Photo by **Debby Hudson** on **Unsplash**

We’re living in a world of information overload. Do you know how they say, that the feather is mightier than the sword? Well, I’d like to add, that a gazillion of feathers lying around, some broken in two, some burned beyond recognition and some nails in a feather costume makes the feather loose all its power again.

In these times, you’re able to know just about everything, but it’s getting harder and harder to *find* the *right* and *up to date* information.

I think, this is true for the world in general, but also in the micro cosmos of most teams or departments nowadays.

There’s no problem in *having* information and documentation, but in having *good* and *usable* documentation.

This was (and in some parts still is) a problem we had with the IT-Department at KPS.

And this is how we tackled that.

Who we are

We’re the IT department of KPS. KPS is a consultancy company who helps its customers on their digital transformation journey. We support the complete infrastructure ranging from network access, datacenter management, routing, Linux and Windows server management, platform management to user support and DevOps project consultancy.

We’re divided into four teams to evenly balance the different aspects of our work.

Where we were

Originally coming from a merger of two companies team members were used to different documentation tools and ways of documenting. Thus, we had various places to look up information and various levels of quality throughout the different documents.

After we finally agreed to a central documentation space and tool, we migrated most of these documents. This made all information accessible at one place, but the lack of a standardized document quality still made it hard to find the right documentation.

And we struggled with it. So finally, we gathered together to form a documentation taskforce who should suggest a common standard for documentation for the IT department.

In this documentation taskforce we put members of all teams who knew something about their team's own previous documentation.

Starting is hard

So there we were. Left with a heap of documentation and trying to figure out how we should deal with it.

The first thing we did is to take a step back and look at *what* we want documented. What different types of documentation do we need for *our* daily work and for our *customers* daily work?

So we defined those different documentation types. For example, because we are supporting our customers and their workplace infrastructure we had some documents for self service. These documents showed our customers step-by-step explanation to common problems or how to install a new system or app on their computers.

Then, we have a team that builds up networking infrastructure needing infrastructure diagrams to better grasp what different parts they're managing and how to solve problems in that.

Of course we're doing meetings from time to time so there were meeting notes.

Et cetera, et cetera, et cetera.

Breaking down document types

So for each of these document types (in the end, we defined 13 different types) we asked ourselves what *elements* each type usually has.

Is there a page header in a guideline document? Absolutely. Does an infrastructure documentation have a schematic layout? You betcha. Does meeting notes have screenshots? Mostly not.

Mind you, these elements are *common*. It *can* happen, that a meeting note *has* some kind of screenshot, but it's probably unusual.

„Suggesting" instead of „Defining"

Remember that I said „(the taskforce) should *suggest* a common standard" and not *define*? That was on purpose.

Throughout the whole process we were absolutely sure about one fact:

Nobody wants to do documentation.

So we had in mind, that we had to make the barrier for documentation as low as possible. We wanted to make it *easy* for people do document *correctly*.

That's why we made suggestions on how to make good documentation, and didn't define it. It should be a team process.

The styleguide

After we had gathered all elements available in the different document types we needed to write down how these elements should look and what they should contain.

And in most cases, this is a matter of taste. However, there are certain aspects, that one should keep in mind when writing technical documentation.

- **Be aware of your audience**. Who will be reading your documentation? What set of skills do they have? Are they common to technical terms? Are they common to *specific* technical terms? What language is their native tongue?
- **Don’t be fancy**. You’re writing technical documentation, not a children’s book. You don’t need fancy colors, thirteen different fonts or sparkle effects (Though they’re pretty cool) Keep it nice and short
- **You’re not the reader**! In most of the time and most of the documents you will write, you won’t be the actual one reading them. Those are your colleagues, your users or managers. Write for *them*, not for you.
- **Keep your style**. Changing document formats, layouts or styles with every document is very hard on the eye. Your readers need to feel home with every new document. They need to know what is coming up.

The **Godot game engine documentation** has a good reference for writing and styling technical documentation. As it is a multipurpose game engine with a very diverse and international audience, they really put some thoughts in that.

Language!

According to **Wikipedia**, the world has an estimated amount of 5,000 to 7,000 spoken languages. But language goes further than just wether you’re speaking French, English, Latvian or **Shona**. Different people in different tongues not only have a different vocabulary but also different ways of describing and reacting to topics.

Thanks to English being the **language with about 1.2 billion speakers worldwide**, most documentation nowadays ist written in english and then usually translated from that.

However, simply writing in english usually doesn’t do the trick. Especially when you’re writing for people who are not fluent in it. I like to cite the aforementioned Godot docs writing guidelines here:

> A technical writer's job is to pack as much information as possible into the smallest and clearest sentences possible.

Smallest and *clearest*. This is a technical doc, it shouldn't transport emotions of the author (like a novel) or entertain (like a children's book). It should simply contain all required information in the most *clear* and *understandable* way.

But still, languages as spoken languages need to be taken into account as well.

If you're aware that your readers are not fluent enough to read a full-fletched english documentation (even if you write short and clear), you need to translate the documents.

As said before, we wanted to keep the barriers of documenting low so we can motivate and engage colleagues to actually write documentation.

Because most of our colleagues have german as their native tongue, we agreed to use german for internal documents that don't span outside our department.

However, we also write documentation for our users and we're an international company so for those documents we agreed on writing the docs *in english only* (just so that we don't have to write docs twice).

Problem with structure. The.

When a tree falls but nobody's around, does it make a sound? When a document is written but nobody finds it, was it written in the first place?

The task of properly structuring things is a life long task of humanity. There are literally thousands of apps in the „productivity" category that basically just try to structure things.

It's like the holy grail.

And, honestly, we didn't find it.

Instead we agreed on two things, which may look like it, but could as well just be a gold-

colored plastic cup.

The first thing was using *tags*. Tags are not a thing of the #internet, by the way. Tags have a long tradition in bibliography (where they're called index terms). They condense the content of a document into one or more words. Multiple documents share common tags and thus aid in navigation to related documentation.

Also, tags are a great way of suggesting search words for the internal search of your documentation tool.

So we recommended setting *meaningful*, *short* and *few* tags for each document. And to reuse existing ones (Our documentation tool suggests existing tags when typing the name of a tag).

The second thing was *suggesting* a documentation structure. I emphasized *suggesting*, because we really couldn't dictate any kind of structure for our department as a whole. We recommended a common top level structure, which was basically a department-wide section followed by team-wide sections. The team sections were free to be filled by the individual teams, while the common section was a bit more restricted.

We came up with the following suggestion:

- Common
- Team A
  - Topic
  - Subtopic (optional)
    - Document type
    - Document
- Team B

For each level we recommended using a macro (a programmed function in our documentation tool) that lists all child documents of the level. That way you have a nice overview of all your topics, or all document types or all documents of a certain document type for a topic.

We know, that this might not be optimal for each and everyone, but *at least it is a start*.

Keeping up with the community

> A wiki is a hypertext publication collaboratively edited and managed by its own audience directly using a web browser
>
> **Wiki – Wikipedia**

In 1994 Ward Cunningham developed the first Wiki system (WikiWikiWeb) and it introduced a whole new way of collaboration when writing documentation. It allowed to

- comment and discuss on documents
- make editing available to everybody
- introduced proper versioning for digital documents

Well, looking at how often I put links to the english Wikipedia into this document, I guess the system was successful.

We embraced the basics of Wiki-documentation for our documentation as well: We saw that we're one team so we didn't need any restrictions to documents. Everybody could see, edit and comment any document in our internal space.

I'm absolutely sure, that allowing edits from as many people as possible helps creating up-to-date and vibrant documents.

Additionally, with documents for users we allowed comments on them as well so they could suggest better documentation from their point of view.

Speaking of point of view: **Always** let somebody else read your document before it reaches a „published" state. Writing a document is dumping your brain into text. But it is *your* brain and that can very much differ from another person's brain.

Where we are

Finally, we put this all together into a new document that obeyed the style we suggested and presented that to our colleagues for comments.

Most of our suggestions and recommendations were directly approved and some required some more discussion.

In the end we created a „documentation guideline“ from this document that needs to be followed by the complete department when writing documentation. As this was a team effort and we kept the barriers low, the complete department was „in the boat“ for making this documentation happen.

Now, changing a massive load of already existing documents isn’t a thing that comes overnight, but we layed the ground for new documents and reworks of existing documents so gradually our documentation will shine more and more.

Parting suggestions

Let me give you some nifty (or not so nifty, you decide) tidbits from our guideline. Again, this is our documentation, suited to our needs, so effectively: YMMV.

Document states

It is essential, that documentation is *up to date*. However, we live in a world where being *up to date* is fairly impossible because everything changes everywhere at any time – especially in the IT business.

So documentation will be out of date at some point. The important thing here is to *let the reader know*, that it is.

For this, we’re using document states.

Every new document starts in the *draft* state. Things are highly uncertain, incorrect,

incomplete and sometimes incoherent. **Don’t trust this document**.

After a possible review process (which we kept as small as possible. Remember, barriers?) a document moves into the *published* state. This document is complete, correct and trustable.

Then, time happens and things get out of control.

At some time, a document might move into one of the *obsolete* or *rework* states.

Obsolete documents are basically trash. They’re not applicable, not correct or not usable anymore. They just outgrew their purpose. And that’s okay. Nothing bad about that. The reader just has to know that.

Usually, obsolete documents are – after a short discussion with the responsible team – removed. We decided **against** archiving on purpose, because in real life, archived documents are never read or touched again. So why keep them?

Documents in the *rework* state are still applicable, just not completely correct or usable anymore. They should be updated later (gamified documentation days ftw!) and the reader can still use them, but should probably be on the lookout for problems or errors that might arise.

Table of contents

Having a table of contents (*toc*) is **so** important. Not only for quickly referencing parts of a documentation, but also for a new reader starting to read the document.

A toc helps the reader to plan ahead how much time reading a document will take and how much information they can gather from a document.

Also, *meaningful* section headers are pretty important as well. A toc with bogus headers just could be left out at all (don’t take this post as an example. This is a post and no technical document)

Templates

Whenever possible in your documentation tool of choice, use templates. We created a template for every document type we had and included examples (using comment blocks which are not visible on the rendered page) for each section.

This way, authors knew what they had to do and could just drain their brain into the preformatted sections.

Don’t reinvent the wheel

Good documentation has been around for ages. You know these strange things made out of processed wood? With these letters on them? BOOKS! Yes, that’s what they call them.

When you create your style guide, check out how authors did things years and years ago. For example, the proper use of typography is the same in every technical book you find around. There are guidelines for when to use *emphasis*, `code styles`and so on.

Check out your favorite search engine and look those up. Or just, you know, open some of those… books.

Accessibility

I like to point out that you should be thinking of making your document accessible to everybody.

That not only includes absolutely important things like proper color choices for visually impaired folks and alternative texts for images, so that your documents can be understood by screen readers but also making sure, that all aspects of a document are accessible to readers and authors.

Take a graphic illustration for example. If you use an external software for creating that: Does every author have access to that software and to the base document you used to create it?

Use existing plugins for documentation tools that allow direct editing graphics inside a document instead.

And don’t forget about links! You can use them for reference or to break things with various aspects into different documents and link to them. For example, you might have documented the installation guide to a system and now you create a user documentation for it. You don’t need the installation guide there, but for a colleague who maintains the system, it might be interesting nevertheless. So: link it!

The future is yet to come

Documentation is never *over*, it merely keeps evolving.

To keep up to date and not loose tracks we didn’t break up the documentation taskforce. We made a recurring appointment that can be canceled, if nobody has anything to say, but it’s there to remind us to keep our documentation suggestions up to date just like our documentation itself.

Another thing we added was **a tool, that could regularly tell us about outdated documents**. Some (defineable) time after the document has been created, it will notify us so that we can validate it and set document states accordingly.

This is something we introduced, but not really tested well.

But in the end, this was a fun journey and a very productive one as well.

Yes, documentation is not *sexy*, but having the right, good, well readable and up to date document at hand when you’re deep in horse dung on a production system already comes pretty close.

*Originally published to **dev.to***