Ghost CMS

Exploration analysis of Ghost CMS

Wout Verbiest
8-16-2021

Preface

In this document, you can read an in dept analysis of Ghost CMS, a content management system that is calling itself as the number one node.js CMS for professional publishing.

You can read about the installation process of this CMS, support for extra features, plugins or modules, frontend & backend, the API & it's documentation, security concerns and some more general information about this content management system.

This paper took several months to complete. While writing, a new version of Ghost was released. While I was originally only going to focus on the version I started with, I found that the new version of Ghost fixed a lot of my original problems. I decided to keep both versions in this document to make it clear how the platform has evolved over time, how it got more diverse and more production ready. I am genuinely surprised with what the team behind Ghost has achieved over the past months.

Wout Verbiest, 16 august, 2021

Contents

Ghost CMS

Ghost, The company behind the CMS

Ghost is a fully independent, non-profit organization. The company does not rely on outside funding or shareholders. Everything they create, they give away as intellectual property under a free, open source license. They are a team of 21 people, with a revenue of 3.3 million US dollars in 2020. They also saw a growth of 200% in 2020. Truly inspiring company, product and philosophy in my opinion.

Setup & installation of Ghost CMS

Similarly to Wordpress, there are 2 ways to use Ghost. The first way being: the CMS is hosted on their own servers, and you, as the admin, pay a monthly fee to use those servers. But since this content management system is open source, there is also an option to use this CMS and spin it up in a Docker Container, run it locally, run Ghost in Ubuntu, or several cloud providers such as Digital Ocean, Linode or (of course) their own Ghost hosting service called 'Ghost(Pro)'.

To install a local instance of Ghost CMS, you need the ghost-cli, which can be installed via Node Package Manager.

```
npm install ghost-cli@latest -g
```

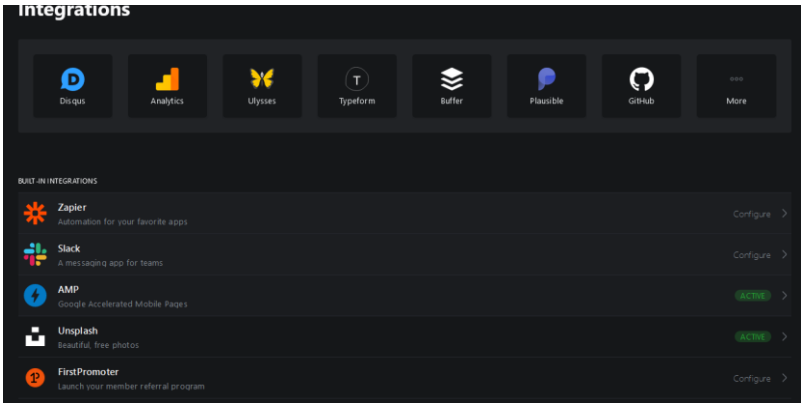
Then, in an empty directory, you can execute the following command in a terminal to generate a local setup of ghost

```
ghost install local
```

One annoying thing: I already had the latest version of Node (16.+) on my PC. This was not supported by Ghost for some reason. I had to downgrade to 14.5 to get it working.

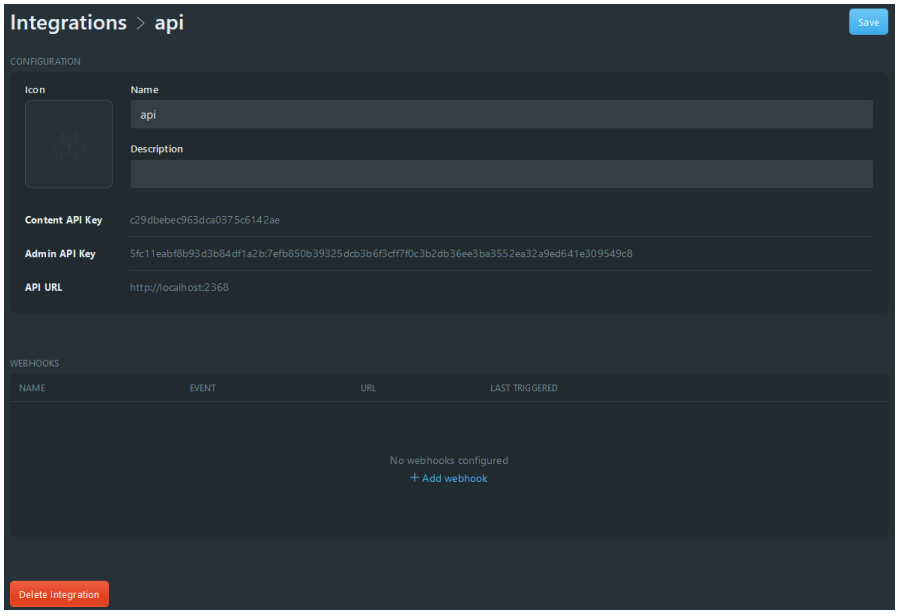
Extra features, plugins, modules

Ghost CMS has a large library of integrations. These integrations are rather small modules that can be added to the site. You can find these integrations in the admin area of the site, where you can also read how to add each module. (Mostly by adding a script to the footer of the site)



You can also create a custom integration via ghost. This is actually a piece of code that runs outside of Ghost. You create a webhook which listens for a certain event, which then triggers the URL every time this event happens.

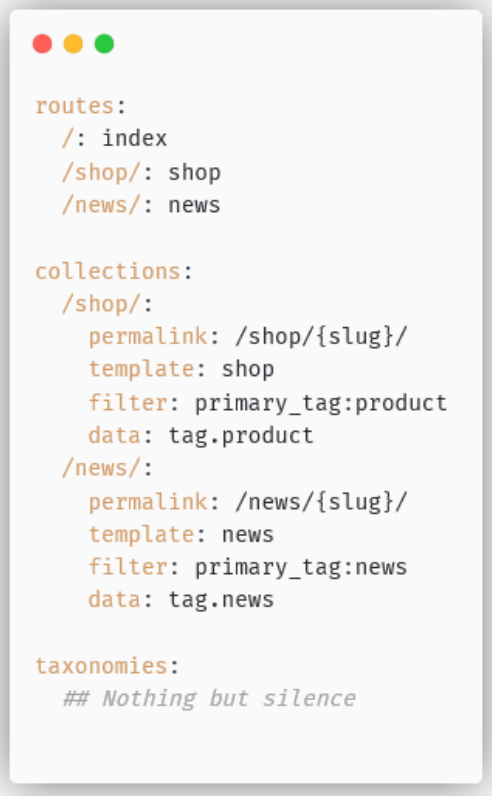
A slightly stranger choice I found was that this is also where the api keys are generated. So an external application is also a kind of 'integration' of the site.



Despite the large number of integrations on offer, their functionality is very limited. Often these integrations try to do something that Ghost is not meant to do. The ideal example is a web shop. There is no good integration for this. What happens is, you load via a few custom lines of code a new js file into the browser which does everything by itself, and forwards it to an external service. This is done via a kind of modal on the site. The link to get a product in the shopping cart is also a custom piece of html & js. No way to integrate this into the site in an easy way. Integrations to change the core functionality of the application are therefore not really included.

How to add custom content types

It’s not really clear how to add custom content types in Ghost. In the online version hosted on the ghost servers, it’s impossible, but on the code local, there is some kind of workaround. You can use the combination of custom routing, custom tags and collections build, in the settings/routes.yaml file.



```yaml
routes:
  /: index
  /shop/: shop
  /news/: news

collections:
  /shop/:
    permalink: /shop/{slug}/
    template: shop
    filter: primary_tag:product
    data: tag.product
  /news/:
    permalink: /news/{slug}/
    template: news
    filter: primary_tag:news
    data: tag.news

taxonomies:
  ## Nothing but silence
```

Routes are the basis of the application. Here you make a distinction between different parts of the site. in this case it is divided into index, shop and news. Collection is a separation from a route. For example shop has a permalink that leads to the product, so in this case slug is the name of the product. The template of this part is the name of the template you want to use from the theme. filter and data are things you want to give to the theme to render data correctly.

Front End

There is, by default, a front-end. In the settings/design section of your admin panel, you can explore a lot of free and paid themes.

Ghost 3.X

Themes are distributed in several different download places. After downloading them, you just need to drop the zip in the theme section of the admin panel, and everything is setup for you.

Ghost 4.X

Themes are distributed in a single place, hosted by Ghost. There is no need to download your theme! They have come up with a rather smart system that allows you to change themes by clicking Urls. Downloads happen automatically in the background. No more hassle with incorrect ZIPs from now on! If you want to create your own theme, or use an older theme, you can still upload a zip if you wan to.

API & backend

Ghost 3.X

This is where the flaws come. This CMS is great for blogging, especially on the Ghost servers, especially if you don't know anything about coding. It's a simple CMS where you can create a blog really fast, it looks great, it's really useful and I really enjoyed using the CMS. But then the struggles came.

Backend

The first thing I tried in the backend was add a new user (via the front-end) but since email service was not working, I tried changing it to my own smtp server on combell, sadly did not work. Tried to use several different email services like mailgun and gmail, but did not work either. At last I just hardcoded a user directly in the database. Otherwise, the backend is pretty clear how everything works and I have not really had any problems.

Api

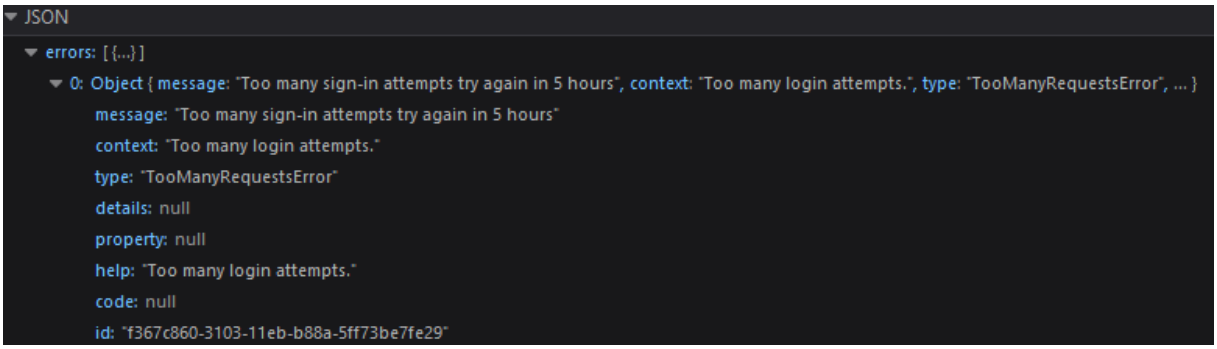
Content API

The next thing I tried was the content api, where you can read all the content. This worked great. There is no user login required, just a content key that is generated in the integrations section.

Admin API

After that, I tried the ghost admin api, which was super confusing to me. The admin key generated in integrations, is not the key you need to access the api, but some sort of code you need to use to generate a new code. I tried it several times, but it did not work for me. After a few tries, I got a 429 HTTP response and was blocked from the server for 10 minutes. The ideal time to make some coffee and think a bit more about it. When I came back, I read about admin authentication via user login. This was great! I opened up my postman, tried the endpoint, entered my credentials and I was in. I got returned a cookie with some session information. Great! This cookie could be used to access all the admin features, like reading, editing and deleting posts.

Time to try the api in front end! This was the part where I really wanted to throw my pc out of the window. The first thing I tried to implement was the user login, obviously. I tried over and over again, and kept failing, while I was getting 401 returns for the first 5 times and 429 for the ones after.



I could not send any request anymore. Not in postman, not in chrome, not in firefox. It was locked for 5 hours. Restarting the server did not help and searching in the code didn't result in anything either. I get that this is a security thing, but there should be a 'development' option in the settings to manually disable this.

So after waiting, my postman worked, but I got 401 request over and over again in my web application. After about 2 days, I saw 'Created' in my terminal, which is the text displayed after a successful login. I was so happy, but I didn't saw the cookie anywhere. Turns out, Ghost uses headers that are on a forbidden response-header list and where filtered out before even reaching the code.

> Browsers block frontend JavaScript code from accessing the `Set Cookie` header, as required by the Fetch spec, which defines `Set-Cookie` as a forbidden response-header name that must be filtered out from any response exposed to frontend code.

The next thing I tried where jQuery ajax, because I thought I might be different. Sadly did not work either. I tried requests from php, since it is server rendered. Nothing. Node? Nothing. Nextjs? Nothing. I was about to give up, until I remembered they have their own libraries for their api that you can use in your projects. I installed them, tried them, but quickly noticed that they don't support login via user and password either, only admin via the api key.

Member API

They also have a member api, released very recently, sadly it's in beta, has no documentation and is not really working great.

Documentation

The documentation about the api was not great, but not bad either. There were some strange parts, it kept bringing me to the version 2 page of the api for example, which is not the latest version. The authentication part actually said it supports fetch on browsers if you add a credentials tag, which didn't really change anything.

Making authenticated API requests

The provided session cookie should be provided with every subsequent API request:

- When making the request from a browser using the `fetch` api, pass `credentials: 'include'` to ensure cookies are sent.

The rest of the api was pretty clear, although it's a weird decision to split the api in several parts, admin, user and member, if they are overlapping in some parts.

Ghost 4.X

Content API

The content api stayed the same as in version 3.X

Admin API

The documentation of the Admin API in version 4.X has made it a lot clearer on how the admin API actually works. Give a better description of how authentication works,… Sadly, they did not change the Set-Cookie header. The JS client still does not support username/password login.

Member API

The member api seems to be fully removed from the new version of Ghost.

Security

I'm not sure about the security either. In the admin panel, a user with the role contributor can't really see, edit or delete any posts they are not an author of. They can write a new post or edit one of the posts they are an author of, but can not publish it. With the api in version 3.X, this is no problem. A contributor can see and edit all posts of all users. This can be a huge security flaw. It seems to be fixed in the new version of the api. A contributor can no longer access the posts he does not have access to.

Comparison with other content management systems

Ghost compares themselves with other content management systems like wordpress and medium, and to be honest, I agree on most of the points they make. They say they are not a CMS that tries to do everything, but that focusses on the blogging, the writing and publishing of the article and the really good SEO optimizations of your site. These are their really strong points. But they lack in a lot of other points, for example the api.

Pricing

The framework is free, but they are trying to upsell you on some sort of hosting service. The prices are between 29$ and 199$ a month.

Documentation

Ghost 3.X

The documentation can be a bit messy. Things are not always correct in the documentation. For example, I had a case where I wanted to use a foreach loop in my view. I defined the variable posts, which had to be in double quotes, and in the foreach loop I looped over "posts", which was correct according to the documentation, but I got as a result it looped over every single letter of posts instead of the posts. Turns out, sometimes it's necessary to use double quotes to use the variable, sometimes u can't use it.

Not every section is documented. For example the members api is not documented at all.

Ghost 4.X

Documentation has improved. A lot more parts make sense now. Although they made it a bit more difficult to find compared to the previous version of the site.

What's has changed in Ghost 4.0?

March 16, 2021: A great update for Ghost CMS has happened. Eight years after the first prototype, the newest version of Ghost, 4.0, was launched. With the update came some major new features. First, they made a dashboard. This makes it easier to see quickly, yet in detail, how your content and users are doing and which things are working and which are not. Memberships and subscriptions are finally available in this update. The new version of the content management system also makes it possible to send email newsletters to people who have subscribed to that service. Premium subscriptions are now also built in! They work via Stripe and are supported in 135 countries. Premium subscriptions are now also built in! They work via Stripe and are supported in 135 countries. To continue, it is now possible to create single use authentication links, and request cancellation feedback. Last but not least, they bought a company that made themes for Ghost CMS (IVEEL). These themes, which used to be premium themes, have now been released free of charge and open source!

Conclusion

After exploring Ghost, we can see that ghost is a CMS with some great points, but also big flaws. It's great for blogging, it has great SEO options, the theming market is great. It's easy to understand, it's fast to setup, it's quick to load… It's a great platform if you need something to write a blog. You can add some custom pages and there are a lot of good integrations.

Ghost is terrible if you want to make your own platform, or use it for something else then blogging. Api documentation is not really on point, and some sections aren't documented at all. There is not really a good way to create custom content types, apart from some workaround with the use of custom routing, collections and tags. Api libraries are not using the latest api (version 3) but the previous versions don't include all api routes. The api documentation is constantly trying to redirect you to v2 as well.

It's a simple CMS not really designed to do complicated things. It has a lot of potential and grew enormously the last year. Developers are working hard on releasing new features, but personally I think this CMS is not ready for using it in complicated custom applications.