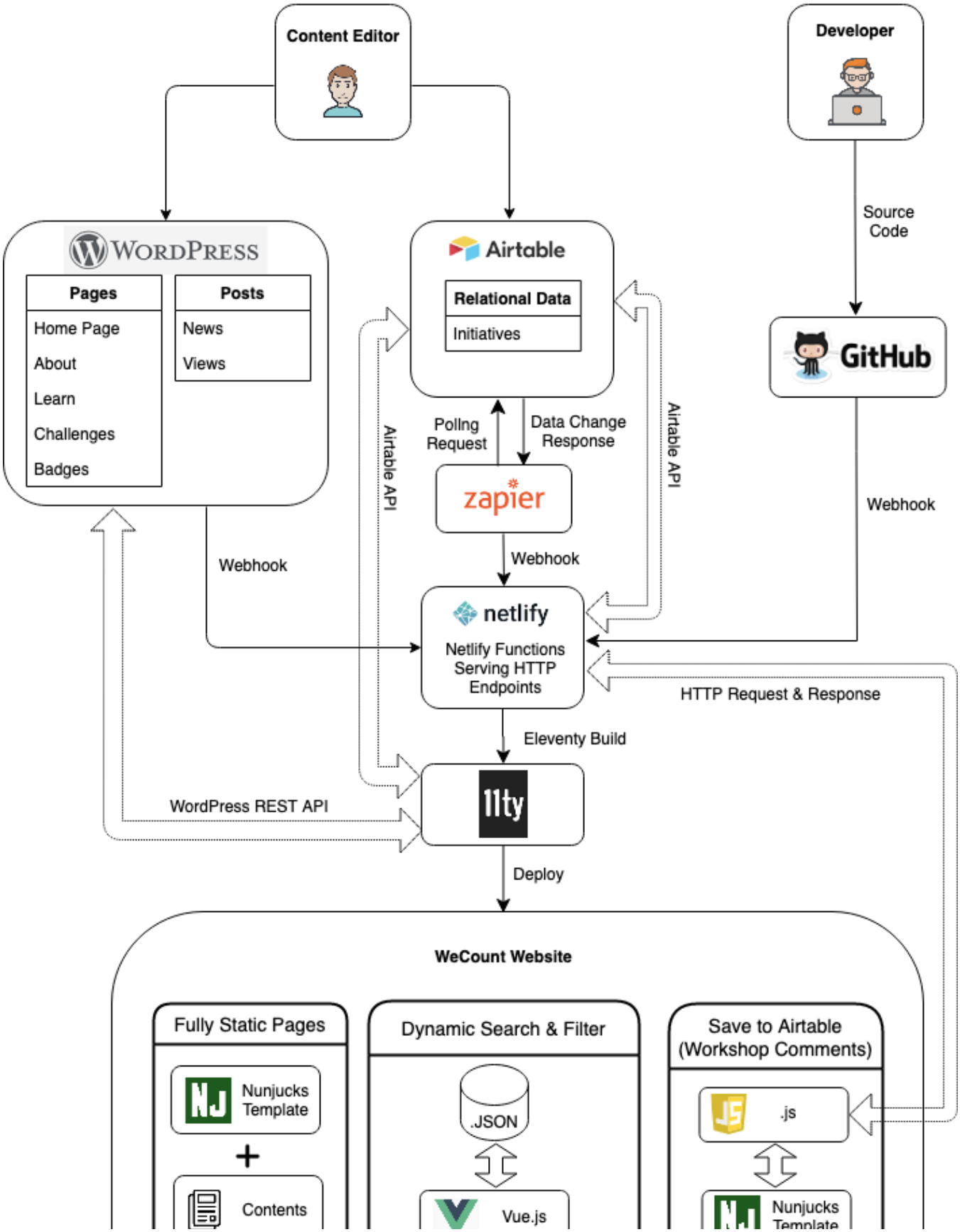
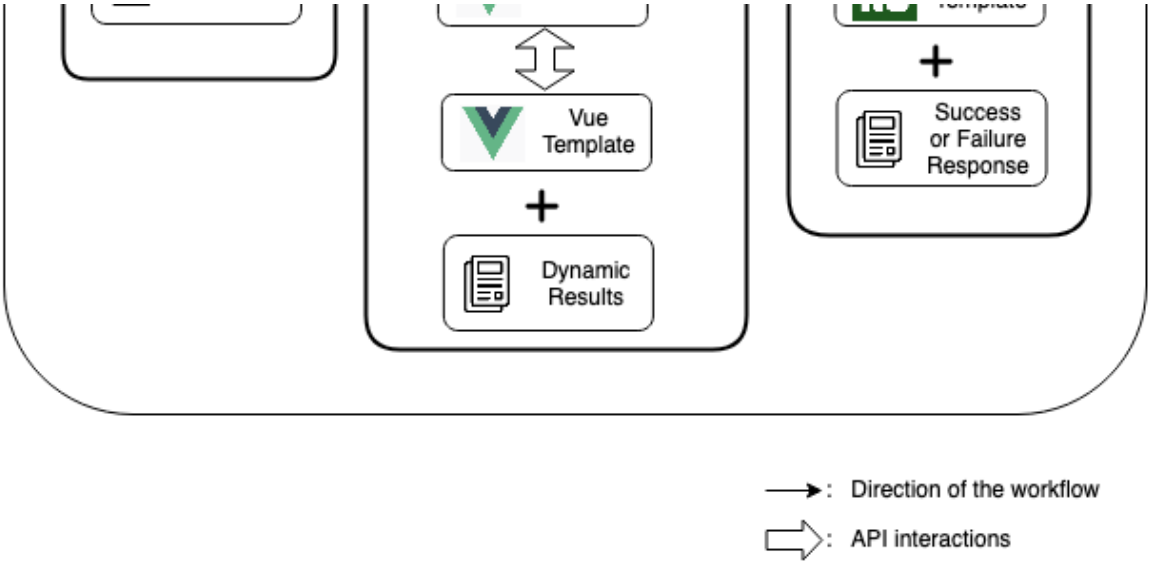
Architecture Rationale

Architectural Diagram

WeCount Website Architectural Diagram





: Direction of the workflow

: API interactions

Why WordPress and Eleventy

The first set of technologies selected for building WeCount Website was to use WordPress as the backend authoring tool and Nuxt as the front end website builder. Any content update in WordPress will trigger Netlify to re-deploy the entire website to reflect the change. The motivation of using this decoupled method instead of directly setting up the website with a WordPress custom theme was because:

1. The skill set of the primary developer includes NUXT, not WordPress;
2. Make sure the selected technologies could integrate the uncertain machine learning and AI features in the future.

Disadvantages of using the decoupled method:

1. Every re-deploy will take a few minutes to complete. Content creators need to wait for the deploy to complete in order to see changes being reflected on the website;
2. The website needs to support styles of WordPress blocks.

After evaluating pros and cons, the decision was to use the decoupled method.

A few months later, Nuxt was seen as overkill since it pulls down a number of unnecessary Nuxt modules into webpages that increases page sizes. Moreover, some complicated workarounds were added into the source code to support features that Nuxt didn't yet support, such as adding ARIA role indicating the current page. These features can be easily implemented using simpler platforms. An evaluation was done at the time and we switched from Nuxt to 11ty. These evaluation notes can be found in the meeting notes document dated Thursday, April 23, 2020 & Monday, April 27, 2020. Reasons that Eleventy is selected:

1. Most IDRC websites are generated using static site generators so that when a project reaches its lifecycle, only the front end website needs to be maintained and the backend server and database can be shut down;
2. Benefits of using a static site generator:
    a. Security: pre-rendered static webpages are greatly simplified and less hackable than a PHP server with a database to perform logic;
    b. Performance: faster to deliver pre-rendered static webpages;
    c. Simplicity and flexibility: The statically generated page contain the minimum but just right amount of html, javascript and libraries required by the page.

Why Airtable and Netlify Functions

Airtable is an online relational database that provides an API for database operations. Netlify functions are scripts written in javascript and deployed with Netlify. They provide server handlers and endpoints to interact with the website pages. Benefits of using them are:

1. Save the WeCount website from hosting its own server and database;
2. Save developers from learning a server programming language;
3. Secrets such as Airtable API key and application configs are securely stored on the Netlify server and only accessible by Netlify functions.

Airtable and Netlify Functions are currently used for building the "Initiatives" page for the part that saves user posted comments into Airtable. User posted comments are sent from the client side to the server endpoint written in Netlify function. When the “Post a comment” button is clicked, it makes a POST request to Netlify. Netlify is hosting a live JS webapp on our behalf, which is a “comments.js” handler written in as a Netlify function. This Netlify function then calls Airtable API to save the comment into Airtable. In a root folder of the project, there is a netlify.toml containing Netlify routing specifications.

What Triggers Netlify Re-deploy

Currently, there are 3 data sources that will trigger the Netlify re-deploy:

1. Any content change in the WordPress immediately triggers the redeploy. WordPress site includes a “JAMStack plugin” which is capable of making live calls out to the Netlify installation;
2. These actions in Airtable tables will notify Zapier, an integration tool, which then trigger the Netlify re-deploy. The Zapier polling on Airtable change is every 15 minutes. It means Airtable data change will be reflected on the website to a maximum time of 15 minutes:
    a. Any field value change to the "workshop" table;
    b. Any "reviewed" flag is flipped in the "comments" table.
3. New commits into the Github source code repository immediately trigger the redeploy. Commits into the **master** branch triggers the re-deploy of the production site while commits into the **dev** branch triggers the re-deploy of the development site.

How Long does a Netlify Re-deploy Takes

About 2 minutes.

Data Processing Methods on Website

Currently, WeCount website has 3 methods to process data:

1. Fully static page: Static pages are rendered by directly using Eleventy supported nunjucks templates with data;
2. Dynamic handling such as search and filter. The workflow:
    a. When Eleventy fetches search/filter required data from data sources, it saves data into JSON files that the search and filter are performed upon at user requests;
    b. The search is performed by Javascript written in Vue.js components. The dynamic search results are rendered in Vue.js template;
    c. Vue.js was selected because a primary developer has experience with Vue.js. This dynamic handling can be achieved through any other approaches or libraries. Eleventy offers the flexibility of using different libraries or strategies on different pages;
    d. There is no routing for dynamic handling pages. Using the search as an example, the search results page is static too. The diagram shows that the search box is a good old-fashioned HTTP GET FORM directing to /search/ . This is backed by search.njk rendering the search result, derived from index.json, which simply contains all the searchable content of the website in a 13K file. This holds some scalability risks given the entire site content needs to be served for a search, but the website is small and unlikely to grow vastly bigger.
3. Airtable operations: When user posted data or action needs to be reflected in Airtable, the user request will first be posted to a Netlify hosted endpoint. This endpoint then interacts with Airtable to perform database operations.

WordPress Instances

- Production WordPress API: https://wecount-cms.inclusivedesign.ca/wp-json/
- Production WordPress Admin site: https://wecount-cms.inclusivedesign.ca/wp-admin/
- Development WordPress API: https://wecount-dev.inclusivedesign.ca/wp-json/
- Development WordPress Admin site: https://wecount-dev.inclusivedesign.ca/wp-admin/

References of Involved Technology

- Eleventy: https://www.11ty.dev/
- Nunjucks Template Language: https://mozilla.github.io/nunjucks/api.html#environment
- Netlify: https://www.netlify.com/
- Netlify Functions: https://functions.netlify.com/
- Airtable: https://airtable.com/
- Airtable API: https://airtable.com/api
- Zapier: https://zapier.com/app/dashboard
- Vue.js: https://vuejs.org/