WIKI INFOGRAPHICS

TECHNICAL PLAN V1 (30 June 2024)

1. Overview

Summary

Wiki Infographics is a multi-year commitment from Wiki Movimento Brasil. This initiative aims to leverage structured information within Wikimedia projects to create informative and visually engaging infographics. These infographics will be available in both fixed and dynamic formats and will be released under an open license, making them freely accessible to the public.

Objectives

- To develop a methodology for extracting and utilizing structured data from Wikimedia projects.
- To design and implement a platform for the creation and dissemination of high-quality infographics.
- To engage people using educational and informational open-licensed infographics.

Scope

- Extraction and processing of structured data from Wikimedia projects.
- Production of a variety of infographics, including those tailored for different topics and audiences.
- Establishment of guidelines and best practices for creating and sharing infographics.

2. Requirements

Features

Data retrieval

The software must be able to connect and retrieve structured data from Wikimedia projects. This capability should support different types of data and different sources (Wikipedia, Wikimedia Commons, Wikidata etc). Users should be able to access the data they want efficiently and with minimal effort.

Data processing

The software needs robust tools for cleaning and preprocessing the information retrieved. This includes, but is not limited to handling missing data, normalization of data formats, and enriching datasets with additional or necessary context. The goal here is to prepare the data for visualization.

Data visualization

This is the core of this application. The software must support innovative and interactive forms of visualization of the data, providing capabilities such as zooming, panning, and filtering, to enable detailed exploration of the data. Users must be able to change color schemes, labels, and legends, tailoring the visualization to their needs.

User interface

The software should provide an intuitive interface where users can create and modify the visualization configurations. The user interface needs to be designed to ensure a good and efficient user experience through an intuitive and user-friendly layout. It should feature clear navigation with descriptive labels and icons, and offer flexible configuration options, such as customizable color schemes, labels, and legends and exportations. As per WMB's development practice, accessibility is paramount, as it is responsive design for mobile devices. Informative messages, contextual tooltips are to be implemented and increase usability. The software should have its performance optimized and any error that might occur should provide clear instructions for the users and guide them towards proper help channels.

3. Technical Stack

The tools and technologies are evaluated based on the following criteria:

- **Features:** The range and depth of visualization capabilities provided by the tool.
- **Performance:** How well the tool handles rendering and interacting with large datasets.
- **Scalability:** The tool's ability to scale with increasing amounts of data, complexity, and types of visualization.
- **Security:** The level of security provided, especially dealing with automated calls to the tool.
- **Usability:** Encompasses ease of use and includes the overall user experience, effectiveness, efficiency, and satisfaction over long-term use.

Infographic creation solutions

Tool	Stack	Ease of Use	Integration with Wikimedia projects	Performance
Tableau	Python, JavaScript, C++, Hyper database	Moderate	Via APIs/Connectors	High
Microsoft Power BI	Microsoft technologies, Popular front-end technologies	High	Via APIs	Moderate
Google Data Studio	JavaScript, Google Cloud Platform	High	Via APIs	Moderate
Infogram	HTML5, JavaScript, PHP	Very High	Via CSV/APIs	Moderate

Wikimedia projects integration tools

Tool/Technology	Stack	Ease of Use	Integration with Wikimedia projects	Performance	Cost
SPARQL Endpoints	Java, RDF	Low	Direct	High	Free
Wikidata Query Service (WDQS) + UI	JavaScript, RDF	Low to Moderate	Direct	High	Free
Pywikibot	Python	Low	Direct	High	Free
Wikibase	PHP, JavaScript	Low	Direct	Very High	Free
SPARQLWrapper	Python Library	High	Direct	High	Free
Wikimedia REST API	Python, JavaScript, PHP	Low to Moderate	Direct	High	Free

Infographics packages

Tool/Technology	Usability	Features	Security	Scalability	Performance	Cost
D3.js	Moderate	Very High (custom visualizations, animations)	Moderate (depends on implementation)	High	High	Free
Chart.js	High	Moderate (pre-built charts, easy to use)	Moderate (depends on implementation)	Moderate	Moderate	Free
Highcharts	High	High (interactive charts, wide variety)	High (commercial grade)	High	High	Free for non-commercial, paid for commercial
Plotly.js	Moderate	High (supports a wide variety of charts)	High (commercial grade)	High	High	Free for basic, paid for advanced features
Vega/Vega-Lite	Moderate	High (declarative, supports a variety of charts)	Moderate (depends on implementation)	High	High	Free

Flask Vs Django for back-end

Criteria	Flask	Django
Framework Type	Micro-framework	Full-stack framework
Learning Curve	Gentle learning curve	Steeper learning curve
Flexibility	Highly flexible, allows customization	More rigid, "batteries-included" approach
Built-in Features	Minimal, requires third-party extensions	Comes with many built-in features
Routing	Simple, explicit routing	Automatic URL routing with views
Data Handling	Customizable, can use Pandas directly	Integrated ORM for complex data handling
Admin Interface	Not included, requires external packages	Comes with built-in admin interface
Community Support	Strong, but smaller than Django	Very strong, extensive community
Use Cases	APIs, Microservices, Data visualizations	Large web applications, CMS, e-commerce
Performance	Lightweight, potentially faster	Heavier, potentially slower
Documentation	Comprehensive and easy-to-navigate	Very comprehensive, might be overwhelming
Modularity	High, better for small components	Less modular, more monolithic
Development Speed	Faster for small, focused applications	Faster for large, feature-rich applications

Summary of important points:

- **Flexibility and Simplicity:** Flask's microframework nature allows us to build a highly customized and modular application. This flexibility is crucial for our project's need to handle specific data fetching and processing tasks.
- **Learning Curve:** Flask has a gentler learning curve, facilitating quicker onboarding and faster initial development, which is beneficial given our project's specialized nature.
- **Modularity:** Flask allows for a more modular approach, making it easier to add and manage specific components such as SPARQL data fetching and infographic generation.
- **Performance:** Flask's lightweight nature can offer better performance for our focused tasks, such as data processing and infographic generation.
- **Custom Data Handling:** Flask's flexibility allows us to directly use powerful data handling libraries like Pandas, which aligns well with our data-centric project.

Similar tools on Wikimedia

Tool/Technology	Link	Description
Dataviz, by Stevenliuyi	dataviz.toolforge.org	From the docs *"This web app provides various easy-to-configure tools for visualizing Wikidata SPARQL query results"*.
Wikidata Charts, by Germartin1	wikidata-charts.vercel.app	This tool uses SPARQL and React to draw line charts of properties or queries.
PageViews Analysis, by MusikAnimal, Kaldari and Mforns (WMF).	pageviews.wmcloud.org	This is a suite of tools to visualize pageviews data of Wikimedia projects

Suggested Technology Stack

Based on the comparisons above Javascript is more used for creating interactive charts amongst the tools. In the same vein, Python seems to be better suited for the back-end. The Flask framework is a well maintained and lightweight option, as well as popular amongst the community.

Separating the front-end and back-end source codes into different repositories will improve the following ways:

- *Maintenance*, since both front-end and back-end will not be built in the same programming language;
- *Contribution*, since programmers with either React or Flask skills will be able to contribute to the part of the code they are most familiar with;
- *Testing*, since both front-end and back-end will be able to be tested independently.

Front-end

- **React.js:** For creating a highly interactive user interface.
    - **Why**: React is widely used, highly customizable, and has a strong community. It integrates well with data visualization libraries like D3.js.
- **D3.js:** For creating complex and interactive visualizations.
    - **Why:** D3.js offers fine-grained control over visualizations, making it perfect for creating custom infographic elements like bar chart races.
- **Highcharts:** Similar to D3.js but less popular.

Back-end

- **Flask (Python):** For handling API requests and processing data.
    - **Why:** Flask is lightweight, flexible, and easy to set up, making it suitable for small to medium-sized applications. It integrates well with Python libraries for data processing.

Data Fetching and Processing

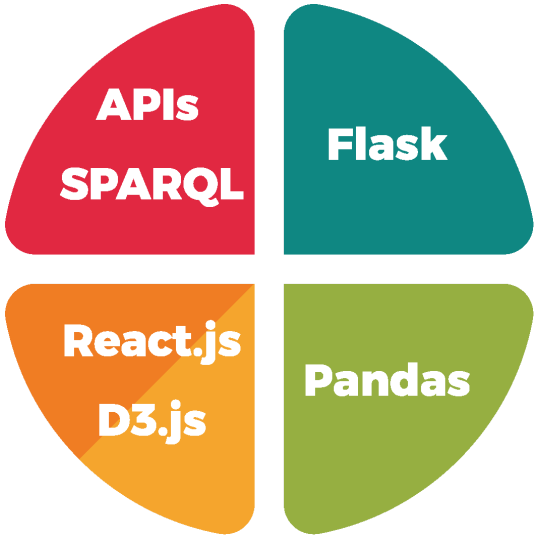
- **SPARQL**: For querying Wikidata.
    - **Why:** SPARQL is specifically designed to query RDF data, making it the best choice to interact with Wikidata.
- **Wikimedia REST API:** For querying structured data from Wikimedia projects.
    - **Why:** This is a solution embedded into the Wikimedia projects and it allows us to gather, among other metrics, user contributions, content growth and visualization metrics.
- **Pandas (Python)**: For data manipulation and preparation.
    - **Why:** Pandas provide powerful data manipulation capabilities, making it easy to clean and prepare data for visualization.

Database

- **MariaDB:** For database registers
    - **Why:** This is a solution already available for tools in Toolforge.

Conclusion

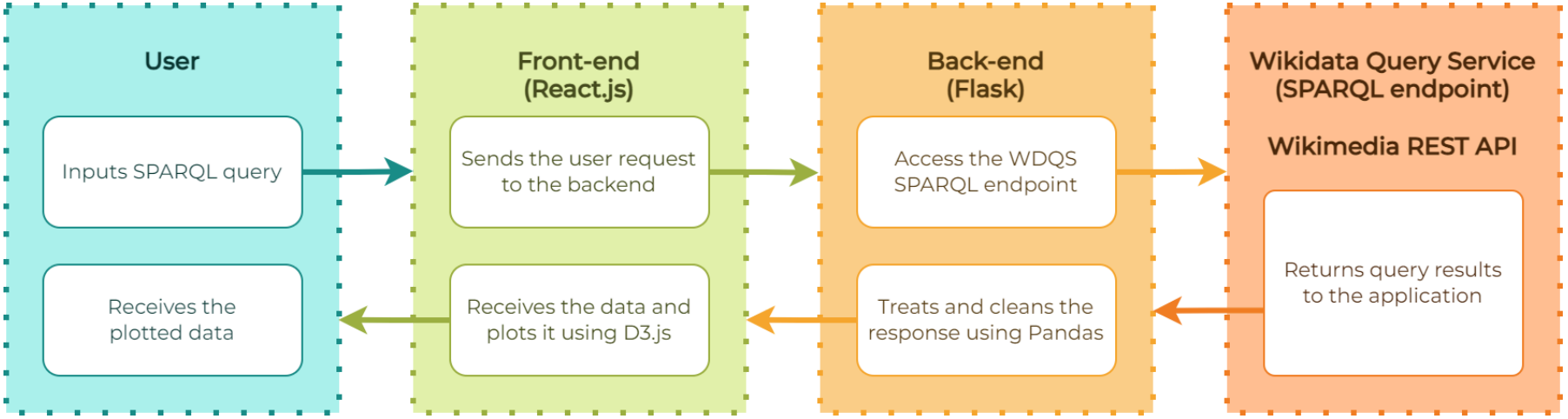
By selecting React.js and D3.js for the front-end, Flask for the back-end, Wikimedia APIs and SPARQL for data retrieval, and Pandas for data manipulation, we can create a **lightweight, adaptable, and robust tool for building infographics integrated with the Wikimedia projects**.



4. Architecture Design

The system architecture is designed to ensure modularity, scalability, and maintainability. The architecture follows a Client-Server architecture with a three tier pattern: Client (React.js), Server (Flask) and Data access (WDQS, Wikimedia REST API and Pandas). The data flow is designed to be efficient, ensuring that data is processed and transferred smoothly between components.

The user inputs SPARQL queries in the front-end, which sends the user request to the back-end. The back-end then accesses and posts the request query into Wikidata Query Service SPARQL endpoint, which in turn returns the results from Wikidata to the backend. These results are then processed and treated using Pandas, and then parsed back to the front-end interface so it can be plotted using D3.js.



5. Development Plan

Per Wiki Movimento Brasil's internal tech culture, the development approach for the Wiki Infographics project will follow an Agile methodology, with continuous feedback and collaboration with stakeholders throughout the project lifecycle. We will establish regular sprint cycles and incremental development and delivery of features. We will rely on GitHub for version control to facilitate collaboration among team members and help manage changes to the codebase. We adopt "mobile first" as a design methodology, as we focus on Global Majority digital practices first.

Agile methodology

The development of the Wiki Infographics Initiative will follow an Agile methodology to ensure flexibility and adaptability throughout the project. The project will be divided into multiple sprints, each focusing on delivering specific features and improvements.

Sprints

- **Sprint Duration:** Each sprint will last 2 weeks.
- **Sprint Objectives:** Each sprint will have clearly defined goals, such as developing specific features, integrating tools, or refining existing functionalities.

Scrum

- **Daily check-ups:** Short meetings or asynchronous updates to discuss progress, obstacles, and actions for the day.
- **Sprint Planning:** Meetings at the beginning of the sprint, focused on planning tasks and setting sprint goals.
- **Sprint Review:** Meetings or asynchronous updates at the end of each sprint to demonstrate completed work.
- **Sprint Retrospective:** Meetings to discuss what went well and what could be improved for the next sprint.

Mobile-first design

Considering that 70% of people from the Global Majority access the Internet using mobile devices, for a long time now it is our practice to develop tools with these devices in mind. We use both a design and a content-based approach, as we have to consider responsivity and Internet bandwidth required to use the tools we develop. Only necessary elements should be implemented in the design of the tool. In regards to development, we always prioritize solutions that are more efficient and have lower bandwidth costs for the end user. We prime the user experience in our developments.

Task Breakdown

- **Analysis**
  - Analyze structured data sources within Wikimedia projects.
  - Benchmark existing tools and technologies for infographic creation and Wikimedia projects integration, especially Wikidata.
- **Initial platform development**
  - Set up the development environment and define requirements.
  - Develop the core infrastructure for data extraction and processing.
- **Infographic template Design**
  - Determine the range of initial infographic templates to be offered and gather input on desired features.
  - Develop wireframes and mockups for the user interface.
- **Platform integration**
  - Integrate the infographic generation tools with Wikimedia's existing infrastructure.
  - Ensure seamless data flow and real-time updates.

- **Testing and feedback**
    - Conduct functional and usability testing.
    - Perform performance testing.
    - Implement automated testing scripts.
    - Solicit feedback from initial testers, selected based on the methodology to be defined and with diverse pooling from the Wikimedia community.
- **Final deployment and documentation**
    - Prepare the application for deployment to a production environment.
    - Conduct final testing in the production environment.
    - Document the installation, configuration, and usage of the Wiki Infographics MVP application.
    - Develop user guides and tutorials.

Version Control

All the code will be stored in Wiki Movimento Brasil's GitHub repository, available at https://github.com/WikiMovimentoBrasil/wiki_infographics. Pull requests will be analyzed by the Products and Technology team and/or other interested stakeholders.

6. Risk Management

Risk identification

Risk	Description	Consequences	Probability	Impact
Bad performance and low potential for scalability	Inability to handle large volumes of data and concurrent users while maintaining performance.	This can cause slow response times, system crashes, and inability to scale with increasing usage.	20%	High
Low user adoption and engagement	Not engaging enough users and/or not meeting user expectations.	Low user adoption rates, reduced user satisfaction, and limited software usage.	20%	High
Technical complexity	Complex integration with Wikimedia APIs and other data retrieval tools.	This can delay development and increase potential for technical failures.	20%	Low to Moderate
Dependency on non-perennial tools and protocols	Over reliance on tools and protocols that change constantly.	Disruptions due to API changes, protocols and tools updates without backward compatibility.	10%	Moderate
Bad designed user interface and poor user experience	Non intuitive and inaccessible interface for users to interact with generated visualizations.	This could lead to low adoption rates and user frustration.	10%	High

Risk mitigation

Risk	Mitigation strategy
Bad performance and low potential for scalability	• Perform load testing to identify performance bottlenecks early in development; • Implement asynchronous requests and study caching strategies for frequently accessed data and queries.
Low user adoption and engagement	• Conduct user research and usability testing throughout the development lifecycle; • Incorporate user feedback to enhance features and usability; • Implement outreach strategies to promote software benefits and encourage adoption.
Technical complexity	• Conduct thorough API feasibility studies and prototype testing early in the project; • Allocate extra time and resources for troubleshooting and debugging; • Engage with Wikimedia community experts for support.
Dependency on non-perennial tools and protocols	• Maintain flexible software architecture to accommodate API changes and tools updates; • Establish communication channels with the Wikimedia community for timely updates and compliance.
Bad designed user interface and poor user experience	• Conduct usability testing with representative users throughout the development process; • Incorporate user feedback into iterative design improvements; • Provide comprehensive user documentation and tutorials.

7. Documentation

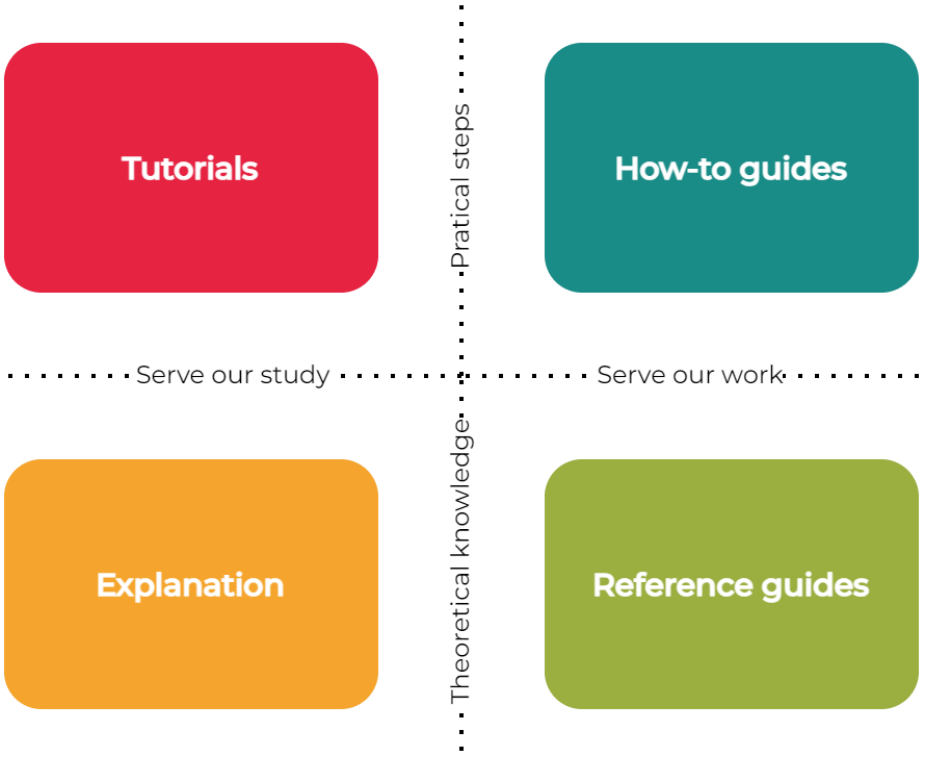
There are different types of documentation, with different purposes. For this project, we will be using a system that classifies 4 types of documentation: tutorials, how-to guides, reference guides and an explanation.

Tutorials

Tutorials are lessons with steps for one to complete a task. They are based on a beginner level of knowledge about the project and its dynamics and are learning-oriented. This can be done in several formats, such as videos, brochures and/or directly into the tool.

How-to guides

How-to guides are materials that answer specific questions one might have about the use of the software. They are based on an intermediate level of knowledge about the project and its dynamics and are goal-oriented. This also can be done in several formats, but we will prioritize formats that can be easily updated, such as a Wikitech page.

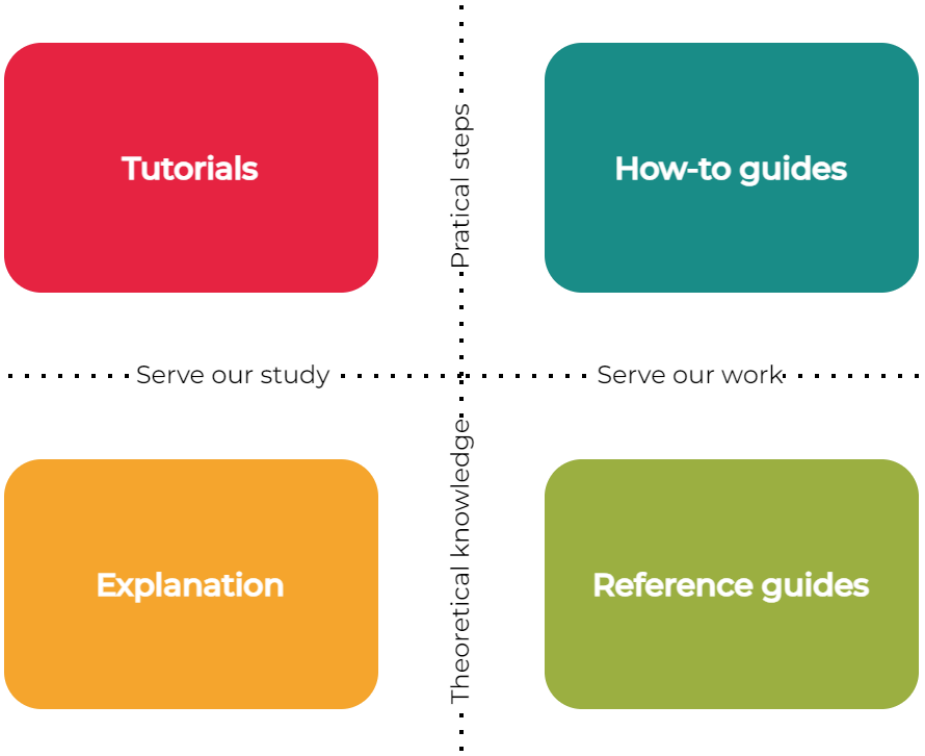


Reference guides

Reference guides are technical descriptions of the software and how one can install, contribute to and maintain. They are based on an experienced level of knowledge about the project or its infrastructure, and are information-oriented. This is commonly done through the repositories that store the source code of the software. We will use the Github Wiki structure and the *readme* file of the software.

Explanation

Explanations expand the context of certain topics related to the software. They enrich our knowledge about the problems we are trying to solve, the decisions we take and why we are taking them. This will be done in different ways: This technical development plan, case studies, and Diff blog posts.



8. Deployment Plan

Timeline

Milestone	Description	Jun	Jul	Aug	Sep	Oct	Nov
Benchmark tools and technologies	Begin benchmark tools and technologies for infographic creation and Wikimedia integration	■					
Technical plan	Write technical development plan	■					
Initial prototype	Develop the core infrastructure for data extraction and processing for Wikidata	■	■				
	Develop front-end interface		■				
	Build performance and other automated testing scripts		■				
	Create initial documentation (tutorials and reference guides)			■			
Feedback from initial testers	Gather feedback from initial testers, invited from the Wikimedia community			■	■		

Milestone	Description	Jun	Jul	Aug	Sep	Oct	Nov
Minimal Viable Product	Develop core infrastructure for data extraction and processing for other Wikimedia projects				■		
	Implement system for log and store queries				■		
	Build performance and other automated testing scripts				■		
	Improve and expand documentation (how-to guides, tutorials and reference guides)				■		
Deployment	Deploy the updated product on Toolforge					■	
	Launch the Wiki Infographics on Wikidata Lab					■	
	Finalize documentation (Explanation)					■	■
Maintenance and Support	Monitor usage of the software, collect user feedback and handle issues						■

9. Maintenance and Support

We will observe performance and usage metrics of the tool, gathering issues and feature requests from various sources, primarily GitHub tickets and Wikimedia discussion pages, and respond to them promptly. To ensure proper operation after the release of Wiki Infographics, we will establish a quarterly development cycle for major updates, while deploying minor fixes and updates as needed. Continuous monitoring and tuning of system performance and the user interface will be conducted to handle increased loads and improve user experience based on feedback.

By executing these strategies, Wiki Infographics will remain reliable, engaging, and user-friendly, ensuring long-term success and good user experience.

Expedient

Writing

Éder Porto

Products and Technology Manager, Wiki Movimento Brasil

Lucas Belo

Projects assistant, Wiki Movimento Brasil

James Okolie

Outreachy intern

Advisory

João Alexandre Peschanski

Executive Director, Wiki Movimento Brasil

Subcommittee (Re) Imagine the sociotechnical infrastructure of the Wikimedia Movement

Alberto Leoncio, Member

Augusto Resende, Member

Erika Guetti, Member

Geisa Santos, Member

Tiago Lubiana, Member

Veronica Stocco, Member