

SCAM Manual

jul

Version : 0.9.0



Contents

Synopsis	3
Installation	4
Walkthrough: writing the aide	6
Creating your first post	6
Attaching a content to an entry.	8
Accessing the text	8
Developping your first comment and setting your book title	9
Visualizing your document	11
HTML output	11
PDF	12
Rinse and repeat	13
Playing with the help example	14
Future roadmap	15
Less features in the frontend more reliable	15
Testing without a QA Plan, but making sure to test.	15
Getting back to good practice of software	15
Changelog	17
The chaos monkey and how scam nearly ended	18
By the way, what is my problem ?	19
Am I gonna lose my marble ?	20
If we use docker: it's debian's fault	21
Lesson learnt docker is the way to deliver	22
Dev corner	23
« Design »	23
Synchronous	25
Model	25
HTML As A Model (HaaM)	26
Creativity boosters (limitations)	26

Serendipity	27
Vanilla Markdown export with assets (gruik inside)	27
Self embedded HTML	28
Psycodelic experience	29
Liste des liens	30
WTFPL 2.0 Do any thing you want with this book except claim you wrote it.	

Synopsis

To look smart you either say it in latin or write in LaTeX and add the Naviers Stocke equation

$$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot \vec{j} = 0$$

to look smarter

But most of us ain't smart enough to use LaTeX, at most we can use markdown an easy to learn text renderer.

So here is my solution to be a professional scamer : this is a front end to a pandoc toolchain based on mind mapping for structuring the thoughts with a real time rendering of the markdown and quite a few tricks.

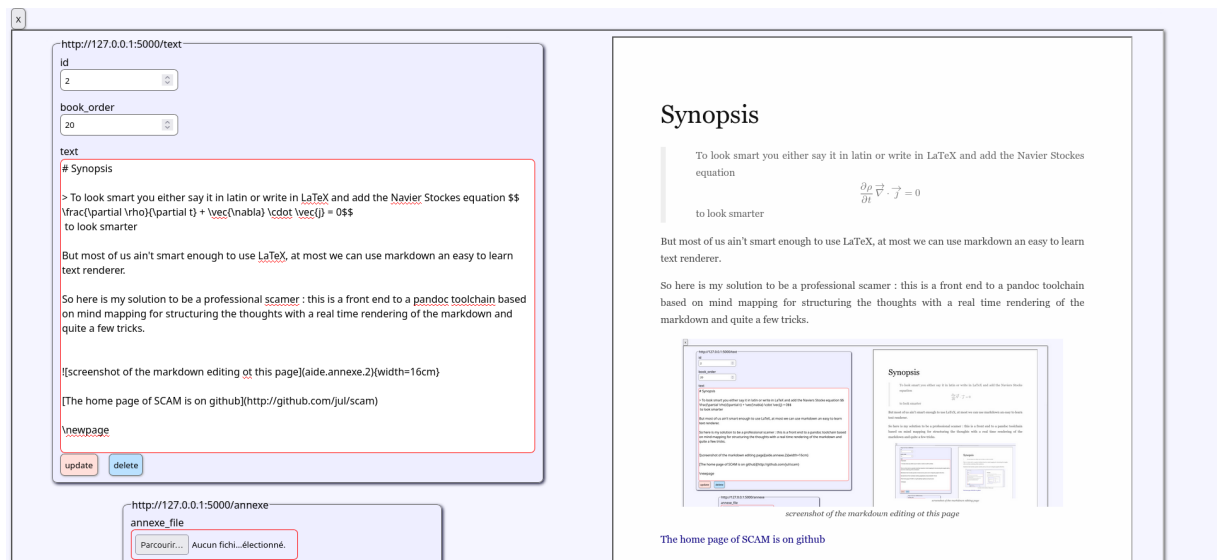


Figure 1: screenshot of the markdown editing of this page

The home page of SCAM is on github

Installation

I am too lazy to write an install script, make a debian package, or a full pip requirements solution because I need binary installs.

As a result I resorted to the solution of the lazy man which source tells you all you need : making it a docker file.

```
FROM debian:sid
ARG DB
ENV DB $DB
ENV LANG C.UTF-8
RUN mkdir -p /scam/assets
RUN mkdir -p /usr/share/man/man1 && mkdir -p /usr/share/man/man7
RUN echo 'APT::Cache-Start 100000000;' > /etc/apt/apt.conf.d/00podman
RUN apt-get update && apt-get -y dist-upgrade \
    && rm -rf /var/lib/apt/lists/*
RUN apt-get update && apt-get -y --no-install-recommends install \
    python3 python3-pip python3-venv python3-setuptools \
    python3-sqlalchemy texlive pandoc graphviz virtualenv \
    python3-magic sqlite3 texlive-xetex texlive-latex-extra \
    texlive-fonts-recommended texlive-lang-french graphviz lmodern

RUN sed -i 's/^Components: main$/& contrib/' \
    /etc/apt/sources.list.d/debian.sources
RUN apt-get update
RUN apt-get install -y ttf-mscorefonts-installer fontconfig
RUN fc-cache -f -v

RUN useradd scam -d /scam --uid 1000 -m -s /bin/bash
RUN mkdir /venv && chown scam:scam /venv
USER scam
RUN virtualenv --system-site-packages /venv
RUN . /venv/bin/activate
WORKDIR /scam
COPY --chown=scam . /scam
ENV PYTHONPATH=/venv/bin
RUN /venv/bin/python -m pip install --no-cache-dir \
    --disable-pip-version-check -r /scam/requirements.full.txt
EXPOSE 5000
CMD . /venv/bin/activate && cd /scam \
```

```
&& DB=${DB:-scam} /venv/bin/python /scam/scam.py
```

with the following requirements :

```
archery
dateutils
multipart
filelock
Mako
pandocfilters
pandoc-include
panflute
passlib
python-dateutil
SQLAlchemy>=2
SQLAlchemy-Utils
time-uuid
```

to use it I recommend the side car technique¹ wich can be used this way so that you can access the assets dir which contain the book :

```
docker build -t scam .
docker run -i -t -e DB=db/aide --mount type=bind,src=.,dst=/scam/db \
    --mount type=bind,src=./assets,dst=/scam/assets \
    -p5000:5000 scam
firefox http://127.0.0.1:5000
```

¹I think I forgot to check if images were present before generating the PDF assuming people would check the HTML first. Stupid me.

Walkthrough: writing the aide

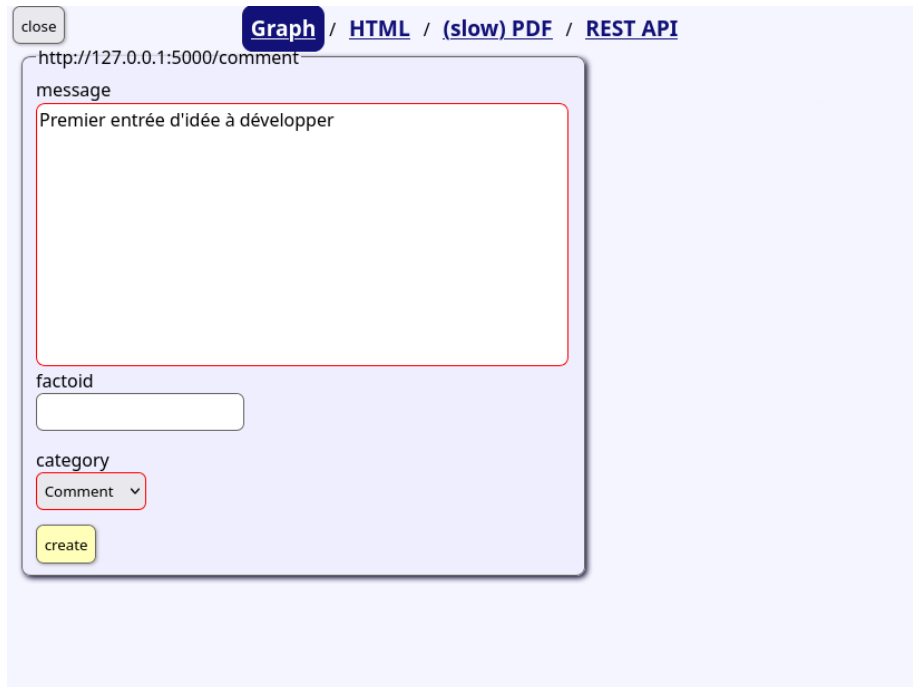
Creating your first post

The landing page give you one option : POST :D



Figure 2: Graph interface : time to create your first post

On click you should see this and be able to fill the value:



The screenshot shows a web interface with a light blue background. At the top left is a 'close' button. To its right are navigation links: 'Graph' (highlighted in dark blue), 'HTML', '(slow) PDF', and 'REST API'. Below these is a URL bar showing 'http://127.0.0.1:5000/comment'. The main form area is a light blue box with a dark blue border. It contains a 'message' label followed by a large text input field with the placeholder text 'Premier entrée d'idée à développer'. Below the message field is a 'factoid' label followed by a small text input field. Underneath is a 'category' label followed by a dropdown menu showing 'Comment' with a downward arrow. At the bottom left of the form is a yellow 'create' button.

Figure 3: First post

and then click *create*

Attaching a content to an entry.

I chose the policy that one micro item is related to one and only one attachment of the embedable kind you want.

The **annexe** widget below is used to load and delete annexes (attached files) that will be stored in the database.

These items will be available as pictures in the markdown editor by the given name on the top.



Figure 4: by clicking on the « attach » button you can attach a content to the post it entry

Accessing the text

To access the post entry you click on the node/post and it will open a modal window in which the **text** button is accessible for editing your book text

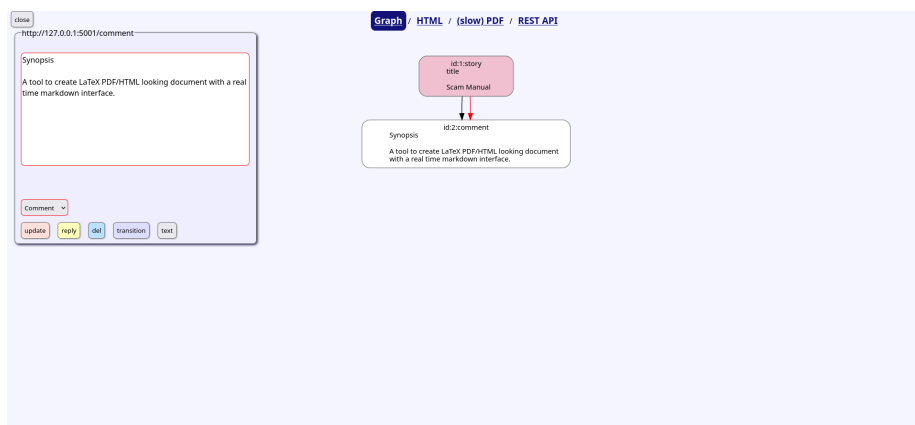


Figure 5: Once you click on a node in the graph the post editing interface appear

Developping your first comment and setting your book title

First comment is specific in the sense it is also used for the title. With pandoc you can add metadata used for LaTeX.

The *markdown* extension used here is the pandoc one

Here is a typical Pandoc flavored Markdown entry to setup the LaTeX settings here with the french settings :

```
% TITLE
% AUTHOR
% DATE \
  \
  \ ![] (aide.annexe.1){width=15cm}

---
mainfont: Georgia
header-includes:
- \usepackage[french]{babel}
- \usepackage{hyperref}
- \definecolor{myblue}{rgb}{0.28, 0.24, 0.48}
- \hypersetup{colorlinks=true, allcolors=myblue}
- \let\tmp\oddsidemargin
- \let\oddsidemargin\evensidemargin
- \let\evensidemargin\tmp
- \reversemarginpar
---
```

Your real time markdown input is definitely confused but that's fine. It is a quirk :D

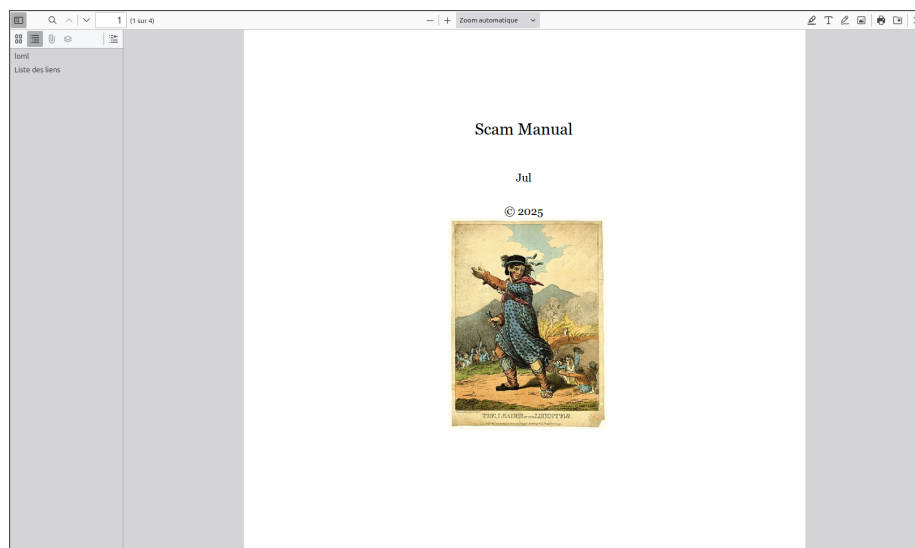


Figure 6: screenshot of the rendered PDF with the title

Visualizing your document

Now that you have rinned and repeated a few time the post entry/text process you may want to check the output.

There are 2 main output : HTML and PDF.

HTML output

To see the result

it's now time to visit the HTML rendering URL. You should have a side by side view of the generated standalone HTML and the generated PDF.

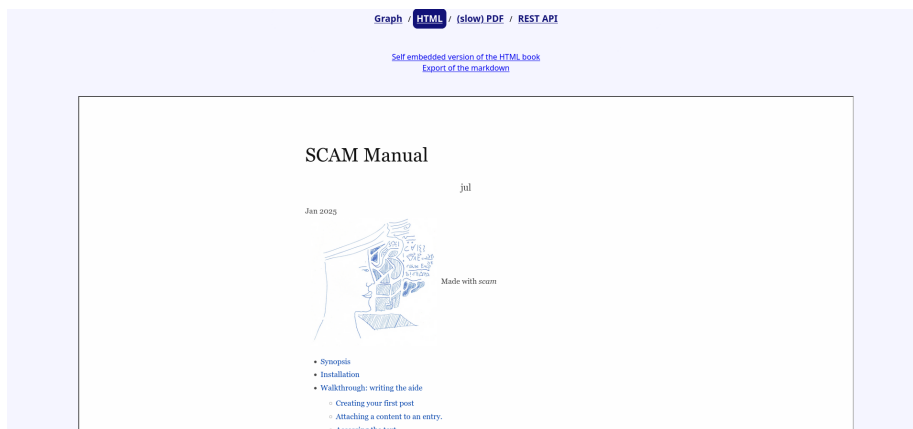


Figure 7: book rendering

You notice that there is a nice self embedded HTML link. This includes CSS and pictures inside the document for serving the document as a single file.

PDF

The PDF renderer accessible from the menu will give you the PDF rendering.

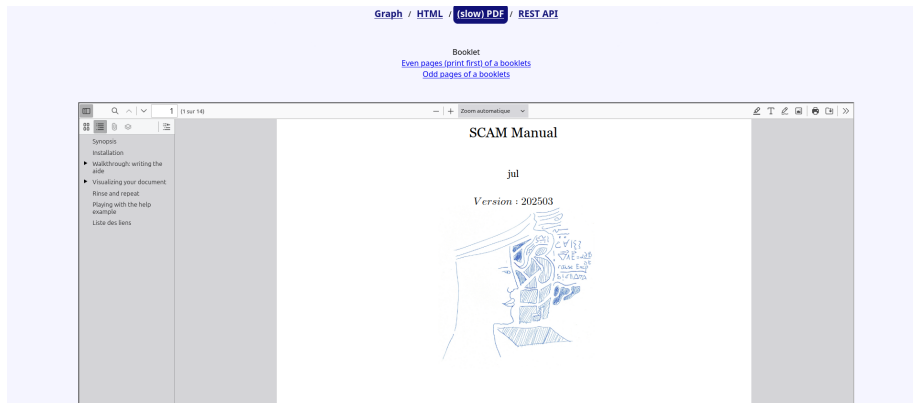


Figure 8: PDF view of the document

Rinse and repeat

After a few more entries that are boring because very repetitive if you consult the graph URL you should have now more entries.

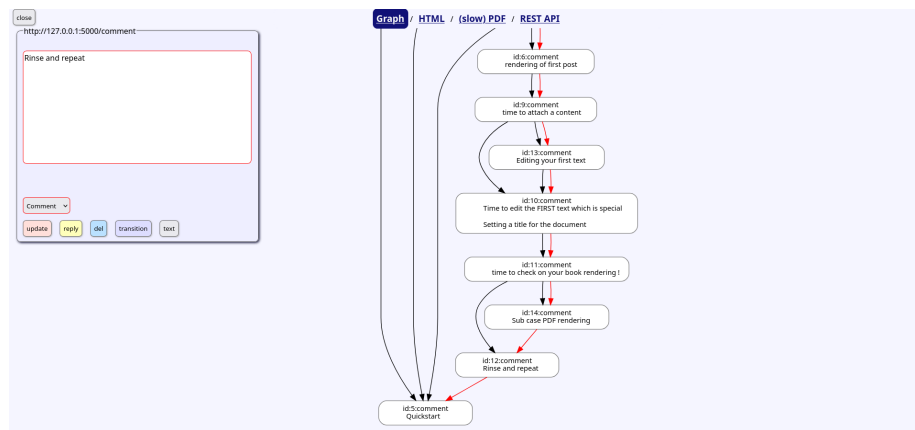


Figure 9: Your graph should now expand itself as the book

The « book order » is the red lines, they follow the ascending id order but can be overridden with the book_order rank available in the **text** view.

Playing with the help example

This book is available in the repository as a sqlite database².

To try it :

```
docker run -i -t -e DB=aide --mount type=bind,src=.,dst=/scam \  
-p5000:5000 --user 1000:1000 scam \  
firefox http://127.0.0.1:5000
```

²I am beginning to find this feature useless. Less is more, hence I'm thinking of removing it.

Future roadmap

I am at the step I have to audit this pile of code made in non standard way to improve it.

This means that in a foreseeable future I have to introduce more classical way to handle the software.

Here are a few items I need to not drown in the mess I am and reflect on what scam is : **it is a Proof of Concept of the type *façade* of a software I long for.**

Less features in the frontend more reliable

As an improvement to the *façade* I am going to focus on less features (like removing the booklet, and making the « all in one » html the default). By identifying a core of features I aim to identify the backend magic that is important to straighten up ; I feel like some mutualisation of code would be welcomed and the bash/python stuff is full of coupling that would be better cleaned up.

Testing without a QA Plan, but making sure to test.

Updating this manual using almost all the features and including a modified version of the manual with new releases will be a best effort.

Getting back to good practice of software

What I need :

- changelog ;
- version number ;
- communicating on where to open a ticket.

Starting NOW, the version of the manual is the same as the software which is 0.9.0.

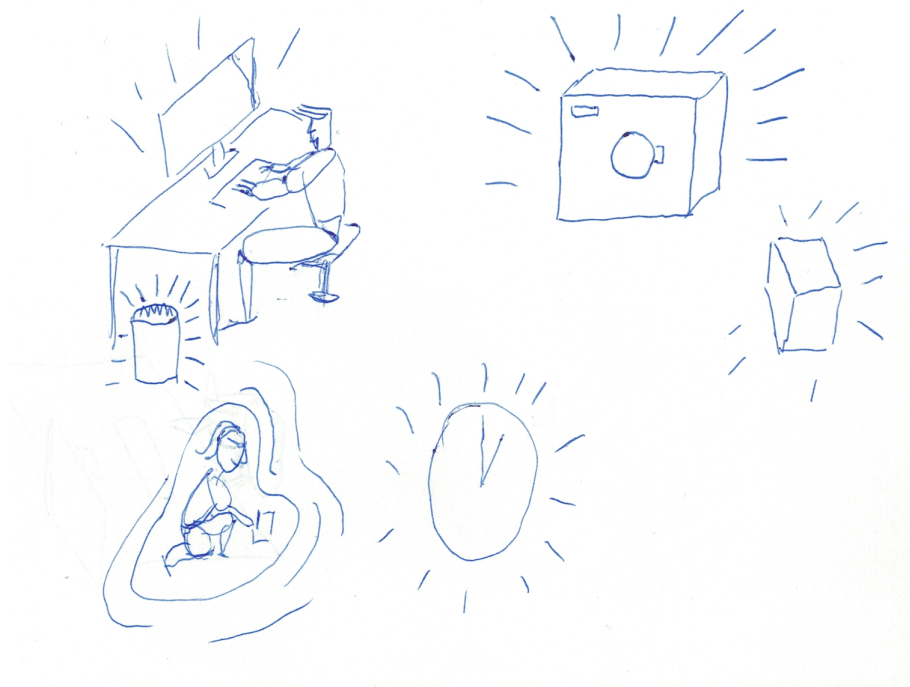


Figure 10: This includes putting an image to check the PDF chain will embed it

Changelog

0.9.0

- fixing a bug in relative addressing of database
- separating the venv python and the scam directory in Dockerfile
- improving the manuel/begin to tag the releases

nameless

All that was before before this revision

The chaos monkey and how scam nearly ended

And it's in the middle of the testing while my wife was calling for an absolute matter of life and death (probably a wet laundry) that I thought I would never be able to deliver, because after I inserted an image and checked the *pdf* output from the docker, there was only one image used.

Of course, in this kind of simple bug with a lot of moving parts is not my favorite part so I panicked.

pandoc not working in docker but working fine in my « *it works for me* » configuration must be a joke, right ?

I chose docker to not have to worry about packaging, but it comes with its extra layer of complexity. Maybe it is due to *overlayfs*?

So after reading the kernel doc about overlayfs I must admit I'm not convinced. It must be overlayfs.

So I added, some *sleep(1)*, and a lot of *sync*, and I have the same result.

Must be the extra pandoc filter I add, so ... let's remove them.

But wait! I know *mkdoc.sh* works for html and fails for pdf and there is only one line of difference

```
pandoc aide.book.pdf.int.md -so aide.book.pdf
```

Let's try to run the command in docker run -ti bash

Hum, it's failing too.

My dev machine (the most powerful of the house) is on linux mint because it is the familial computer. Maybe I should try on my debian ? Ah ! My wife is calling for putting a dish washer on.

So ... let's try *strace* : a tool to trace all system calls, maybe it is gonna see something I don't.

By the way, what is my problem ?

I have 11 pictures in the manual, but only one is used ... maybe I should look at the **open** on a working example

```
strace pandoc aide.book.pdf.int.md -so aide.book.pdf 2>&1 | grep open
```

```
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.10", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.11", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.12", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.13", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.14", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.15", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.16", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.17", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.19", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.2", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.4", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.6", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.9", O_WRONLY|O_CREAT|O_NOCTTY
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.2", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/9afdf7d3d1558598ca7e9ef19e9a2250951a0ca6.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.4", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/326c745813740c9730231dc2b72288fcb86385ad.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.6", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/9b5c1a462f1cd77731d44292be62366f0f6ef7de.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.9", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/662866d782004b094d00cce8a634920b2608bb93.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.13", O_RDONLY|O_NOCTTY|O_NONB
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/bba1e044f648f0bcd4e0679051a0fe116253c57f.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.10", O_RDONLY|O_NOCTTY|O_NONB
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/4ef73a5f4c47c6c293a1d33189de608d200716e1.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.11", O_RDONLY|O_NOCTTY|O_NONB
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/3a6745b33ed9914e71782a1255c1551ef6e3d0c2.
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/aide.annexe.14", O_RDONLY|O_NOCTTY|O_NONB
openat(AT_FDCWD, "/tmp/tex2pdf.-628bc009db183999/bafc5bc59c8ecc47ee4f1760e7a360758fc3b090.
```

Hum, it's a tad obscure, but it seems pandoc first open the file (to get the magic number I guess), and then reopen them after opening a random file or so it seems.

Let's strace inside the docker (after installing strace in the Dockerfile and waiting 5 minutes of compilation) ...

```
strace pandoc aide.book.pdf.md -so aide.book.pdf 2>&1 | grep open
```

```
....
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.1", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.png", O_WRONLY|O_CREAT|O_NOCT
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.2", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.png", O_WRONLY|O_CREAT|O_NOCT
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.4", O_RDONLY|O_NOCTTY|O_NONBL
```

```

openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.png", O_WRONLY|O_CREAT|O_NOCT
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.6", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.png", O_WRONLY|O_CREAT|O_NOCT
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.9", O_RDONLY|O_NOCTTY|O_NONBL
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.png", O_WRONLY|O_CREAT|O_NOCT
openat(AT_FDCWD, "/tmp/tex2pdf.-d57ab5e33aff4e21/aide.annexe.13", O_RDONLY|O_NOCTTY|O_NONBL

```

For the part that matters, it seems a very weirded bug of overlays, it would be better to find the code in pandoc, because something is really bugging me

After using the fact the code must be using *tmpdir* or *temp_dir* and trying bizarre stuff I find the part that handles images in the PDF and especially this part:

```

sha = showDigest (sha1 (UTF8.fromStringLazy fname))
pngOut = normalise $ tmpdir </> sha <.> "png"
pdfOut = normalise $ tmpdir </> sha <.> "pdf"


```

And that's the moment, I still don't have the good reflex (I am mad at myself), but instead go further down the rabbit hole by clicking **blame** on github.

Am I gonna lose my marble ?

And by clicking on blame I got this wonderful commit describing my old problem and its solution

Commit c1ccbc5

 Jgm committed on Jun 25, 2022

PDF: use sha1 hash of filename when converting svg.

The previous code threw away the directory component of the filename in constructing a new one. This led to surprising results if you had e.g. 'foo/pic.svg' and 'bar/pic.svg'; in the final PDF they'd be the same image, because the latter would overwrite the former in the temp directory.

1 parent [41af426](#) commit c1ccbc5

1 file changed +4 -2 lines changed

src/Text/Pandoc/PDF.hs

```

52 import Text.Pandoc.Walk (walkM)
53 import Text.Pandoc.Writers.Shared (getField, metaToContext)
54 import Control.Monad.Catch (MonadMask)
55
56 #ifdef WINDOWS
57 import Data.List (intercalate)
58 #endif
59
60 -215,8 +216,9 @@ convertImage opts tmpdir fname = do
61
62     E.catch (Right pngOut <$ JP.savePngImage pngOut img) $
63       \e :: E.SomeException -> return (Left (tshow e))
64
65     where
66       pngOut = normalise $ replaceDirectory (replaceExtension fname ".png") tmpdir
67       pdfOut = normalise $ replaceDirectory (replaceExtension fname ".pdf") tmpdir
68
69       svgIn = normalise fname
70       mime = getMimeType fname
71       doNothing = return (Right fname)
72
73     E.catch (Right pngOut <$ JP.savePngImage pngOut img) $
74       \e :: E.SomeException -> return (Left (tshow e))
75
76     where
77       sha = showDigest (sha1 (UTF8.fromStringLazy fname))
78       pngOut = normalise $ tmpdir </> sha <.> "png"
79       pdfOut = normalise $ tmpdir </> sha <.> "pdf"
80
81       svgIn = normalise fname
82       mime = getMimeType fname
83       doNothing = return (Right fname)

```

See commit here

So ... most persons used to linux know the solution already, me too, but I went full retarded on this bug. Instead of thinking of the obvious, I was like : did *debian* used a specific modification of the code?

No.

If we use docker: it's debian's fault

Here I am, looking in the wrong direction what is obvious to long time users of debian (like me, but not this day): debian stable ships years old mainstream softwares. Because software are shipped with their version of when debian was stabilized.

Hence, my debian stable ships pandoc from 3 years without the correction I need because it was frozen 3 years ago I guess.

If I had not been interrupted I would have tried at the very beginning *scam* on my debian stable and notice I had the same bug and I would have *simply checked the version* and discovered the trouble.

So now if I want to have a working version of scam for my docker stable I only need to change

```
FROM debian
```

to

```
FROM debian:sid
```

in my *Dockerfile*.

And don't tell me I could pinpoint the version of pandoc and mixing debian stable and unstable.

I used to do it, it was a hell to deal with the resolution of conflicts of dependencies it generates sooner or later.

Lesson learnt docker is the way to deliver

I don't know if you run macOS, windows, linux or BSD. I have tried to build my Dockerfile on freeBSD with podman with success.

I would definitely advise to install the required soft (latex, pandoc, graphviz, the fonts), a virtualenv for the additional dependencies (flask, pandoc helpers...) on distributions with a good upstream sync like freebsd or maybe ubuntu (really maybe because they can be surprising too), but it's easier for me to troubleshoot one environment as docker, and for all dusty distro like debian-stable, red hat, centos ... docker is my preferred way of delivering an imperfect packaged solution.

I control all the versions that are installed and given I do my whole test I should not miss such a bug another time.

This whole panic attack making me rethink the future of scam lasted 5 whole days. Days during which I thought I should really come back to more straightforward methodology because having piles of uncommitted changes and bugs that mixed together in a base of code you change a lot is a recipe for disaster.

You see I prefer small commits to fix small bugs. And releases, bug tracker are really helpful.

Dev corner

« Design »

Well, there was no design.

All I know is that web services came first and that all User Interface is based on calling them in ajax.

Web services are presented as a serie of form, one for each table, and the actions required to be filled in the request are the *input type=submit* button.

You can check how it is done in the lite suite I use to test part of the design.

http://127.0.0.1:5000/comment

created_at_time

jj / mm / aaaa --:-- 📅

id

user_id

comment_id

message

factoid

category

Comment ▾

Figure 11: Exemple of a REST access to the table comment

Synchronous

Since all my *events* meaningful (saving) are plugged on *onclick* events in javascript, it is a synchronous application ensuring that when my wife and kid interrupt me, the document is always in a sane state. So far, I love it so much because it actually acted as a safeguard in the here fore mentioned situations in real life during the writing of this manual, that I am reluctant to put an « autosave » feature.

But, nice to have would be a « CTRL+S » binding that does the save with a nice modal message.³

CAVEAT : hit update or CTRL+S often or you shall lose work.

Model

A careful examination of the entity relationship diagram will point at **dead** data such as the user table, or the factoid column that is not used anymore.

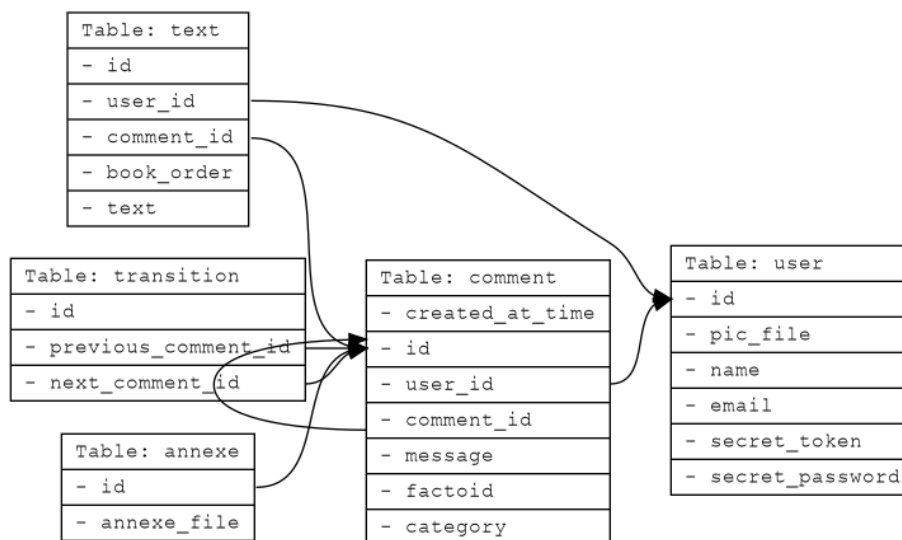


Figure 12: Entity Relationship Diagram

As you can notice the **comment** table is the center of the model.

³The save by hitting CTRL +S is actually in the base code but I don't know how to show it to the user. Lol.

HTML As A Model (HaaM)

Well, I made sure the web services serving the REST API would be in sync, by creating the model from the HTML forms served there.

The rule being one form, one table. One input, one column. And by adding some attributes to enhance the HTML I have an HTML dialect called **HaaML** as *HTML as a model Markup Language* to add sql related attributes and map them into the right SGBDR type.

Making me thus the “*piped piper of HaaML-in*” from which your mother guarded you from.

The only documentation of this language is of course the source code of scam.py itself.

Hence, to change the model, you have to change the HTML served to describe the model.

I still am not sure whether it's genius or stupid. But facts are : the web service always reflect with a 100% certainty the database model served by the application. Which as far as I am concerned is the right way to do it.

Creativity boosters (limitations)

Pretty much these limitations are due to some of my lack of skills.

You have the following important limitations :

- there is a one to one association between *annexe* (joint picture) and a *comment*⁴ ;
- there is a one to one association between *annexe* (joint picture) and a (markdown) *text* and the joint is on *comment_id* or *id*⁵ ;
- the table **transition** adds an edge between comments on the graph⁶
- the *factoid* column of the **comment** table is not used anymore ⁷;
- the **user** table is not used anymore ⁸;

⁴because I did not wanted to code a file explorer of the potential attachment (I hate front end development as much as back end one);

⁵I did messed up thing if I remember well, and I'm pretty sure that instead of a joint on *comment_id* I do it in one tiny **vicious** place of the interface on *id*.

⁶I am beginning to find this feature useless. Less is more, hence I'm thinking of removing it.

⁷I changed the interface by removing features, but I had the lazyness to write the migration script and change the custom SQL I wrote. Planning for migration scripts and version handling is quite a lot of work you know, that I will not prioritize because, I have the skills, but not the will.

⁸same thing as above

Serendipity

Also called « un planned features ».

Vanilla Markdown export with assets (gruik inside)

You noticed there on the page for the html export that there is a link for the full markdown export.



SCAM Manual

jul

Version: 202503



Figure 13: yep on this page

Actually, it is a modified markdown with with some custom pandoc processing with pandoc lua filters applied and a tad of pandoc magic.

So you will need this file to make your own builder, and also the images that are automatically generated upon calling the HTML view⁹. The picture needed to complete the markdown export will be located in the assets directory with the name `assets/$DB.annexe.*` where DB is the name of the DB you use (default is **scam**).

⁹I think I forgot to check if images were present before generating the PDF assuming people would check the HTML first. Stupid me.

Self embedded HTML

I am testing single HTML self containing page with the pictures embedded inside.

Test it for me, and if it works it will become the default and only HTML view.

Psycodelic experience

Most the making of this software *-due to an intense real life pain-* have been made under drugs such as ketoprofen, opium and more.

Pain and drugs totally change the way you code, if you want to go on foreward, you have to sharpen your mind through the pain to stay focused, ensuring a maximal brutality to hack your way.

As the maintainer of the source code I have been pleasantly surprised by how easy it was for me to understand what I did, even though it totally goes against the currently admitted best practices of full stack development (such as using fucking frameworks for every layers of the stack).

It is like discovering your true self in coding because you had to code so diminished intellectually that it prevented your past self to be too smart for your present self with more brain.

Since I don't like pain, it is an experience I don't recommend.

Liste des liens

<http://127.0.0.1:5000/> The landing page
<https://pandoc.org/MANUAL.html#pandocs-markdown> markdown extension used here is the pandoc one
<http://127.0.0.1:5000/book> it's now time to visit the HTML rendering URL
<http://127.0.0.1:5000/pdf> PDF renderer
<http://127.0.0.1:5000/svg> you consult the graph URL
<https://www.kernel.org/doc/Documentation/filesystems/overlayfs.rst> the kernel doc about overlayfs
<https://fr.wikipedia.org/wiki/Strace> strace
<https://github.com/jgm/pandoc/blob/main/src/Text/Pandoc/PDF.hs#L229> part that handles images in the PDF
<https://github.com/jgm/pandoc/commit/c1ccbc553fc756ffc6954574727bc4a1dd7be783> See commit here
<http://127.0.0.1:5000/model> Web services
<https://github.com/jul/scam/blob/main/test/load.py> the lite suite I use to test part of the design
<https://github.com/jul/scam/commit/1957c08d958d8a8b8e67324d19e6602fa7a1612c> The save by hitting CTRL +S is actually in the base code
<http://127.0.0.1:5000/model> web services serving the REST API
<https://github.com/jul/scam/blob/main/scam.py#L63> scam.py
<http://127.0.0.1:5000/book> page for the html export
<https://github.com/jul/scam/blob/main/mkdoc.sh> custom pandoc processing
<http://github.com/jul/scam> The home page of SCAM is on github