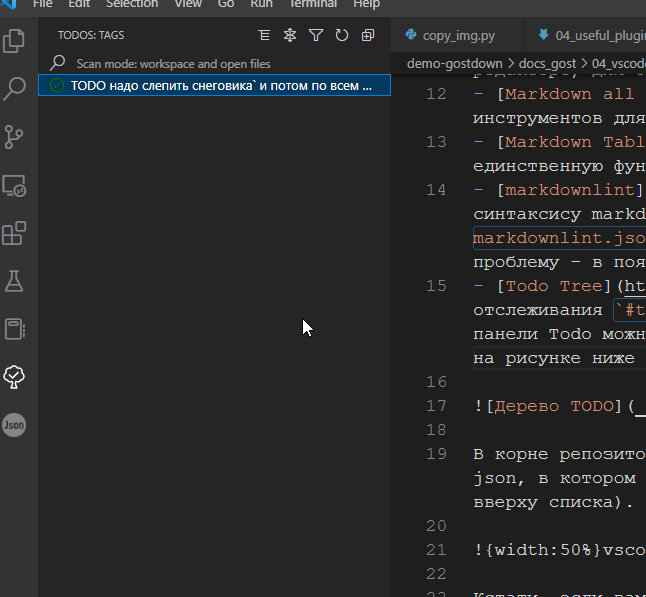


Рисунок  – Дерево TODO

В корне репозитория есть папка .vscode ее я тоже загружаю в репозиторий. Там в том числе хранится файл extensions.json, в котором все нужные расширения перечислены – их можно будет найти легко в магазине расширений (они будут прямо вверху списка). Вот как это будет выглядеть (Рисунок 13).

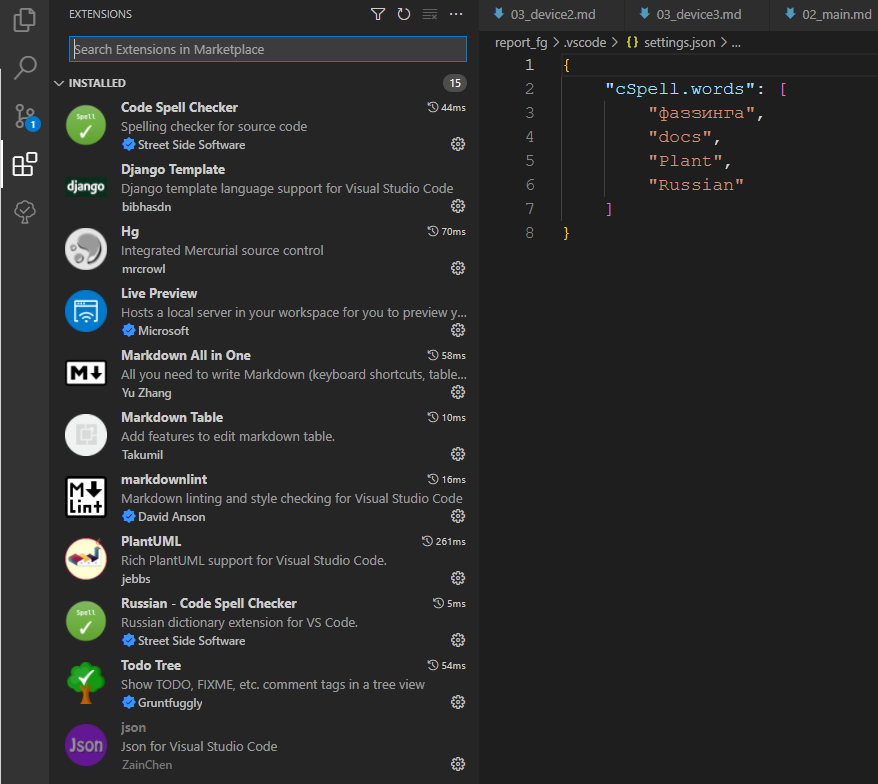


Рисунок  – Расширения VSCode

Кстати, если вам нравится рисовать схемы в PlantUML, то можете еще установить и PlantUML расширение, тогда вы сможете прямо в VScode их рендерить (Рисунок 14).

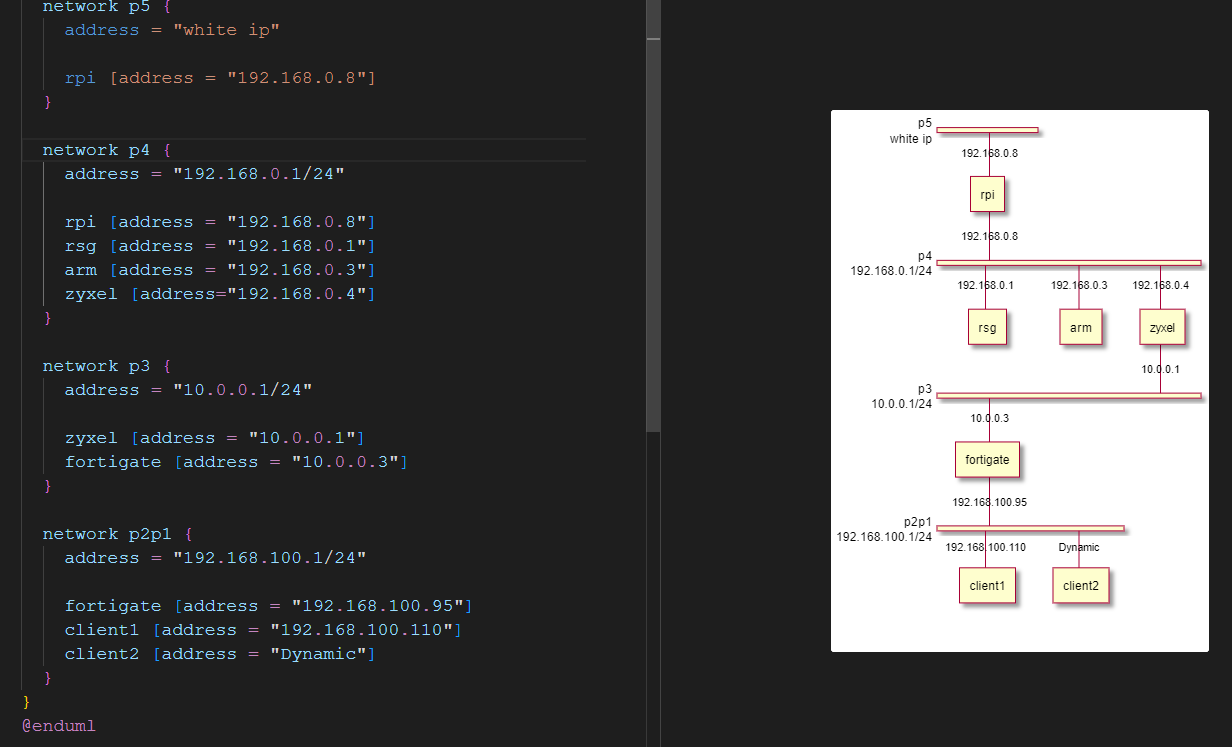


Рисунок  – PlantUML в VSCode

## Сниппеты

Сниппеты можно использовать для быстрой вставки кода. Я использую их для вставки gostdown элементов и кавычек-ёлочек. Как делать сниппеты достаточно подробно описано вот здесь: https://code.visualstudio.com/docs/editor/userdefinedsnippets. Все сниппеты хранятся в пользовательской директории %appdata%\Code\User\snippets\ в файле с названием языка, например markdown.json.

Например, вставим сниппет для добавления названия таблиц:

{   
 "Table": {  
 "prefix": [  
 "#tbl"  
 ],  
 "body": [  
 "Table: Имя таблицы {#tbl:table\_name}"  
 ],  
 "description": "Table reference"  
 },  
}

Теперь в тексте, под таблицей начинаем писать #t... и выбираем в появившейся подсказке #tbl. Автоматически будет вставлен шаблон подписи таблицы по Gostdown (Рисунок 15).

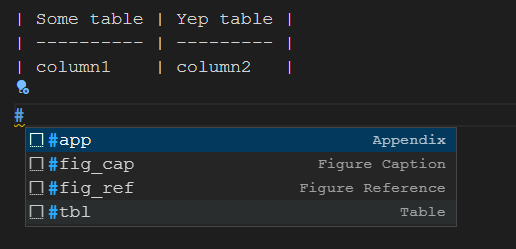


Рисунок  – Вставить подпись к таблице

Также в сниппетах можно использовать текст из буфера обмена (${CLIPBOARD}) и выделенный текст ($TM\_SELECTED\_TEXT). Например, можно выделить название рисунка из ссылки, скопировать, установить курсор в нужное место и вызвать сниппеты #fig\_cap и #fig\_ref (для ссылки на рисунок), см. пример на рисунке ниже (Рисунок 16).



Рисунок  – Сниппеты для рисунков

Также можно вставлять разные символы, например, кавычки-ёлочки. Вот сниппет для кавычек:

"French Quotes": {  
 "prefix": [  
 "<qu"  
 ],  
 "body": [  
 "«$TM\_SELECTED\_TEXT»"  
 ]  
 }

Так как каждый раз вызывать сниппет довольно долго, вместо этого привяжем его к горячей клавише. Для этого:

1. Открыть редактор горячих клавиш (File > Preferences > Keyboard Shortcuts).
2. В правом верхнем углу нажмите кнопку *Open Keyboard Shortcuts (JSON)*. Откроется файл с вашими горячими клавишами (Рисунок 17).

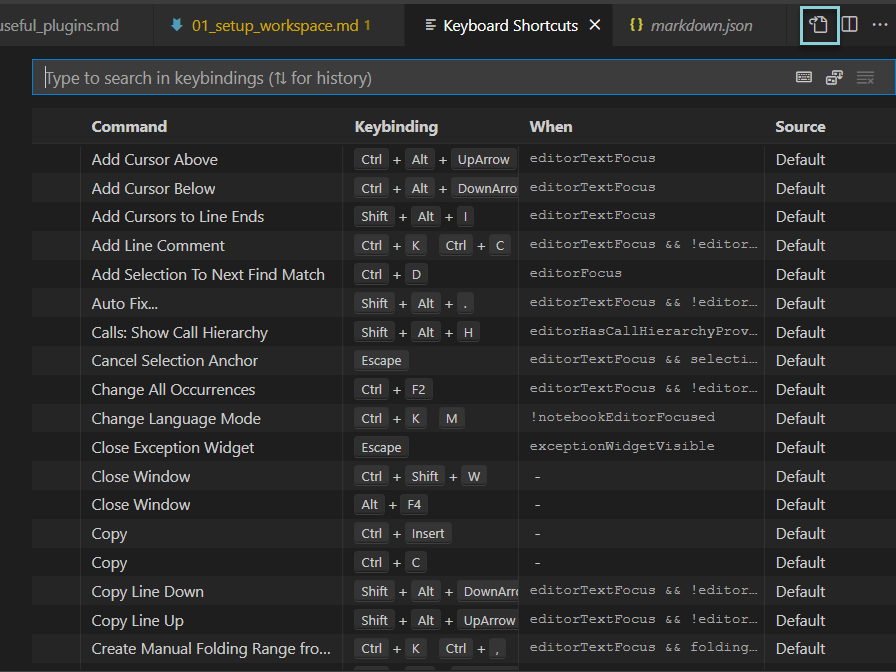


Рисунок  – Открыть редактор горячих клавиш

Вставить код следующего вида:

[  
 {  
 "key": "ctrl+shift+q",  
 "command": "editor.action.insertSnippet",  
 "when": "editorTextFocus",  
 "args": {  
 "langId": "markdown",  
 "name": "French Quotes"  
 }  
 }  
]

Теперь в тексте нужно выделить текст, который хотим заключить в кавычки, и нажать Ctrl + Shift + Q.

## Tasks

В VSCode можно настроить запуск кастомных тасков и скриптов. Много примеров и описание в официальной документации: https://code.visualstudio.com/docs/editor/tasks.

## Запускаем сборку документации

Кастомные таски сохраняются в директорию рабочего пространства в ./.vscode/tasks.json. Такой файл уже есть в демонстрационном репозитории. Вы можете создать его вручную, или командой из терминала VSCode (Ctrl+Shift+P> Tasks: Configure Tasks > Create tasks.json file from template).

В массив tasks добавить вот такую задачу (дефолтную можете удалить):

{  
 "label": "build report",  
 "type": "shell",  
 "options": {  
 "cwd": "${workspaceFolder}\\docs\_gost"  
 },  
 "command": ".\\build-docs.bat",  
 "presentation": {  
 "echo": true,  
 "reveal": "always",  
 "focus": false,  
 "panel": "shared",  
 "showReuseMessage": false,  
 "clear": true  
 },  
 "group": {  
 "kind": "build",  
 "isDefault": false  
 }  
}

Сохраняем. Теперь задачу можно вызвать из меню Terminal > Run Tasks… > выбрать свою таску.

## Запускаем стриппер

Для стриппера в директории \scripts\_and\_macros также есть .bat файл. Сделаем задачу и для него:

{  
 "label": "strip gostdown",  
 "type": "shell",  
 "options": {  
 "cwd": "${workspaceFolder}\\scripts\_and\_macros"  
 },  
 "command": ".\\run\_stripper.bat",  
 "presentation": {  
 "echo": true,  
 "reveal": "always",  
 "focus": false,  
 "panel": "shared",  
 "showReuseMessage": false,  
 "clear": true  
 },  
 "group": {  
 "kind": "build",  
 "isDefault": false  
 }  
 }

Точно также можно вызывать и другие командные файлы и даже внешние программы. В демонстрационном проекте сконфигурировано 4 задачи. Проверьте только перед запуском, что в bat файлах указаны правильные пути к исполняемому файлу python.exe и виртуальной среде (для mkdocs).

Заключение

Это все, что я могу сказать по этому поводу. Такие дела.

Список использованных источников

1. *Голиков И.* Сверточная нейронная сеть, часть 1: Структура, топология, функции активации и обучающее множество [Электронный ресурс]. — URL: <https://habr.com/ru/post/348000/> (дата обращения: 16.02.2022).

2. *Jia Y., Shelhamer E., Donahue J., Karayev S., Long J., Girshick R., Guadarrama S., Darrell T.* Caffe: Convolutional architecture for fast feature embedding / Proceedings of the 22nd ACM international conference on multimedia. — 2014. — С. 675–678.

3. *Ren S., He K., Girshick R., Sun J.* Faster r-cnn: Towards real-time object detection with region proposal networks // Advances in neural information processing systems. — 2015. — Т. 28.

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. [↑](#footnote-ref-1)