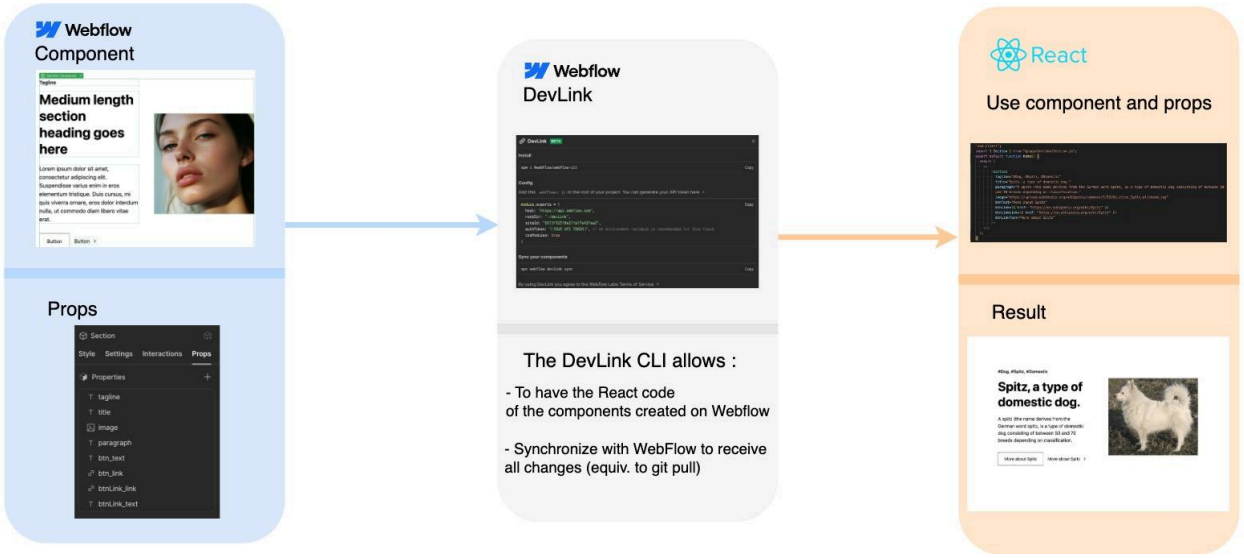
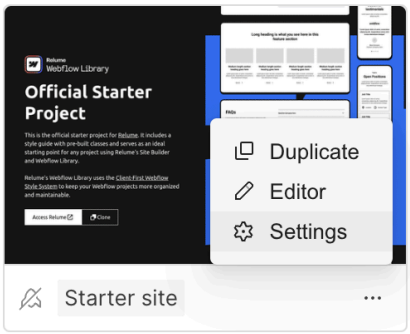
DevLink

DevLink is a WebFlow tool that allows integrating components created within WebFlow into our codebase (React only).

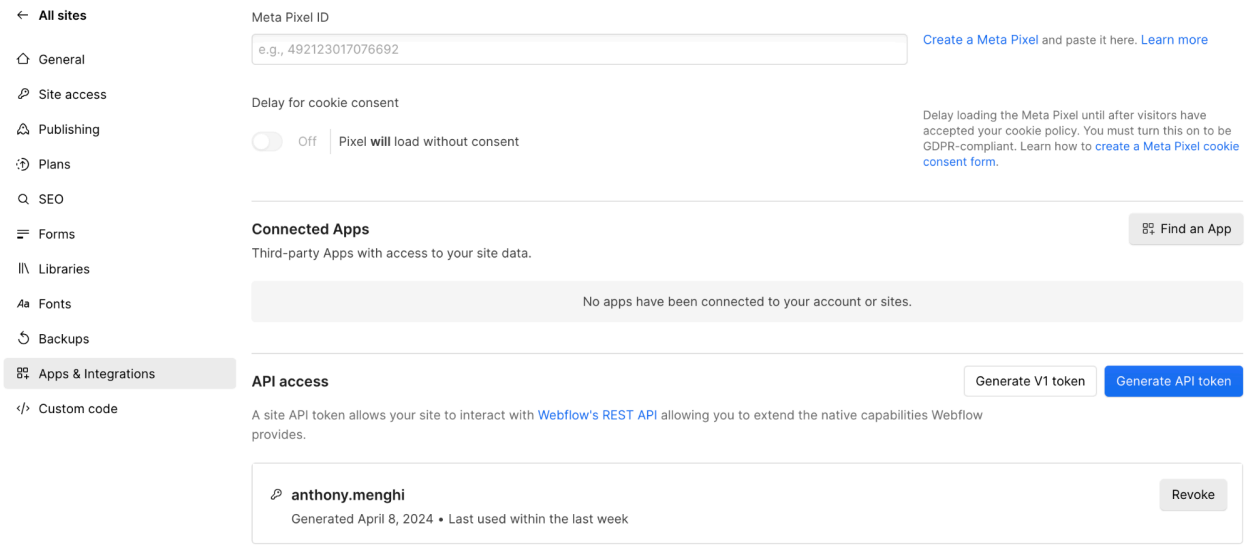


How

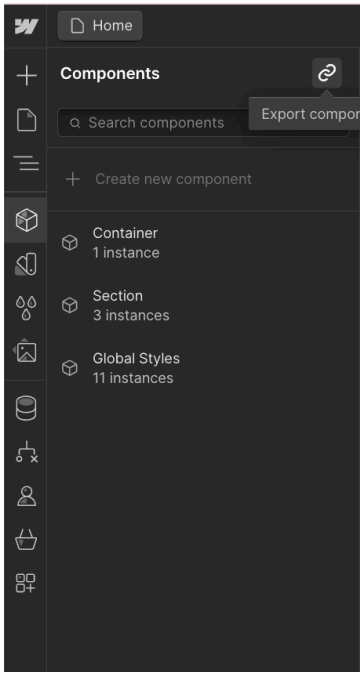
To do this, it's simple: just create a page on WebFlow and generate an API key (in the "Settings" tab, then "App & Integrations"), and then simply create a component.



React Export Test
View site



Next, go to the "Component" tab and retrieve the page's information.



Once that's done, go to your React codebase and install the WebFlow CLI (`npm i @webflow/webflow-cli`). Then, create a `.webflowrc.js` file with the WebFlow information (from the "Component" tab) and run the command `npx webflow devlink sync` to retrieve the components. Simply integrate the components into the application afterwards.

Unsupported Elements

- Collection List
- Ecommerce elements (Cart, Add to cart, Web Payments, PayPal)
- Lottie Animations
- Lightbox

Pros and cons

The positives are:

- It allows synchronizing the UI designer with the front-end developer.
- The front-end developer no longer needs to deal with the component implementation but can focus on the logical part, architecture, tests, etc.
- It helps "save time" on implementation.
- Props with default values are generated correctly.
- The generated code is understandable and modifiable (however, code modified in the React codebase is not synchronized with WebFlow - (this feature is coming later in the year).

The negatives are:

- Atomic design of components is not managed properly; the generated code needs modification but gets overwritten upon the next synchronization. And no synchronization from the codebase to WebFlow (style modification in the code)

  Reference :
  https://developers.webflow.com/data/docs/devlink-documentation-and-usage-guide

  “Remember that DevLink components cannot be “enhanced” with custom code on the dev side (or they can, but when syncing again it’ll override any changes) so we recommend structuring your components so that they can be used only as UI “presentation” components. However, you can pass it your own custom data with component properties (aka. props)."

- Some of the generated code "serves no purpose" and can be cumbersome.

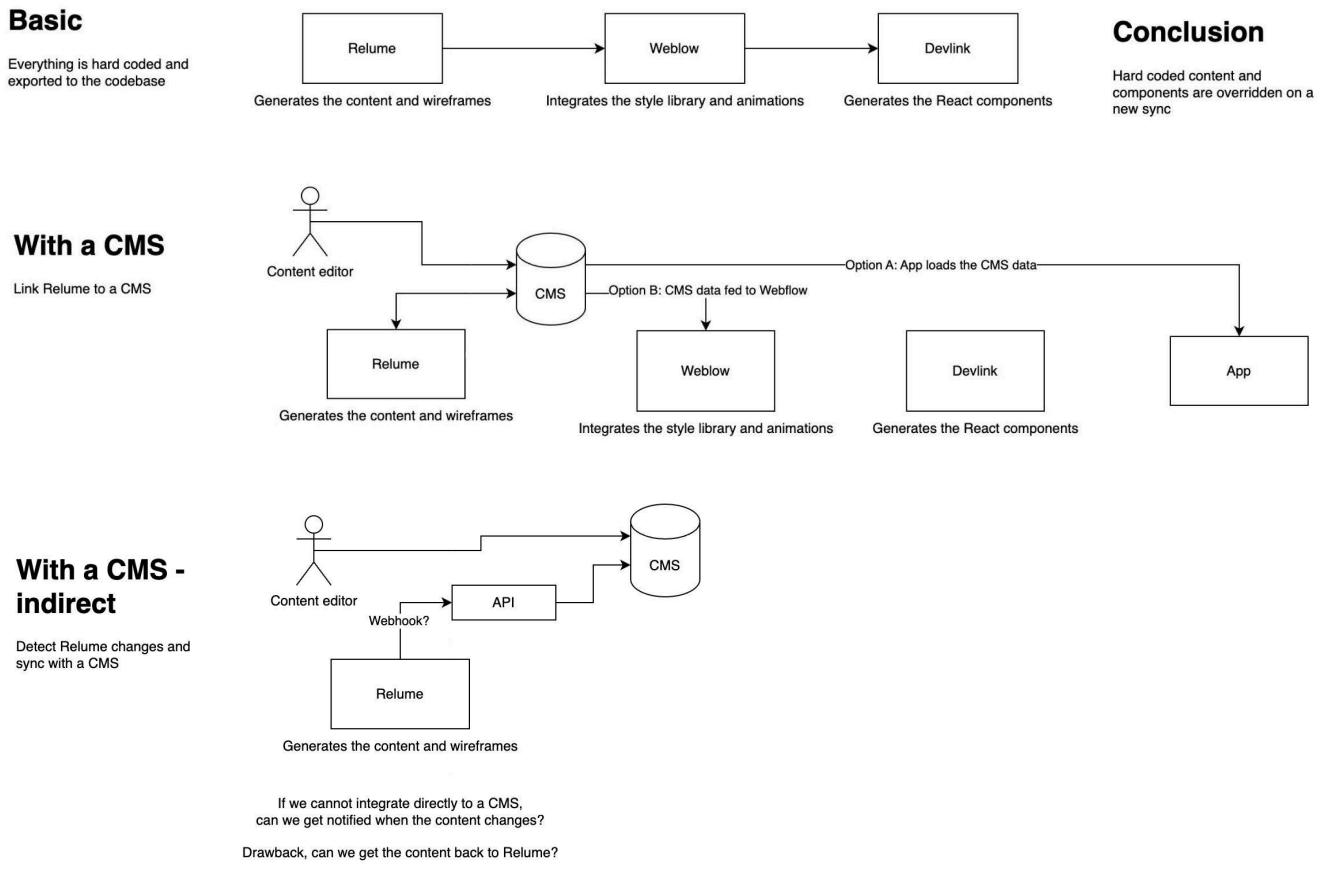
If component data is fetched from React, for an atomic component architecture, I recommend fetching the smallest components (atoms) and manually assembling molecules, organisms, models and pages.

Reflection on the connection between Relume, Webflow, DevLink, and the React application.

Relume

Relume is a fantastic tool; it allows for the rapid creation of a website using AI, effortlessly generates sitemaps and UX wireframes, and seamlessly integrates with Figma and Webflow through a simple copy-paste process. It works exceptionally well..

After a meeting with Marc, we reflected on how the tools should be connected to achieve the best experience and performance.



Possibility of connecting various technologies

We have three possibilities:

- Basic: We generate the site on Relume along with the components and text. Then, we transform it into Webflow for the site, and finally, with DevLink, we export it to have a React application. However, all the text is hard-coded. If you want to change the text or images, we have to redo the synchronization and rebuild.

- With a CMS: The goal here is to connect Relume directly or indirectly (via a webhook) to a CMS so that Relume takes into account the text from the CMS and saves the text with each generation.
    - Option A: Retrieve the data from the React codebase and pass the data as props.
    - Option B: Retrieve the text in the application and the CMS data in Webflow to export with dynamic data. However, this would require a new export every time there is a change in the information.

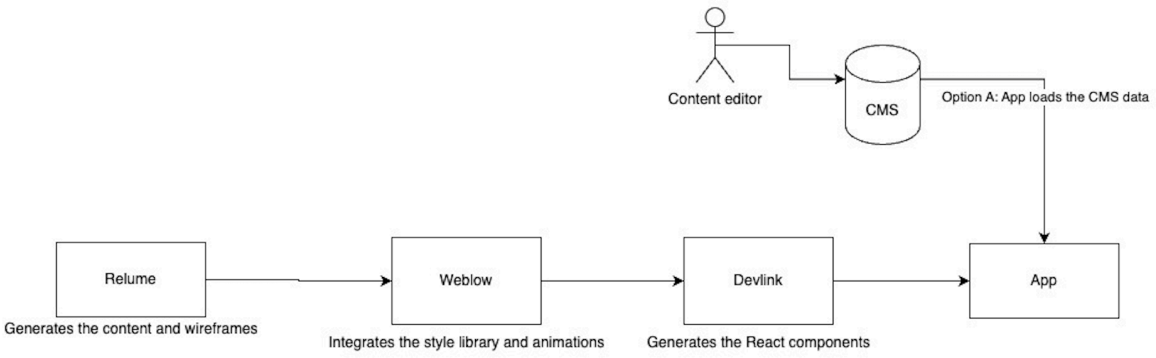
The Basic possibility works very well, this is what we did in the first part with the demonstration of the use of DevLink.

The possibility with a CMS is more complicated.

Relume is very closed, has no documentation and there is no resource on the internet for connecting Relume to a CMS or WebHook. Relume's documentation is done with Youtube videos, documentation which is not suitable for sustainable development or for a developer. A video from Relume's official channel with the title "Adding CMS Collections To Components | Relume Library" shows us how to create components with data with a CMS, but they do it not from Relume but from WebFlow.

Then, WebFlow offers their own CMS, it is stable and works very well. To connect an external CMS like Strapi or ContentFul you have to "code" the operation by adding Javascript code in WebFlow, which makes the tool more complicated to use.

So the best option is to create the sitemap, the wireframes on Relume, add them to webflow, create the styles, etc. Then, generate the React components with DevLink and on the React codebase, call the CMS to dynamically give the data in support of the generated components.


Content editor
CMS
Option A: App loads the CMS data
Relume
Generates the content and wireframes
Weblow
Integrates the style library and animations
Devlink
Generates the React components
App

Webography

- Doc ContentFul : https://www.contentful.com/developers/docs/
- Connect Relume with CMS : https://www.youtube.com/watch?v=6Qw06niEs0I&t
- Connect Webflow with external headless cms : https://www.youtube.com/watch?v=HMsrFzoZYr0
- Webflow documentation : https://developers.webflow.com/data/docs/devlink-documentation-and-usage-guide