Цели и задачи проекта

Цель

Познакомиться с принципом **Docs as Code** и изучить новые инструменты.

Задачи

1. Создать проект **Foliant** для документации на языке разметки **Markdown**.
2. Генерировать документацию в формате HTML.
3. Хранить исходники в **Git**.
4. Публиковать сайт с документацией на **Github**.

??? Example "Задачи на развитие" - [x] Описать генерацию PDF и DOCX. - [x] Генерировать документы по ГОСТу с помощью GOSTdown. - [x] Изменить оформление сайта. - [ ] Создать задачу в Jenkins для автоматической публикации документации при Pull Request. - [ ] Проверять тексты с помощью линтера. - [ ] Описать текущие настройки MkDocs. - [ ] Найти инструмент для условий **IF PDF** - см. Flags. - [ ] Создать скрипт, который добавляет структуру проекта **Foliant** в файл **build-demo-espd.bat** GOSTdown. - [ ] Описать создание сниппетов.

Используемые инструменты

В проекте используются инструменты:

- **Foliant** – инструмент для разработки документации. Позволяет создавать сайты и документы в форматах PDF и DOCX из Markdown-файлов;
- **Python** – язык программирования, на котором разработан Foliant;
- **Pandoc** – инструмент для конвертации файлов. Используется для создания документов в форматах PDF и DOCX;
- **MiKTeX** – открытый дистрибутив TeX для Windows. Используется для астройки и верстки PDF-документов;
- **Mkdocs** – генератор статических сайтов;
- **MdToPdf** – альтернативная библиотека для создания PDF-документов;
- **GOSTdown** – набор шаблонов и скриптов для автоматической вёрстки документов по ГОСТ 19.xxx (ЕСПД) и ГОСТ 7.32 (отчёт о научно-исследовательской работе) в форматах docx из файлов текстовой разметки Markdown.

Процесс разработки

Процесс разработки текстов в проект состоит из следующих этапов:

1. Подготовка к работе.
2. Разработка текстов.
3. Работа с Git.
4. Настройка шаблонов сборки, сборка и публикация документации.

Полезные ссылки

- Работа с Git через консоль
- markdownlint demo
- GOSTdown

Подготовка к работе

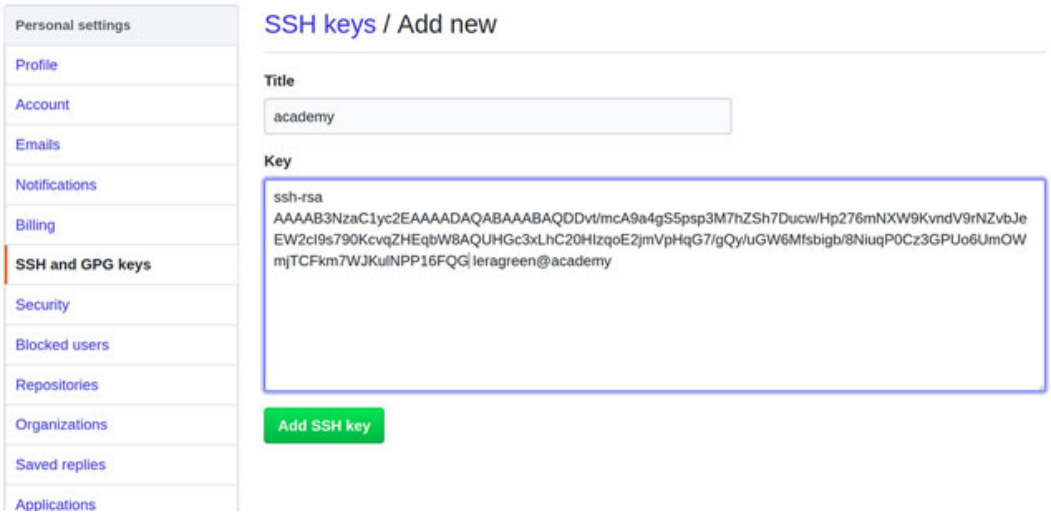
[TOC]

Установка Git

1. Скачайте файл инсталлятора с сайта Git и установите Git.
2. Откройте терминал и введите команду `git --version`. Появится информация об установленной версии Git.

Настройка Git

1. Откройте терминал и выполните команды:
   - `git config --global user.name "профиль"`
   - `git config --global user.e-mail`.
2. Сгенерируйте SSH-ключ с помощью утилиты Putty. Подробнее см. статью "Как сгенерировать SSH-ключ для доступа на сервер".
3. Откройте профиль на Github и перейдите на страницу "SSH and GPG keys".
4. Нажмите на кнопку **New SSH key**. Откроется страница:



5. Укажите имя ключа в поле **Title**.
6. Вставьте ключ в поле **Key**.
7. Нажмите на кнопку **Add SSH key**.

Установка Foliant и необходимых компонентов

Чтобы установить **Foliant** и необходимые компоненты:

1. Установите Python.
2. Установите **Foliant** с помощью pip:

```
$ python -m pip install foliant foliantcontrib.init
```

3. Установите менеджер пакетов Chocolatey.
4. Установите **Pandoc** с помощью Chocolatey:

```
choco install pandoc
```

5. Установите **MkDocs**:

```
pip install mkdocs
```

6. Установите MiKTeX.
7. Установите nodejs.
8. Установите MdToPdf с помощью npm:

```
$ npm install -g md-to-pdf
```

Создание проекта Foliant

1. В командной строке перейдите в папку, в которой будет создан проект **Foliant**.
2. Создайте проект:

```
$ foliant init
```

3. Укажите имя проекта, например, "Hello Foliant". Появится сообщение:

```
Project "Hello Foliant" created in hello-foliant
```

Чтобы посмотреть содержимое проекта, выполните команды:

```
$ cd hello-foliant
$ tree
.
├── docker-compose.yml
├── Dockerfile
├── foliant.yml
├── README.md
├── requirements.txt
└── src
    └── index.md
1 directory, 6 files
```

Проект содержит файлы и папки:

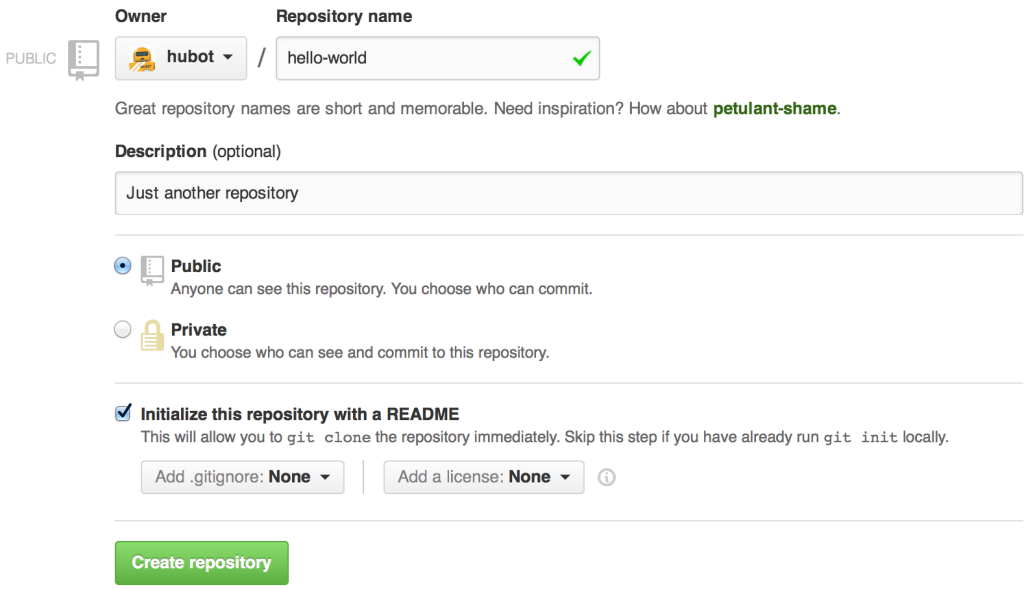
- **docker-compose.yml** и **Dockerfile** – файлы, необходимые для создания проекта в Docker;
- **foliant.yml** – конфигурационный файл проекта;
- **README.md** – файл с информацией о проекте;
- **requirements.txt** – список пакетов Python, необходимых для проекта: бэкенды и препроцессоры, темы для MkDocs и т.д.;
- **src** – папка с исходными файлами проекта. По умолчанию в папке создается файл index.xml.

Создание репозитория

Чтобы создать репозитой на https://github.com/:

1. Зарегистрируйтесь или войдите в свой аккаунт на https://github.com/.

2. Нажмите на кнопку **New** в колонке **Repositories**. Откроется окно:



3. Укажите названия репозитория в поле **Repository name**.

4. Установите переключатель **Public**.

5. Не устанавливайте флажок **Initialize this repository with a README**.

6. Нажмите на кнопку **Create repository**.

7. Нажмите на кнопку **Upload files** и загрузите файлы проекта.

Разработка текстов

Markdown

Для разработки документации используется язык разметки Markdown.

Подробнее о синстаксисе см. статьи:

- Официальный сайт
- Использование языка разметки Markdown для написания документации
- Markdown Cheatsheet
- Python-Markdown

Инструменты

Для работы с Markdown можно использовать любой текстовый редактор.

При создание этого проекта использовался редактор **Sublime Text** с плагином **MarkdownEditing**.

Добавление разделов

Чтобы добавить разделы в проект **Foliant**:

1. Перейдите в папку **src**.
2. Создайте файл с расширением *.md.
3. Откройте конфигурационный файл **foliant.yml**.
4. Добавьте имя созданного файла в список **chapters**.

Работа с Git

Создание ветки

1. Откройте терминал и выполните команду: `git branch`. Появится список веток, выделена текущая ветка.
2. Если текущая ветка – **master**, создайте ветку: `git checkout -b имя-новой-ветки`. Если текущая ветка – не **master**, переключитесь в основную ветку: `git checkout master` и создайте новую.

Отправка изменений в Github

1. Сохраните изменения всех файлов:

```
git add -A
```

2. Зафиксируйте изменения:

```
git commit -m "ваше сообщение"
```

3. Отправьте изменения в репозиторий:

```
git push origin название-текущей-ветки
```

4. Откройте репозиторий в Github.
5. Перейдите на закладку **Pull requests** и нажмите на кнопку **New pull request**.
6. Чтобы принять пуллреквест, нажмите на кнопку **Create Pull Request**.
7. Чтобы слить изменения в ветку **master**, нажмите на кнопку **Merge pull request**.
8. Нажмите на кнопку **Confirm merge**.

Актуализация локального репозитория

1. В локальном репозитории перейдите в ветку **master**:

```
git checkout master
```

2. Загрузите изменения из ветки **master** мастер-репозитория:

```
git pull my-project master
```

3. Отправьте изменения из своей ветки **master** в ваш форк на GitHub:

```
git push origin master
```

Теперь форк и оригинальный репозиторий находятся в актуальном состоянии.

Сборка и публикация документации

HTML

Выбор и настройка шаблона MkDocs

По умолчанию проект **Foliant** конвертируется в HTML с помощью шаблона **mkdocs**.

Чтобы сменить шаблон:

1. Выберите шаблон на странице MkDocs Themes.
2. Установите шаблон. Например, шаблон **Materials**:

```
pip install mkdocs-material
```

3. Откройте конфигурационный файл **foliant.yml** и добавьте строки:

```
theme:
  name: 'material'
```

Шаблон настраивается в файле **foliant.yml**. Описание параметров см. в документации для конкретного шаблона. Например, для шаблона **Materials** см. статью Getting Started.

При необходимости можно создать собственный шаблон. Подробнее см. статью Custom themes.

Локальная сборка сайта

Чтобы локально собрать сайт:

1. Выполните команду:

```
 foliant make site --with mkdocs
```

   В папке проекта создается папка "<Название проекта>.mkdocs".

2. Перейдите в папку с сайтом:

```
cd flnt-test.mkdocs
```

3. Запустите веб-сервер:

```
python -m http.server
```

4. В браузере откройте страницу: http://localhost:8000/.

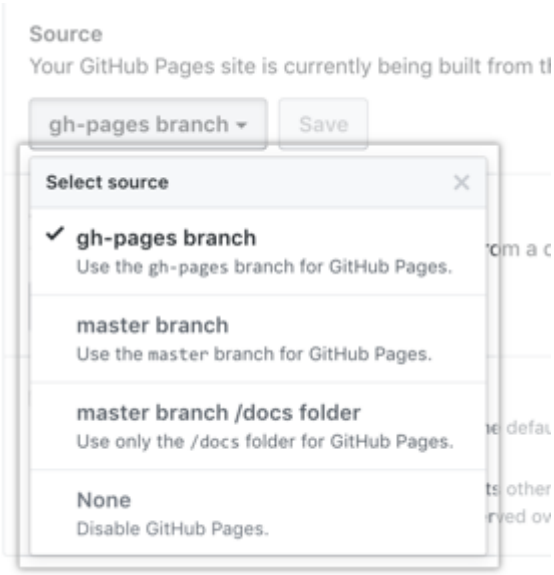
Публикация на GitHub

Чтобы опубликовать сайт на GitHub:

1. Откройте настройки репозитория и перейдите в раздел **Danger Zone**.

2. Убедитесь, что ваш репозиторий публичный. Если нет, нажмите на кнопку **Make public**.

3. Перейдите в раздел **GitHub Pages** и в выпадающем списке **Source** выберите ветку **gh-pages branch**.



4. Выполните команду:

```
foliant make ghp -p \my-project
```

PDF

md-to-pdf

Библиотека md-to-pdf генерирует PDF-файлы, которые можно настроить с помощью CSS и highlight.js.

Чтобы создать PDF-файл, выполните команду:

```
foliant make pdf --with mdtopdf
```

Pandoc

Pandoc — универсальная утилита для работы с текстовыми форматами.

Чтобы создать PDF-файл, выполните команду:

```
foliant make pdf -p \my-project --with pandoc
```

DOCX

DOCX-файлы создаются с помощью Pandoc.

Чтобы создать DOCX-файл, выполните команду:

```
foliant make docx -p my-project
```

Создание документов по ГОСТ

Для создания документов по ГОСТ предназначен набор шаблонов и скриптов **GOSTdown**.

GOSTdown создает документы по ГОСТ 19.xxx (ЕСПД) и ГОСТ 7.32 (отчёт о научно-исследовательской работе) в форматах docx из файлов текстовой разметки Markdown.

Подробнее о **GOSTdown** см. в репозитории.

Установка и настройка

1. Убедитесь, что на компьютере установлен Microsoft Word 2010 или выше.
2. Убедитесь, что на компьютере установлен Pandoc. По умолчанию **Pandoc** устанавливается в папку C:\Users\user\AppData\Local\Pandoc.
3. Откройте компонент «Система» в панели управления: Панель управления\Все элементы панели управления\Система.
4. Нажмите на ссылку **Дополнительные параметры системы** в левой панели.
5. Нажмите на кнопку **Переменные среды...**.
6. Убедитесь, что переменная **PATH** содержит путь к папке, в которой установлен **Pandoc**.
7. Скачайте фильтр **pandoc-crossref** из репозитория.
8. Распакуйте архив и поместите файл **pandoc-crossref.exe** в папку с **Pandoc**.
9. Установите шрифты компании «Паратайп»: PT Serif, PT Sans и PT Mono.
10. Убедитесь, что на компьютере установлена систем контроля версий Git.
11. Запустите **PowerShell** с правами администратора и выполните команду:

    ```
    set-executionpolicy remotesigned
    ```

12. Перейдите в папку, в которую необходимо установить **GOSTdown**, и склонируйте репозиторий:

    ```
    git clone https://gitlab.iaaras.ru/iaaras/gostdown.git
    ```

13. Запустите **build-demo-report.bat** и **build-demo-espd.bat**.

14. Убедитесь, что скрипты

Создание документов

Чтобы создать документ по ГОСТ 19.xxx (ЕСПД):

1. Скопируйте разработанные MD-файлы в папку с **GOSTdown**.

2. Отредактируйте файлы:

   - **demo-template-espd.docx** - шаблон документа, который содержит титульный лист;
   - **demo-espd-beginning.md** – первая часть документа, которая содержит аннотацию и содержание;
   - **demo-espd-end.md** – последняя часть документа, которая содержит приложения, обозначения и сокращения, список использованных источников.

3. В файле **build-demo-espd.bat**:

   i. Между **demo-espd-beginning.md** и **demo-espd-end.md** перечислите MD-файлы, которые необходимо включить в итоговый документ.

   ii. Укажите шаблон. По умолчанию используется файл **demo-template-espd.docx**.

   iii. Укажите названия итоговых DOCX- и PDF-файлов.

   iv. Чтобы внедрить шрифты в итоговые файлы файл, укажите параметр **embedfonts**.

   Пример файла:

   ```
   powershell.exe -command .\build.ps1 ^
   -md demo-espd-beginning.md,index.md,start.md,docs.md,git.md,publish.md,demo-espd-end.md ^
   -template demo-template-espd.docx ^
   -docx test.docx ^
   -pdf test.pdf ^
   -embedfonts
   ```

4. Чтобы включить cdrdjpye. нумерацию рисунков и таблиц, в файле **demo-espd-beginning.md** удалите `chapters: true` из заголовка файла.

5. Запустите **PowerShell** с правами администратора и запустите **build-demo-espd.bat**:

   ```
   .\build-demo-espd.bat
   ```