Html結果はbuild\html\index.htmlにあるので、それを読んでください。このドキュメントはhtmlの作り方を示すものです。

以下のような手順で、Sphinx ドキュメント用のフォルダを分けつつ HTML をビルドする構成が分かりやすいかと思います。  
なお、以下では「docs」ディレクトリを作成し、Sphinx 関連ファイル(設定やビルド出力)をそこにまとめて格納する例を紹介します。

※sphinxコマンドなどを実行する際には、poetry run を頭につけてください。

**1. ディレクトリ構成イメージ**

最終的に次のような構成を目指します（主要部分のみ記載）。「docs」フォルダに Sphinx の設定ファイルや生成された HTML などをまとめて管理します。

```text

D:.

├─docs ← Sphinx関連の設定ファイルやHTMLビルドを管理するフォルダ

│ ├─\_build ← HTML等のビルド出力先(自動生成)

│ ├─\_static ← 静的ファイル(CSS/画像など)を置くフォルダ(必要に応じて)

│ ├─\_templates ← テンプレート用のフォルダ(必要に応じて)

│ ├─make.bat ← Windows用のビルドスクリプト(自動生成)

│ ├─Makefile ← Linux/Unix用のビルドスクリプト(自動生成)

│ └─source ← rstファイルやconf.pyなどのSphinxソース

│ ├─conf.py ← Sphinx設定ファイル(自動生成 + 追加修正)

│ └─index.rst ← トップページのrst(自動生成 + 追加修正)

├─ragpon ← 実際のプロジェクト(ソースコード)フォルダ

│ ├─application

│ ├─config

│ ├─domain

│ ├─examples

│ ├─ml\_models

│ ├─repository

│ ├─service

│ ├─\_utils

│ └─\_\_pycache\_\_

├─tests

└─dist

```

上記のように、プロジェクトのトップディレクトリ直下に「docs」というフォルダを作り、そこに Sphinx 周りのファイルをまとめます。

**2. Sphinx のインストール**

まだの場合は、以下のように pip でインストールします。

```bash

pip install sphinx

```

また、ドキュメントを自動生成（API ドキュメントの雛形）するための sphinx-apidoc も同時にインストールされます。

**3. 「docs」フォルダの作成と初期化**

プロジェクト直下へ移動して「docs」ディレクトリを作成し、そこに移動してから Sphinx の quickstart を実行します。

```bash

cd C:\~~ # プロジェクト直下へ移動

mkdir docs

cd docs

poetry run sphinx-quickstart

```

対話形式でいくつか質問をされるので、好みの設定を入力してください。最低限下記のような項目を確認します：

* **Separate source and build directories?**: (yes/no)  
  → 「yes」にすると、source と build ディレクトリが分割されます。今回の例では「yes」を推奨します。
* **Project name**: プロジェクト名
* **Author name**: 作成者名
* **Project release**: バージョン  
  →pyproject.toml記載のバージョンと合わせるのが良さそう？です。
* その他は基本的にデフォルトでもOKです。

完了すると、「source」「\_build」などを含む一連のファイルが docs ディレクトリに生成されます。

**4. conf.py の修正**

**4.1 パスの指定**

Sphinx がプロジェクト内の Python パッケージ(例: ragpon)を正しく参照できるように、sys.path にプロジェクトソースの絶対パスを追加します。  
docs/source/conf.py をエディタで開き、以下のような箇所を修正してください。

```python

import os

import sys

# 「../」(docs フォルダから見て一つ上)をパスに追加

sys.path.insert(0, os.path.abspath('..'))

# もしさらに階層を上げたい場合は

# sys.path.insert(0, os.path.abspath('../..'))

```

今回の例では、プロジェクト直下に ragpon ディレクトリがあるため、  
docs → D:. → ragpon の参照ができるように docs から見て一つ上(..)をパスに入れています。  
これで import ragpon ができるようになります。

**4.2 拡張機能の追加 (必要なら)**

もし Google スタイルや NumPy スタイルの docstring を使用している場合など、拡張が必要なら conf.py の extensions に必要なモジュールを追加してください（例：sphinx.ext.napoleon など）。

```python

extensions = [

'sphinx.ext.autodoc',

'sphinx.ext.napoleon', # Google/Numpyスタイルのdocstringをパースしたい場合

'sphinx.ext.viewcode', # ソースコードのリンクを表示したい場合

# ... など

]

# ここでhtml\_theme = "sphinx\_rtd\_theme"とすると、htmlの見た目が良い感じになるので、併せて変更する。

```

**5. rst ファイルの自動生成 (sphinx-apidoc)**

Sphinx には、プロジェクト内の Python パッケージを走査して .rst ファイル(ドキュメント雛形)を一括生成する sphinx-apidoc というコマンドが付属しています。  
これを利用すると、ragpon 内にあるモジュールごとに .rst をまとめて生成できます。

以下のコマンドは docs ディレクトリ内で実行すると想定します。  
sphinx-apidoc の第一引数に出力先(例：source)、第二引数に解析対象パッケージ(例：../ragpon)を指定します。

```bash

cd C:\~~ # プロジェクト直下へ移動

cd docs

poetry run sphinx-apidoc -f -o ./source ../ragpon

```

* -f : 既存の rst ファイルを上書きするオプション
* -o ./source : 生成する rst ファイルの出力先
* ../ragpon : 解析対象のパッケージ(フォルダ)

このコマンドにより、source フォルダに ragpon.\*.rst のようなファイル群が生成されます。  
生成ファイルに追記することで自分でカスタムしたドキュメントを挿入したり、不要なモジュールは手動で削除したりすることができます。

**6. トップページなどの編集**

docs/source/index.rst や、 sphinx-apidoc によって生成された ragpon.\*.rst を必要に応じて編集します。  
index.rst に以下のように追記すれば、トップページから生成された各モジュールを参照できます。

```rst

.. toctree::

:maxdepth: 2

:caption: Contents:

ragpon.application

ragpon.config

ragpon.domain

...(適宜書く)

```

sphinx-apidoc 実行時に modules.rst というファイルができている場合は、その中にすでに TOC が書かれているので、index.rst からそれを読み込む形に変更しても良いとのことで、メンテナンスのしやすさでこちらを推奨します。

```rst

.. toctree::

:maxdepth: 2

:caption: Contents:

modules

```

**7. ドキュメントのビルド**

最後にドキュメントをHTMLにビルドします。  
docs フォルダ内にある make.bat(Windows) または Makefile(Linux/Unix) を使ってビルドします。

**Windows の場合:**

```bash

cd C:\~~ # プロジェクト直下へ移動

poetry run make.bat html

```

**Linux/Unix/macOS の場合:**

```bash

cd C:\~~ # プロジェクト直下へ移動

poetry run make html

```

ビルド完了後、docs/\_build/html ディレクトリが作られ、その中に HTML ファイル一式が生成されます。  
\_build/html/index.html をブラウザで開くと、生成されたドキュメントを確認できます。

**8. 運用ヒント**

1. **不要なフォルダの除外**  
   不要な \_\_pycache\_\_ や tests ディレクトリなどをドキュメントに含めたくない場合は、sphinx-apidoc 実行時に -e や --exclude オプションを付与して除外することを検討してください。  
   例：poetry run sphinx-apidoc -f -o ./source ../ragpon --exclude ../ragpon/tests\*
2. **ドキュメンテーション指針**
   * docstring(コメント) しっかり書いておく。
   * 大きな機能ごとに rst を分割し、index.rst から目次を組み立てる。
   * autodoc 等の拡張機能を活用して、常にコードとドキュメントを同期して保つ。

以上の流れで、「docs」フォルダにまとめる形で Sphinx ドキュメントを管理しつつ HTML をビルドできます。お試しください。