GSOC'24 PROPOSAL

By

Nandini Arora

On

EcoWagtail: Embracing **GOLD** Principles for Sustainable Web Experiences

Table of CONTENTS

Abstract

An average website produces **1.76g of CO2** for **every page view**; so a site with **100,000 page views** annually **emits 2,112kg of CO2**. The more complex a website is, the more energy it requires to load – and the greater its climate impact. This project aims to pioneer the integration of the GOLD (Green, Open, Lean, Distributed) principles into the development of Wagtail CMS templates.

Three aspects to focus on creating the Wagtail project templates are:

1. Have as **low of a carbon footprint** as possible.
2. **Accessible** to as many people as possible.
3. Follow all **performance** and **SEO** best practices.

Considering static site hosting, scale-to-zero backends, and hybrid client/server rendering, the proposal explores optimal tech stacks to develop green templates for a sustainable Wagtail journey and analyze them on the benchmarking suite.

As per the data of April 2023, **4349 Wagtail** sites produce a total of **8240 tons of CO2e/year**.

Goals & Benefits

1. **#Sustainability**: Ensuring Sustainable development by minimizing the carbon footprint and reducing the environmental impact of websites built with Wagtail.
2. **#Accessibility**: Applying best accessibility practices for the templates to stick to AA standards of Wagtail and thus working towards improvement.
3. **#SEO:** Implementing optimistic and SEO best practices to enhance search engine visibility of Wagtail-based websites.

Optimal Tech Stacks

1. Jamstack

Jamstack is an architecture designed to make the web faster, more secure, and easier to scale. Its core concepts include pre-rendering and decoupling. JAMstack has its advantages in reducing the Carbon Intensity:

1. **Performance & Efficiency**: JAMstack's architecture, centered around pre-rendering static pages and serving them over CDNs, boosts performance and SEO.
2. **Scalability**: The static nature of JAMstack sites allows them to be easily distributed across multiple servers worldwide without the need for complex scaling logic. This makes handling high traffic more efficient and environmentally friendly.
3. **Sustainability:** By serving pre-built files and reducing the need for dynamic server processing, JAMstack sites consume less energy, contributing to a lower carbon footprint.

Wagtail can be used as a headless Wagtail with Jamstack. The process of creating environment-friendly sites, leverages Wagtail as a content source (headless CMS) and uses modern frontend technologies (Next.js/ React.js) for presentation.

After the setup, Static-Site generators like Gatsby or 11ty can be used to create the templates.

A headless Wagtail setup, when combined with static site generation and effective use of CDNs, is likely to have a lower carbon emission footprint compared to a traditional Wagtail setup due to reduced server-side processing and efficient content delivery

Setting up headless Wagtail

```
# In your Django settings.py, add 'wagtail.api.v2' to INSTALLED_APPS

INSTALLED_APPS = [
    ...
    'wagtail.api.v2',
    ...
]

# Wagtail API url in URLS.PY

from wagtail.api.v2.router import WagtailAPIRouter
from wagtail.api.v2.endpoints import PagesAPIEndpoint

api_router = WagtailAPIRouter('wagtailapi')
api_router.register_endpoint('pages', PagesAPIEndpoint)

urlpatterns = [
    ...
    path('api/v2/', api_router.urls),
    ...
]
```

2. Using Static Site Generators (11ty)

11ty (Eleventy) is renowned for its straightforward setup and minimalistic approach to static site generation, making it accessible for developers of varying skill levels.

By generating static HTML at build time, 11ty ensures that content is delivered to users with minimal server processing, significantly enhancing site speed and reducing load times. This efficiency is crucial for both user experience and SEO rankings.

Static sites generated by 11ty require less computational power to serve content compared to dynamic websites, leading to lower energy consumption on the server side.

11ty's templating engine to create templates uses Nunjucks which is a templating language similar to Django templating language. Below is an example where content is fetched from headless Wagtail CMS in the react application. This snippet shows displaying Blog Post from Wagtail in the React app.

```jsx
import React, { useState, useEffect } from 'react';

const BlogPosts = () => {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    // Define the URL for the Wagtail API endpoint
    const apiUrl = 'http://your-wagtail-site.com/api/v2/pages/?type=blog.BlogPage&fields=title,body';

    // Fetch blog posts from Wagtail API
    fetch(apiUrl)
      .then(response => {
        // Check if the response is successful
        if (!response.ok) {
          throw new Error(`HTTP error! status: ${response.status}`);
        }
        return response.json();
      })
      .then(data => {
        // Update state with fetched posts
        setPosts(data.items || []);
      })
      .catch(error => {
        // Log any errors that occur during the fetch operation
        console.error('Error fetching posts:', error);
      });

  }, []); // The empty array ensures this effect runs only once after the initial render

  return (
    <div>
      <h1>Blog Posts</h1>
      <div>
        {posts.map(post => (
          <article key={post.id}>
            <h2>{post.title}</h2>
            <div dangerouslySetInnerHTML={{ __html: post.body }} />
          </article>
        ))}
      </div>
    </div>
  );
};

export default BlogPosts;
```

Develop Template

```
{# In src/posts.njk #}
{% for post in posts %}
  <article>
    <h2>{{ post.title }}</h2>
    <div>{{ post.body | safe }}</div>
  </article>
{% endfor %}
```

3. HTMX

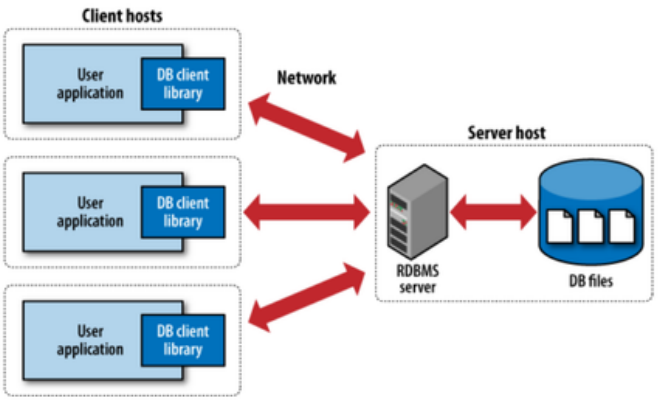
HTMX is used to add dynamic behavior to the web pages using attributes directly in HTML. It helps in creating interactive UX without using JS. It minimizes amount of data transferred leading to lower energy consumption, and decreases full page requests thus decreasing the load on server.

```
<!-- Blog posts container -->
<div id="blog-posts">
  <!-- Existing blog posts rendered by Wagtail -->
</div>

<!-- Load More button with HTMX attributes -->
<button hx-get="/api/v2/blog/load-more?start=5"
        hx-target="#blog-posts"
        hx-swap="beforeend"
        hx-indicator="#loading-indicator">Load More</button>

<!-- Loading indicator -->
<div id="loading-indicator" style="display: none;">Loading...</div>
```

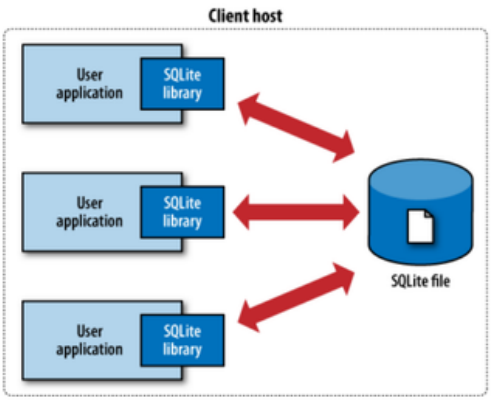
In the above snippet, **hx-get** specifies the URL to fetch more blog posts, **hx-target** identifies where to place the fetched content, **hx-swap** defines how new content should be inserted, and **hx-indicator** points to an element that will be shown while the request is in progress.

4. SQLite

SQLite doesn't require a separate server process to run, making it an excellent choice for projects. Its serverless architecture and efficient data storage minimize the energy required for database operations. The lightweight nature of SQLite reduces the need for extensive database server infrastructure, potentially allowing for less energy-intensive hosting solutions.



(a) Traditional client-server architecture



(b) SQLite serverless architecture

5. FaaS (Function as a Service)

Function as a Service (FaaS) and serverless are often referred to synonymously, but they have two specific definitions. While serverless refers to any category where the server is fully abstracted from the end-user, FaaS is a subset of serverless computing that's focused on event-driven triggers where code runs in response to events or requests. FaaS can be used with Wagtail to streamline various functionalities.

Google, Microsoft, and AWS are carbon net neutral and run on renewable energy wherever possible. When they cannot, they compensate for fossil fuel consumption by offsetting through tree planting, funding wind energy projects, and other initiatives that benefit the planet.

Tools such as Microsoft's Emissions Impact Dashboard can allow Wagtail to monitor how much carbon is produced by all the services used across Azure. This allows us to test whether any resources can be removed to reduce environmental impact. The Microsoft Cloud for Sustainability API allows us to apply the Emissions Impact Dashboard data to build their carbon-reduction tools.

6. React.js with Wagtail

React.js can be easily used to create the Wagtail project templates. React's modular approach to creating reusable components helps in reducing the development time and enhancing sustainability.

React uses a Virtual DOM to minimize direct manipulations of the DOM, which can be costly in terms of performance.

While single-page applications (SPAs) can face SEO challenges, React can be rendered server-side (using tools like Next.js) to produce fully rendered pages for search engines, mitigating potential SEO drawbacks of client-side rendering.

Accessibility Considerations

1. Color Contrast

1. Using Dark Mode would benefit users who struggle with light sensitivity, eye strain, and other visual conditions.
2. The contrast for the text on the background must be at least **4.5:1**.

2. Alt text field

1. For normal fields, add an alt text field to the image's panel.
2. For StreamField, add an extra field to the image block.
3. For rich text – Wagtail already makes it possible to customize alt text for rich text images.

3. Semantics

1. Using correct semantics helps the screen reader know what properties the tag carries. Using <h1>, <h2> instead of everywhere.
2. Although the styles applied may look the same but wrong semantics must be avoided as it's not a good practice.

4. Keyboard only

1. As HTML is fully keyboard-only, all standard HTML controls offer built-in support for keyboard interaction. Using proper semantics eliminates the issue completely.
2. When implementing custom functionalities, it's crucial to ensure elements are focusable and that keyboard focus is managed logically.
3. Stick to browser-independent events like **onclick** that cater to various input methods, including keyboard interactions.

5. ARIA

1. Using ARIA landmark roles to define page structure, making navigation easier for screen reader users.
   ** ... **
2. Applying ARIA roles and properties to dynamic content updates, ensuring changes are communicated to assistive technologies.
   ** Status here. **

6. Miscellaneous

1. Referring to the Wagtail official documentation, Adding validation rules to prevent empty headings in rich text and StreamField blocks, ensuring meaningful structure and navigation.
2. Form accessibility can be enhanced by avoiding layout helpers like **as_table**, distinguishing required fields visually, and grouping related fields with **fieldset** and **legend**.

Low Carbon Emission Strategies

Taking reference from the **Sustainability Roadmap**, Some improvements can be adapted for project templates in Wagtail including Media, Data Processing and thus making sustainable default templates for Wagtail. The Strategies include:

1. Media

a. **Responsive Images:** As screen size changes, it is better to change the resolution of images which helps in optimization. Two available features in Wagtail for creating responsive images are given below:

i. **{% srcset_image %} tag**: To specify multiple widths, enhancing browser efficiency in selecting the most suitable image size.
ii. **{% picture %} tag:** For employing different image formats, and optimizing for various devices.

b. **Using Next-gen Image types**: Using AVIF & Webp image types thus making the website much lighter. Also use of Image compression tools like Shortpixel, TinyPNG, and ImageOptim minimizes file size without unwanted degradation of image quality. Using vector images is also an optimal solution.

c. **Using CDN**: A content delivery network (CDN) places content and services closer to the user and is considered best practice among web developers. The CDN can also be used as a distributed cache avoiding all the processing that normally happens in the CMS.

d. We could also implement lazy-loading, YouTube embed links, and code-splitting techniques while using React.js + Wagtail for creating templates.

2. Template Fragment Caching

It's a technique that involves storing specific parts of the template during the first query and then retrieving easily on further requirements. The above technique can be implemented as mentioned in this **tutorial.**

Template caching is beneficial as it significantly reduces the number of database queries, lowering the server's workload and improving page load times. This is especially important for websites that grow in content and user traffic, ensuring a smooth and efficient user experience.

SEO and Performance

Page title and Meta Description

1. To enhance SEO we can leverage the inbuilt promote tab, for customizable page titles and Search Description fields.
2. We can consider adding Open Graph meta tags like **og:title**, **og:description**, **og:image**, and **og:url** in templates for better content representation on search engines and social platforms.

We can create robots.txt files for websites. It is a file that is usually located on the root of the website and its purpose is to instruct web robots (search engine robots) how they should crawl the website.

Creating a sitemap

1. The first step is to include the Django.contrib.sitemaps in the INSTALLED_APPS in your settings.
2. The next and final thing is to include the path where the sitemap will be in our urls.py.

Dynamic META Tags

1. Ensuring that each template dynamically generates meta tags like <title>, <meta name="description">, <meta property="og:title">, <meta property="og:description">, and others based on the content of the page.
2. This helps search engines understand the content of each page, making it easier for them to index and rank the pages accurately.

Canonical URLs

Implementing canonical URLs in the templates to prevent issues with duplicate content. This is particularly important for sites with pages accessible via multiple URLs. Adding a **<link rel="canonical" href="{{ page.full_url }}">** tag in the **<head>** section of the templates can signal to search engines which version of a page is the primary one, helping to consolidate ranking signals and reduce potential search engine penalties for duplicate content.

Structured Data

Incorporating structured data (Schema.org) into the templates to provide search engines with specific information about your pages' content. This can be done using JSON-LD or microdata within the HTML. For example, in a blog, we can use Article schema to provide details like the author, publish date, and article section, which can enhance the presentation of the pages in search results.

To measure SEO on a Wagtail site, we can integrate tools like Google Analytics and Search Console for insights on traffic, user behavior, and search performance. Leverage Wagtail's features for sitemaps and redirects, and use external tools for keyword tracking and backlink analysis to refine your SEO strategy.

Project Timeline

1 May - 26 May	• Community Bonding period. • Interacting with the mentors and other contributors to understand more deeply about the strategies to apply. • Going through the codebase and documentation for a better knowledge.
26 May - 3 June (1 Week)	• Design the Low-carbon UI/UX for the project templates 3-4. • Review from the Mentor and the community members. • Conduct Usability to test to check which design works perfect. • Make iterations and finalize the Designs.
4 May - 14 June	• Frontend development begins. • Creating the static pages based on finalized designs using React.js.
14 June - 28 June	• Working on second template with 11ty as static site generator.

28 June - 7 June	• Comparing the two for lower carbon footprint. • Study deeply and incorporating the best.
8 July - 20 July	• Working on the third Wagtail template using HTMX, CSS and JS. • Here also the pages would be with simple UI/UX and low carbon footprint.
21 July - 28 July (1 Week)	• Using Benchmarking tools to compare again, Lighthouse, Green Metrics Tool.
29 July -7 July	• Creating the final template with serverless architecture or simple headless Wagtail. • Thus we created 3-4 templates incorporating all the necessary features.
7 Aug - 14 Aug	• Finally comparing and determining the best techniques for sustainable templates.
15 Aug - 26 Aug	• Working on the Documentation. • Additional features or changes to be implemented.

About me

Name	Nandini Arora
Degree	Bachelor of Technology
Year	Sophomore
University	Indian Institute of Technology BHU, Varanasi
Country	India
Time Zone	Indian Standard Time (UTC+05:30)
Email	nandiniarora584@gmail.com
GitHub	@nandini584
LinkedIn	Link
Slack	Nandini Arora
Resume	Link

Hello Wagtail Team, I am **Nandini Arora**, an Engineering student at **IIT BHU, India**. I got introduced to the world of programming in my high school, where writing programs and solving problems excited me and Computer Science as a subject captured a lot of my interest. I have worked with multiple people on multiple projects as I have always been up for work, where I have the opportunity to learn.

Building Indie Projects are always fun but it never provides a chance to work on large codebases which are made for mass usage. **Open Source Development** has given me a chance to work on such large-scale projects with so many peers.

Coding Experience

I have worked with many Programming languages including **JavaScript, C, C++, Python, Java, and Golang**. I am also well-versed in JS frameworks including **React.js, Next.js, and Astro**. Django was something new to me but while working with wagtail I realized it was the chance for me to gain a new skill. I have also worked with databases like **MySQL and MongoDB** and have also written **Jest** unit tests.

I have made multiple projects using **React.js, Firebase, Nodejs** and also know the **MERN** stack. I have primarily worked in Web development and have experience with many JS frameworks.

Wagtail

My experience with Wagtail has been really amazing, Wagtail has been my very first Open Source Organization and I owe my learnings about open source and community to Wagtail mentors who have helped me through the journey. I have been contributing to wagtail since December beginning, and have a basic idea of what Wagtail as a CMS Platform is trying to provide. Below are some of my Contributions to the Organization:

#11307	**ActionController allows non-input controlled elements to call select method**	**Merged**
#11306	**Ensure dialog (modal) shows the close button with a visible cross icon**	**Merged**
#11372	**Label-visible content mismatch 'Admin' and 'Edit your account' in sidebar**	**Merged**

#11411	Updated the anchor sizes	Merged
#11458	Scss stylesheet clean up	Merged
#11411	Follow Up	Merged
#11449	Broken design update accessibility	Ongoing
#11077	get_gravatar_url: allow custom default	Ongoing

Past Projects & Experiences

- **MYRO BOT:** As part of my internship, I got to handle a project from Dubai where I had to create a Website for the company **MYRO**. MYRO is a Wall painting Robot, which can be controlled using an external tablet device. I designed the UI, coded it in React.js, used Firebase for authentication, Firestore for database, and Email.js for custom Blog writing and contact form respectively.

- **Saras-Gpt:** I got a chance to work with alumni of our college and built a PWA for the entrance exam aspirants to create a revising tool for them. The app analyzes the performsnce and provides a report after every assessment. The questions are retrieved from PostgreSQL. The frontend of the app is built on React-native.

- **Aero-Ease:** This particular project was built in collaboration of a team of 4 people for the Inter IIT Tech Meet. Aero-Ease is a platform used during flight cancellation, When a flight is cancelled the passengers are to be redistributed and accommodated according to a preference order. I majorly worked on the Frontend part of the project. The technical specifications included working with REST API, & TypeScript.

Why do I wish to participate in the GSoC?

Ever since I learned my first programming language, **JAVA** I have been fascinated by computer science, and have always wished to explore more and more into it. Learning a new programming language has always been an interesting experience that attracted me to this field. But what could be more interesting than all these functioning beautifully together?

Open Source provides an amazing opportunity to **work with a huge industry-level codebase** and it feels like working with multiple people in a team with **mentors to guide** around.

GSoC is that opportunity for me where I can get all this experience. Being a self-learner programmer without any guidance can be difficult and I have done it for a year or so, this could be a chance for me to work and learn from experienced mentors who have been there in the industry for a long time.

Why do I wish to work with Wagtail in particular?

I want to work with Wagtail in particular because it's **my first Open Source Organization** where I made meaningful contributions. I have learned a lot in these **months of contributing to the organization** though I understand I have a long way to go. I have mentors like **Thibaud & LB**, who have patiently explained things and helped me out with the code.

I also want to work in Wagtail because it's a **CMS platform**, I have always used them to make my work easier and also had the idea of making one, but I couldn't be successful due to certain reasons, but through Wagtail there's chance for me to work around.

Why this particular project?

I am enthusiastic about working on this project, given its current relevance. It ensures How we as developers can contribute to a greener and safer environment.

As I went through articles, case studies, and proposals on Low carbon emission my interest started growing, I realized that investing my time in learning about Sustainable Software Development is a worthy pursuit. It blew my mind how a minor optimization of Images or implementing sustainable coding practices can create a huge impact.

I want to contribute to this noble cause and assist Wagtail in achieving its sustainability goals. I am also excited about the sustainability team that Thibaud proposed, and I would love to be part of it and work with other sustainability enthusiasts.

Availability

I will be freely available to work on the project during my summer holidays i.e. from May 11th to mid-July and will be able to provide **30-32 hours a week** or more. Even after my summer vacations I will continue to work on the project.

I promise to work diligently and put in my best efforts.

Thankyou.

References

1. Wagtail SEO tips
2. Making Wagtail more SEO-friendly
3. Ways to create carbon Websites
4. Sustainability Considerations
5. Accessibility guide
6. WCAG
7. Cloudflare and FaaS
8. JAMstack headless CMS
9. Headless Wagtail
10. Sustainable Web Designs
11. Sustainability Roadmap
12. Greener coding - Making a 'gold' reference configuration with the Wagtail bakery app · wagtail/wagtail · Discussion #8843 (github.com)
13. Quantifying Greenness
14. Web sustainability Guidelines
15. Wagtail Website Emissions
16. Improving the sustainability of digital products
17. Jamstack and Wagtail
18. Sqlite docs
19. HTMX
20. 11ty