

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**.

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 настройки 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](#)
- [PlantUML](#) — все, что нужно бизнес-аналитику для создания диаграмм в программной документации

Подготовка к работе

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:

Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys**
- Security
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications

SSH keys / Add new

Title

academy

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDDvt/mcA9a4gS5psp3M7hZSh7Ducw/Hp276mNXW9KvndV9rNZvbJe
EW2cl9s790KcvqZHEqbWBAQUHGc3xLhC20HlzoE2jmVpHqG7/gQy/uGW6Mfsbigb/8NiuqP0Cz3GPUo6UmOW
mjTCFkm7WJKuINPP16FQGJleragreen@academy
```

Add SSH key

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 репозиторий 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:

Owner **Repository name**

PUBLIC   **hubot** / **hello-world** ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another repository

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

☒ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None** ⓘ

Create repository

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.

Настройка инструментов

В разделе описаны текущие настройки инструментов.

MkDocs

Настройки **MkDocs** хранятся в файле `foliant.yml`:

```
mkdocs:
  mkdocs_path: mkdocs
  slug: flnt-test
  use_title: true
  use_chapters: true
  use_headings: true
  default_subsection_title: Expand
  mkdocs.yml:
    repo_name: fish-train/flnt-test
    repo_url: https://github.com/fish-train/flnt-test
    edit_uri: edit/master/src/
    site_name: Docs as Code. Учебный проект
    theme:
      name: 'material'
      logo:
        icon: 'keyboard'
      favicon: img/keyboard.png
      language: 'ru'
      palette:
        primary: 'indigo'
        accent: 'indigo'
    extra:
      search:
        language: 'en, ru'
    markdown_extensions:
      - toc:
          toc_depth: '2-6'
          permalink: true
      - admonition
      - codehilite
      - pymdownx.tasklist:
          custom_checkbox: true
      - pymdownx.details
```

mkdocs_path. Путь к файлу `mkdocs.exe`. По умолчанию путь к исполняемому файлу содержится в переменной окружения `PATH`.

slug. Название папки без наименования проекта, в которую локально выгружается сайт. Например, если параметр имеет значение «`mkdocs`», то имя папки: <название проекта>.`mkdocs`.

use_title. Если параметр имеет значение `true`, то для заголовка сайта используется значение параметра **site_name**. Если `false`, укажите заголовок сайта вручную, иначе MkDocs не сможет создать сайт. Значение по умолчанию `true`.

use_chapters. Если параметр имеет значение `true`, то значения **chapters** из `foliant.yml` используется как значения **pages** в `mkdocs.yml`.

use_headings.

default_subsection_title.

mkdocs.yml. Параметры из [mkdocs.yml](#):

- `repo_name`. Логин пользователя и название репозитория в Git. Например, `fish-train/flnt-test`.
- `repo_url`.
- `edit_uri`.
- `site_name`.
- `theme`.
 - `name`.
 - `logo`.
 - `icon`.
 - `favicon`.
 - `language`.
 - `palette`.
 - `primary`.
 - `accent`.
- `extra`.
 - `search`.
 - `language`.
- `markdown_extensions`.
 - `toc`.
 - `toc_depth`.
 - `permalink`.
 - `admonition`.
 - `codehilite`.
 - `pymdownx.tasklist`.
 - `custom_checkbox`.
 - `pymdownx.details`.

Разработка текстов

Markdown

Для разработки документации используется язык разметки Markdown.

Подробнее о синтаксисе см. статьи:

- [Официальный сайт](#)
- [Использование языка разметки Markdown для написания документации](#)
- [Markdown Cheatsheet](#)
- [Python-Markdown](#)

Инструменты

editor_id1 Для работы с Markdown можно использовать любой текстовый редактор. editor_id2

При создании 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:

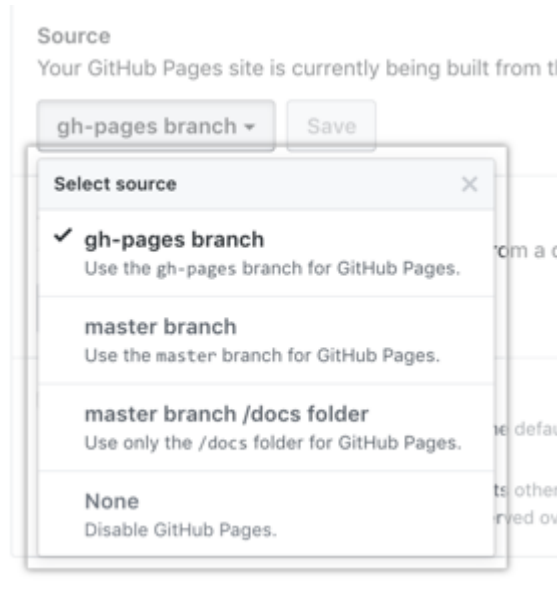
```
python -m http.server
```

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-espdocx** - шаблон документа, который содержит титульный лист;
 - **demo-espdocbeginning.md** – первая часть документа, которая содержит аннотацию и содержание;
 - **demo-espdocend.md** – последняя часть документа, которая содержит приложения, обозначения и сокращения, список использованных источников.
3. В файле **build-demo-espdocbat**:
 - i. Между **demo-espdocbeginning.md** и **demo-espdocend.md** перечислите MD-файлы, которые необходимо включить в итоговый документ.
 - ii. Укажите шаблон. По умолчанию используется файл **demo-template-espdocx**.
 - 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-espdocbeginning.md,index.md,start.md,docs.md,git.md,publish.md,demo-espdocend.md ^  
-template demo-template-espdocx ^  
-docx test.docx ^  
-pdf test.pdf ^  
-embedfonts
```

4. Чтобы включить cdrdjpye. нумерацию рисунков и таблиц, в файле **demo-espdocbeginning.md** удалите `chapters: true` из заголовка файла.
5. Запустите **PowerShell** с правами администратора и запустите **build-demo-espdocbat**:

```
.\build-demo-espdocbat
```

Интеграция Swagger с документацией

Чтобы интегрировать Swagger со своим проектом:

1. Создайте описание API в [Swagger Editor](#) и сохраните YAML-файл. Например, под названием `орепарі_3.0.2.yaml`.
2. Перейдите в [репозиторий Swagger UI](#) и нажмите на кнопку **Clone or download**.
3. Нажмите на кнопку **Download ZIP**.

4. Распакуйте архив и перейдите в папку `dist`.
5. Из папки `dist` скопируйте файл `swagger-ui.css` и вставьте его в папку `css` проекта.
6. Из папки `dist` скопируйте файлы `swagger-ui-bundle.js` и `swagger-ui-standalone-preset.js` и вставьте их в папку `js` проекта.
7. В файл `foliant.yml` добавьте настройки `mkdocs`:

```
extra_css:
  - css/swagger-ui.css
extra_javascript:
  - js/swagger-ui-bundle.js
  - js/swagger-ui-standalone-preset.js
```

8. Скопируйте YAML-файл с описанием API в папку проекта.
9. Создайте MD-файл и вставьте в него код:

```
<link rel="stylesheet" type="text/css" href="css/swagger-ui.css" >
<style>
  html
  {
    box-sizing: border-box;
    overflow: -moz-scrollbars-vertical;
    overflow-y: scroll;
  }
  *,
  *:before,
  *:after
  {
    box-sizing: inherit;
  }
  body {
    margin:0;
    background: #fafafa;
  }
</style>

<div id="swagger-ui"></div>

<script src="js/swagger-ui-bundle.js"> </script>
<script src="js/swagger-ui-standalone-preset.js"> </script>
<script>
window.onload = function() {
  // Build a system
  const ui = SwaggerUIBundle({
    url: "https://petstore.swagger.io/v2/swagger.json",
    //url: "../src/openapi_3.0.2.yaml",
    //url: "/openapi_3.0.2.yaml",
    dom_id: '#swagger-ui',
    defaultModelsExpandDepth: -1,
    docExpansion: "list",
    deepLinking: true,
    presets: [
      SwaggerUIBundle.presets.apis,
      SwaggerUIStandalonePreset
    ],
    plugins: [
      SwaggerUIBundle.plugins.DownloadUrl
    ],
    layout: "StandaloneLayout"
  })
}
```

```

    })
    window.ui = ui
  }
</script>

<style>
.swagger-ui .info .title small pre {
  padding: 1px;
  background-color: #444;
}
.swagger-ui .info .title small {
  font-size: 10px;
  position: relative;
  top: -5px;
  display: inline-block;
  margin: 0 0 0 5px;
  padding: 4px;
  vertical-align: super;
  border-radius: 57px !important;
  background: #89bf04 !important;
}
.swagger-ui .info .title small pre.version {
  background-color: #89bf04;
  border: 0px;
}
.swagger-ui pre.version {
  padding: 0px;
  max-width: 60px;
  border: 0px;
}
.swagger-ui .info .title small pre {
  padding: 0px;
}
.swagger-ui .info .title small {
  background-color: rgb(137, 191, 4);
}
.swagger-ui table th, .swagger-ui table td {
  padding: 10px !important;
}
.swagger-ui table th {
  color: white;
  font-size: 16px;
}
.swagger-ui .col_header {
  color: black !important;
}
div#swagger-ui {
  border: 1px solid #dedede;
}
.swagger-ui .info .title small pre {
  padding: 1px;
  background-color: #444;
}
.swagger-ui .info .title small {
  font-size: 10px;
  position: relative;
  top: -5px;
  display: inline-block;
  margin: 0 0 0 5px;
  padding: 4px;
  vertical-align: super;
  border-radius: 57px !important;
  background: #89bf04 !important;
}
.swagger-ui .info .title small pre.version {
  background-color: #89bf04;
}
.swagger-ui li.tabitem {
  list-style: none !important;
}

```

```

}
.swagger-ui .response-col_description__inner p {
  color: white;
  font-style: normal;
  font-size: 12px;
}
.swagger-ui pre.version {
  padding: 0px;
}
.swagger-ui .info .title small pre {
  padding: 0px;
}
.swagger-ui .info .title small {
  background-color: rgb(137, 191, 4);
}
.swagger-ui a.tablinks {
  margin-right: 20px;
}
.swagger-ui td.col.response-col_status {
  padding: 10px !important;
}
.swagger-ui .opblock .opblock-section-header h4 {
  font-size: 18px !important;
  font-weight: bold;
  padding: 0px;
}
.swagger-ui td.col, .swagger-ui td.col.col_header.response-col_description {
  padding: 10px;
}
.swagger-ui h4.opblock-title_normal {
  font-size: 16px;
  font-style: italic;
}
.swagger-ui h4.opblock-title_normal[id] {
  padding-bottom: 15px;
  font-style: italic;
}
.swagger-ui {
  border: 1px solid #dedede;
}
.swagger-ui select {
  font-weight: normal !important;
  font-family: monospace;
}
.swagger-ui table {
  table-layout: auto !important;
}
.swagger-ui .scheme-container {
  padding: 0px 0px 15px 0px;
}
.swagger-ui .renderedMarkdown p {
  font-size: 14px;
}
.swagger-ui tr.response p {
  font-style: italic;
}
.swagger-ui table.model tbody tr td {
  padding: 1em !important;
}
.response-content-type.controls-accept-header small code {
  font-size: 12px;
}
.swagger-ui .opblock-summary-path a.nostyle {
  font-family: monospace;
}
.swagger-ui .info {
  /* margin: -25px 0px !important; */
}
.swagger-ui .main span.url {

```

```

    display: none;
}
.swagger-ui span.opblock-summary-path a.nostyle {
    font-family: Monospace !important;
    size: 16px;
}
.swagger-ui .opblock-description-wrapper, .swagger-ui .opblock-external-docs-wrapper, .swagger-ui
.opblock-title_normal {
    padding: 15px 20px 5px 20px;
}
.swagger-ui h1[id], .swagger-ui h2[id], .swagger-ui h3[id], .swagger-ui h4[id], .swagger-ui h5[id] {
    margin: 0px;
    padding: 0px;
}
.swagger-ui pre {
    font-family: Monaco, Monospace !important;
    font-size: 11px;
}
h6, h6 code.highlighter-rouge {
    font-size: 16px;
}
.swagger-ui .responses-inner h4, .swagger-ui .responses-inner h5 {
    font-size: 16px;
}
.swagger-ui code {
    font-size: 12px;
}
/* disable the try it out buttons
button.btn.try-out__btn {
    display: none;
}
*/

.topbar {
    display: none;
}
</style>

```

Пример описания API

Swagger Petstore

1.0.3

[Base URL: petstore.swagger.io/v2]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on irc.freenode.net, [#swagger](https://t.me/swagger). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)

[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

Schemes

HTTPS



Authorize



pet

Everything about your Pets

Find out more: <http://swagger.io>



GET

/pet/{petId} Find pet by ID



POST

/pet/{petId} Updates a pet in the store with form data



DELETE

/pet/{petId} Deletes a pet



POST

/pet/{petId}/uploadImage uploads an image



POST

/pet Add a new pet to the store



PUT

/pet Update an existing pet



GET

/pet/findByStatus Finds Pets by status



GET

/pet/findByTag Finds Pets by tags



store

Access to Petstore orders



GET

/store/inventory Returns pet inventories by status



GET

/store/order/{orderId} Find purchase order by ID

DELETE

/store/order/{orderId} Delete purchase order by ID

POST`/store/order` Place an order for a pet**user**Operations
about userFind out more about our store: <http://swagger.io>**GET**`/user/{username}` Get user by user name**PUT**`/user/{username}` Updated user**DELETE**`/user/{username}` Delete user**GET**`/user/login` Logs user into the system**GET**`/user/logout` Logs out current logged in user session**POST**`/user` Create user**POST**`/user/createWithArray` Creates list of users with given input array**POST**`/user/createWithList` Creates list of users with given input array

VALID



UML-диаграммы в PlantUML

Установка PlantUML

1. Установите [Java](#).
2. Установите [Graphviz](#).
3. Создайте переменную среды `GRAPHVIZ_DOT` и укажите в ее значении путь до файла dot.exe. Например, `C:\Program Files (x86)\Graphviz2.38\bin\dot.exe`.
4. Скачайте файл [plantuml.jar](#).
5. В корне проекта создайте файл `plantuml.cfg` и добавьте настройки отображения диаграмм. Подробнее см. в статье [Skinparam command](#).
6. Установите препроцессор для Foliant: `pip install foliantcontrib.plantuml`
7. В файле `foliant.yml` добавьте `plantuml` в раздел препроцессоров.
8. Добавьте параметр `plantuml_path` и укажите путь к файлу `plantuml.jar`. Например, `D:\plantuml\plantuml.jar`.

9. Добавьте параметры `params` и `config` .

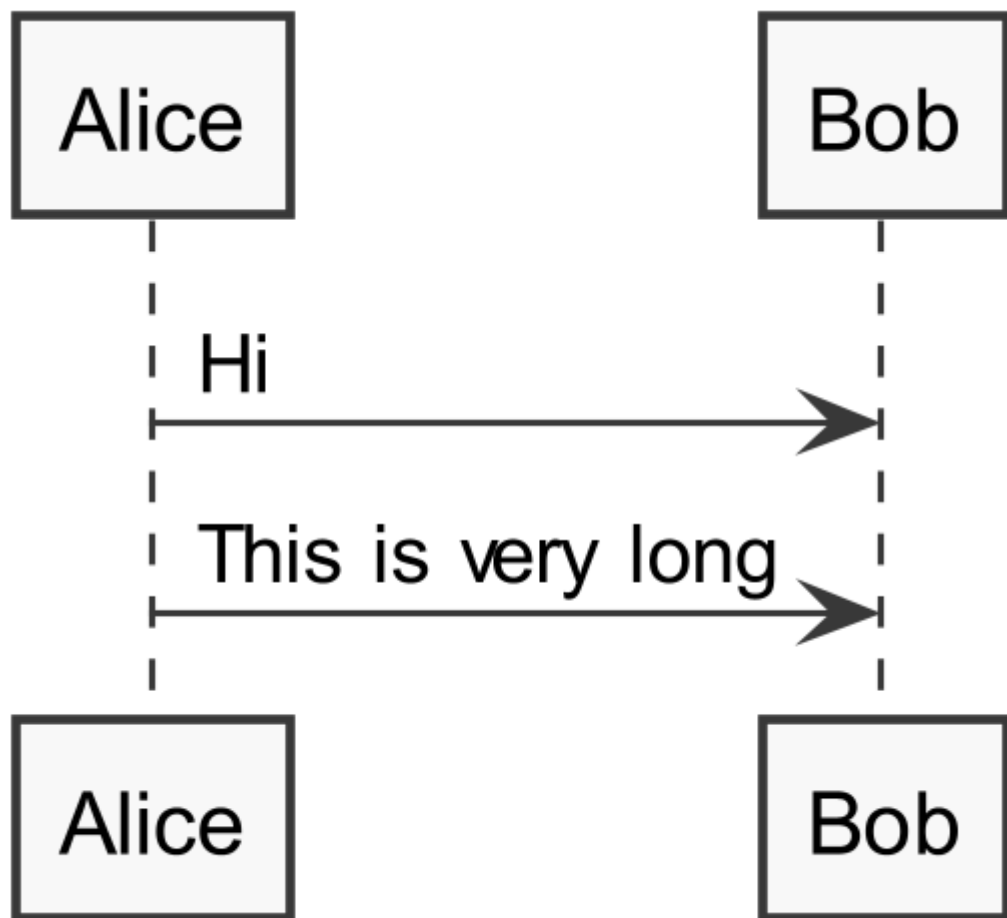
10. В параметре `config` укажите путь до файла `plantuml.cfg`: `!path plantuml.cfg`.

Подробнее об использовании PlantUML в Foliant см. статью [Plantuml](#).

Настройки отображения диаграмм см. в

Примеры диаграмм

Диаграмма последовательности



Тестирование сниппетов

1. Добавить информацию из файла целиком. Например, `git.md`.
2. Добавить кусочек текста между заголовками.
3. Добавить отдельный абзац.

Файл целиком

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.

Текст между заголовками

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**.

Абзац

Для работы с Markdown можно использовать любой текстовый редактор.