Front end architecture priorities, findings and recommendations

Purpose

The document aims to provide a technical direction for our platform in order to drive our products and technology forward to meet the goals of our Medium Term Plan.

It contains the findings of a collaborative review and analysis of the work and research performed under the Platform Evolution Program over the past 2 years. Its development was sponsored by the Core Platform Team and was created in collaboration with a consultant, Mentrix, which began in Spring 2019 and concluded in June 2019.

The key result of this analysis is the identification of several key opportunities that provide value to the WMF in supporting our goals, while at the same time requiring critical technical work and decisions that will enable us to modernize our platform.

As a next step, the opportunities in this document should be used as the basis for establishing a Front End Architecture Working Group to be sponsored by Technology and Product department leadership. This working group will evaluate the five specific challenges laid out below in order to generate solutions and well scoped projects to be developed by cross functional teams selected from both Product and Technology staff.

Table of contents

Background

Over the past two years, staff at the Wikimedia Foundation have been engaged in efforts to modernize our platform in order to achieve our Movement Strategy as well as address long standing issues facing our engineers. These efforts began with staff at the WMF and at WMDE discussing priorities in the Audiences Technology Working Group (ATWG) which formed in the Winter of 2017. These discussions led to the formation of both the Platform Evolution Program (PE) and the Core Platform Team (CPT). Starting in Summer 2018, members CPT engaged in consultations with staff, volunteers and community members to define our technical direction, culminating with the first Wikimedia Technical Conference in Fall 2018. From there the CPT was able to turn the work and feedback generated from all these stakeholders into goals for the PE Program and input for the WMF Medium Term Plan and the Annual Plan for FY1920.

While it has been a long road, the result has been the comprehensive documentation of our technical goals and priorities based on collaborative process involving a cross-section of stakeholders both within and outside the WMF. The reason we have put so much effort into creating these goals is so that we can now prioritize projects and make technical decisions based on this shared understanding of our technical priorities.

The CPT has already begun the first phase of work based on these priorities, including building a REST API in MediaWiki, Splitting RESTBase into separate components and beginning to librarize MediaWiki into components. Other teams at the WMF have done the same… The Product Department is leading the effort of unifying our parsers in order to simplify our technology stack.

This document signifies the beginning of the next phase of work stemming from the PE program and our shared priorities - work that truly drives our technology stack forward to meet our Medium Term Plan goals and in order to achieve our organizational mission.

Scope

The scope of this document focuses on changes geared towards our user interface (“front end”), the technical discussion and transformations that the recommendations will require us to push forward our capabilities in both structured data and modularity. These two critical capabilities enable better knowledge equity, mobile support, modern user experiences and improvements to multimedia. This focus was chosen specifically because it provides visible and immediate user value through modernizing our user experience (one of our Medium-Term Goals). However, structured data and modularity also support our initiatives around machine learning, meaning that we further our medium-term goals in this critical area as well simply by making the architecture changes to improve our front end.

What this document is “not”

This document is focused on providing challenges to be solved that ultimately improve our architecture in order to support our long term goals. However, it does not:

- Provide solutions to those challenges
- Discuss “illities” associated with any solutions, such as scalability, security, maintainability, etc…
- Address constraints such as resourcing, time or budget.
- Community relations

While these considerations are important, they are not discussed in this document because they fall within the scope of the proposed working group which will generate solutions and must consider these and other potential impacts on our infrastructure, organization, community and users.

Introduction

Priorities

The Wikimedia Foundation's medium-term plan includes interdependent priorities whose success depends on modernizing the current system(s) architecture. Priorities include:

- Delightful UI experiences across all devices
- Integration of content from multiple projects (and other initiatives) to deliver new, dynamic knowledge experiences
- Integration and discoverability of rich content including video, audio, and interactive media, as well as the infrastructure to serve it with high performance, high redundancy, and low latency to all parts of the world

Activities

The goal of this elaboration was to understand the current architectural impediments and collectively move towards those priorities. This document highlights architectural patterns that represent core challenges and opportunities. Activities we engaged in included:

1. Interviewing the internal engineering team
2. Reviewing previous documentation and discussions
3. Modeling of current business process flows
4. Discussing, debating and describing the findings
5. Recommending opportunities for growth

Questions

During the exploration, we had questions in mind like:

- How does content flow through the system now and where is it entangled? Where are the sticking points when considering system-level change?
- How do we want to build user interfaces and how is that different from what we are doing now?
- How do we make it easier for engineers to scale, maintain and test Wikimedia software projects and products?
- What changes enable us to deliver more features and adjustments in less time?

This document does not try to answer these questions in isolation but instead, establishes a foundation that supports the upcoming Working Group. They will dive deeper into the highest-priority patterns.

Summary of challenges

Many of the challenges the WMF faces when modernizing their system(s) are shared across the industry. Three are relatively unique and the most essential challenges for the Working Group to work through:

A “Front End" must be defined

Although we do and will have front ends on many clients apps, voice assistants and otherwise, it is useful to think of the front end from the browser perspective as it helps us define what we need to make these different experiences available to all.

In the browser context, the front end can be described as:

> *Anything the browser touches. What a user sees, clicks or taps as well as what may be running invisibly within the browser.*

This definition of a “front end” can be easily extended to our other experiences by generalizing it to:

> ***Anything the client touches. What a user sees, clicks or taps as well as what may be running invisibly within the client.***

This gives us a good starting place. We can further delineate our front ends by the experience of the user. In this light, there are five different front ends that Wikimedia project user experience:

- **Reading**: A highly-cached, quickly-loaded displayer of pages front end. This category could be broken down into desktop, phone and app reader front ends because they are not the same.
- **Editing**: A composition front end that provides tools like Visual Editor, Wikitext editor and Content Translation
- **Curating**: A workflow-driven front end that uses a suite of tools to interact with content and content events. Currently implemented in various WMF and community developed tools such as: Watchlists, bots, gadgets, special pages, etc…
- **Discussing:** A conversational front end providing users a means to interact with other readers, editors, and curators. Currently implemented with Talk Pages and Flow.
- **Administrating:** A set of tools that allows administrators to setup and run a wiki. This is mostly implemented in scripts, command line utilities and some special pages.

While these front ends sometimes overlap and conflated, true power comes from establishing clear boundaries between each. With such encapsulation, we can create more powerful interfaces by composing them in different ways (Such as overlaying Discussing on top of Reading).

In addition to how you use it, the front end is also defined by how you build it. A front end can be defined by the workflow of front end developers, though that flow differs depending on whether you are an on-wiki, tool or platform developer. Most importantly to our discussion, there is an emerging definition of front end that defines front end as “products built without building a new backend.” This type of front end depends on a modern software and/or systems architecture.

Building modern software tools that improve one of those experiences complicates others. A challenge for the Working Group will be to balance the needs of these “front ends”, identify the highest-value modernization opportunities and outline the tradeoffs.

Systems of software must be intentionally designed

The Wikimedia Foundation's projects are systems of software. Integrating multi-project content and building new product UI experiences depend on communication between parts of the system(s). The primary impediment to this communication is that many of these “parts” are entangled within the mechanics of the software itself. These entangled mechanics are both necessary and constraining.

To move forward, the Working Group will explore opportunities for modularization, disentangling software from system capabilities. The goal is to map a progressive evolution towards better communication between projects and within the system itself. What roadmap for change delivers software parts that interact more easily with each other? What bridges between them need to be built?

Software patterns don't (always) scale. Wikitext, for example, works well for the software but does not structure data for healthy system-level communication. Also, software activities are designed to react when pages change, whereas system activities can happen anytime for many reasons, including when embedded content or data from another page changes. Communication *between* software components, in their own time, is critical. The Working Group should consider ways to improve system communication while respecting the integrity of the software.

Essential note: Most individual MediaWiki software instances (3rd party use cases) do not share these challenges. The Working Group will need to define the tradeoffs between software simplicity serving those use cases and engineering modern architectural enhancements that can improve Wikimedia's systems and products in order to enable leadership to make the best possible decisions in these cases.

Page content must be modularized and compose-able

The priorities, challenges and modernizations discussed here depend on one thing above all else: the ability to “build” or compose a page of content from blocks of various content.

At present a rendered page is a blend of content, sometimes from different sources, but the current software architecture displays a fully-rendered page originating (wholly) from Wikitext. There are not component parts to break up a “page” into smaller parts that can be shared. Practically this means the full content of our articles, whether it be text, media or data are “trapped” inside Wikitext. When rendering a page on a desktop this isn’t problematic.

However, when features or products require modularity, semantic meaning or structured data, they become much harder to develop, more prone to errors or even worse, impossible to create. This is in part due to Wikitext not being easily interpreted by machine learning apps or content layout tools, and these use cases often require time intensive brute force techniques and unscalable heuristics to overcome. This limitation has impacted the development of modern UI features such as responsive layout, creating stackable content such as page summaries, and structured data use cases such as Structured Data on Commons.

Wikitext has been critical to the proliferation of Wikimedia content by empowering hundreds of thousands of editors to publish millions of high quality articles on Wikipedia and other projects. Wikitext as a markup language has many upsides, but its use to manage layout and data have come with tradeoffs that have much deeper implications in 2019 than they did in 2009. To be successful with our recommendations, we need to always keep in mind our users and the need to provide them great authoring tools as we make architectural changes in pursuit of the goals of the Medium Term Plan.

This is an essential area of focus. Recommendations for the Working Group include multiple ways to experiment with decomposing a page without starting with a “Major Overhaul of Everything”.

Recommendations

Desired qualities

The goal of these recommendations is to develop three essential qualities in the system:

Modularity

- There is some separation of concerns and activities.
- Boundaries between software and bridges between software are defined. (See Systems of software above.)
- Changes can be made to one area without impacting other areas.

- Data shared between parts can be decomposed, pulled apart, and composed, put back together based on the context. The process scales, new combinations can be made without writing more software.
- Enables things like libraries, stores, encapsulation, services etc

Just-enough coupling

- Currently, the architecture is highly coupled, which is another way of saying entangled. Decoupling is not an event but a process, a series of steps that change the overall pattern.

Timing: reduce temporal coupling

- Create less procedural, sequential control of relationships and communication.
- Page generation is a nondeterministic activity by nature. Communication generally requires more ability to predetermine responses.

Exploration Areas

Our elaboration effort exposed five areas of potential exploration. We recommend these areas of focus for the Working Group. They satisfy valuable priorities, test the waters for further disentanglement, and/or raise issues that need to be resolved, regardless of the path chosen.

1. Develop <FEATURE X> using modern front end tooling

This effort is focused on two important goals for product development: The ability to build delightful, dynamic user interfaces - and providing an architecture which enables front end engineers to quickly develop new user interfaces which are decoupled from the platform and do not require knowledge of how the platform works internally.

An essential discussion point for both “defining the front end” and modularization is how to untangle Javascript and bring it into the front end. How do we enable Javascript developers to keep the code as close to the browser as possible and reuse common functions? OOUI enables the integration of MediaWiki with Javascript enhancements but because it was developed prior to the current set of industry frameworks, it conceptually differs in fundamental ways from those tools.

These architectural differences mean that MediaWiki is not easily integratable with industry standard frameworks and also make it difficult to share the localization and accessibility advances that have gone into OOUI with those engineers attempting to use frameworks like Vue and React. Figuring out how to align with industry standards while integrating the work that has gone into OOUI represents a huge opportunity for Wikimedia to contribute to localization and accessibility of modern JS development for the web.

This area of focus lays the groundwork for moving further towards “front end”. The Working Group will need to explore how the code contribution process might work, among other factors, which will engage Javascript developers in the architecture process.

2. Create a service for rendering components from template content

Templates in the MediaWiki ecosystem are overloaded with different use cases: they provide content structure, reusable components, and allow contributors to modify the display of content with custom logic. This recommendation has strong potential for creating modularization that benefits the system as a whole. It is tricky to discuss though, because "templates" mean different things to different people and have different use cases.

In this case, we don't mean Mustache templates that govern look and feel. We mean encapsulating the "boxes" of content that are added to pages. These may (or may not) take parameters defining some of their content. A first step for the Working Group will be to define what is included in this "template" modularization effort and what isn't.

These “boxes” are an ideal focus area for creating modularity. They represent self contained features and also an opportunity to enable equitable sharing of user features across projects and languages be establishing a cross-project service to share templates. This project will also force us to consider how to handle content layout and structure separately from composable pieces content.

The patterns architected during this initiative will inform other types of modularization (libraries for front end enhancements, for example).

3. Build a non-page based watchlist using APIs

The Watchlist page does not rely on content pages (or need to break them down) but instead, reports on changes to pages. Curators are interacting with that page in ways that are ideal for prototyping a "front end" tool that communicates with the system but does not need to be enmeshed into it. The stakeholder group is smaller and eager to provide feedback.

This area of focus lays the groundwork for architectural patterns that can apply to special pages and gadgets in general.

4. Introduce page summaries as distinct, editable and curate-able content

In order to lay the groundwork for structured data in articles, we should develop a feature that:

- Will use all the multiple types of data storage, such as text and images.
- Are composed from different sources, such as Wikipedia and Wikidata
- For which we can easily generate and populate content
- Have an immediate benefit for our product plans

Page summaries are designed to be an easily distributable “block” of content which can represent the full content of a page and can be displayed in different contexts. They are currently generated via a hardened, well tested service. The content from summaries can be used to create **Open Graph** markup to make our content machine readable to power machine learning applications.

Moreover, page summaries are currently only generated heuristically and are not able to be defined or modified by editors. Creating a modular architecture to store and curate summaries establishes a pattern that can apply to other elements of the page.

Summaries also begin the conversation with the community about the types of modernization inherent in the medium-term plan. These changes will eventually require socio-technical process for change management as they introduce new types of curatable content. This recommendation gives the Working Group an opportunity to consider and architect people processes that enable their technology strategies. And include staff who are especially skilled at this aspect.

5. Introduce topic maps to ontologically categorize content

Categories have been used by contributors in order to create a taxonomy of pages for years. They use them as a means to tag content for discovery or curation or other patrolling activities. This has been accomplished by adding Wikitext markup causing semantic meaning has been embedded within the content. This also means that categories have no inherent structure or hierarchy and are not able to be used across projects.

Topic maps present a new way to separate the taxonomy of the content from the content itself providing structure to our wealth of content and enabling new means of discovery. With this first project we could establish a means to tag to pages and other types of content with editable, curatable, standardized metadata as well as machine generated metadata.

Next steps

Forming a Front End Architecture Working Group

The recommendations above are intended to drive the planning of the architecture changes needed to modernize our platform and serve the known and unknown use cases of the WIkimedia ecosystem for the next decade. As a next step, it is recommended that a Front End Architecture Working Group is formed to further elaborate and scope each of the five explorations. Thoughtful composition, processes and facilitation will be critical to the Working group’s success. These will be developed and defined in a document in order to provide structure to the work and help guide the group towards impactful outcomes.

Generating project proposals

The first goal of the working group should be to generate a proposal to solve one of the five explorations with the output being a well scoped project plan. The proposal should have full project plan with product and technical requirements, as well as resources, milestones, risks and tradeoffs. Aspects of performance, scalability and security should be accounted for in solutions. Metrics should be established to measure success. Projects should be focused on delivering a user facing feature and also be as short as possible (no longer than 3 months) to ensure they can be accomplished. The exact format and content of a proposal will be specified in another document.

Developing and delivering a project

For each proposal, a cross functional team from the staff of the Product and Technology Departments should be created. Team members should be chosen for not only their skills, but also their enthusiasm to exploring new ideas, eagerness to push the boundaries of our technology and focus on delivering value to our end users and community. These teams will focus on delivering the solution proposed by the working group and will work exclusively on the project for the duration. The result of the project should be a shippable user facing feature or product. This focus on delivery is important as it keeps the team driving towards real needs and real users.

Retro and repeat

This is a living document and the recommendations within it represent our best current understanding. As we build, we learn and this document should be updated to reflect those changes in understanding. Following the delivery of a project, both the working group and leadership should perform retrospectives on the process. Improvements and changes should be suggested. New projects should be proposed, existing projects should re-evaluated and the next exploration should be chosen. At this point new working group members can rotate. This template should be followed until all five explorations have been researched and solutions have been developed.

Success

The success of this initiative is measured by the change in our architecture, products and processes. We should see real changes in engineer productivity (which we will likely need a way to quantify), we should see real value delivered to users as measured by the metrics we develop, and we should establish new ways for our engineering teams to work together by experimenting with cross functional teams.

Appendix

Further Details

This section includes further discussion of areas mentioned above and a summary of patterns identified during interviews.

Challenges shared by the team (and the world)

We asked everyone about their day-to-day challenges. How do you continue to deliver emerging, interdependent products that serve users all over the world, on a multitude of devices, without adding undue complexity? Interviews with the team highlighted the challenges and informed the recommendations mentioned above, Javascript being the most often mentioned issue.

Other challenges are not unique to Wikimedia; they are increasingly commonplace. Open-source communities, including Drupal and Wordpress, are grappling with similar roadmaps. The core design of web applications are changing. Originally created to deliver web “pages”, content software must now interact with other platforms and data structures. Activities within the system are designed to happen at “page load” time, the emerging need to interact during other types of events is daunting. Control over styling and page structure, needed to build products, isn't easily separated because the content itself is not structured for sharing.

Further challenges include:

- **Sociotechnical integration**: The organizational boundaries designed to define “ownership” of technology parts aren’t mirrored in the system itself. There is more enmeshment in the technology itself than there is between people building it. These communication challenges increase and as complexity increases. Historical attempts at change has eroded trust in decision making. (<-- This is the most common organizational challenge.)
- **Performance variations**: Architectures that delivers fast results for anonymous users struggle to make changes that deliver fast results to logged in users. The reliance on caching is both a blessing and a blocker.
- **Reliability tradeoffs**: Changes to the software system increasingly rely on “workarounds”. These nonconventions inadvertently create more sprawl and less predictability. Modularization will alleviate this but introduce new debugging challenges.
- **Core technologies are changing**: What is the future role of Javascript (especially for the Wikimedia community)? As Javascript frameworks and microservices proliferate, the adoption process is still shaky and risky for many use cases.
- **Development times can be unpredictable and long:** The amount of effort to ship a

feature is unpredictable and delays in the process are frequent due to both architecture and process issues
- **Editors have a deep attachment to their workflow**: And want control over presentation. They are valuable to the mission and change averse.
- **Transformations are people intensive**: There are not enough people to do the work and there is a constantly-expanding list of new skillsets required. They also require a tremendous amount of proactive communication.
- **Systems are emerging**: Monolithic software architectures are being replaced by multi-platform, systems architecture. The tradeoffs aren't universal though, they are unique to each situation. As complexity shifts from the software to the system, a whole new set of architectural challenges arise.

Emerging definition of “front end"

We do not want to define what a front end *should* look like, the right blend will evolve as the priorities are defined. Industry-wide, there is a conceptual definition of front end emerging, as software becomes decoupled parts. It includes:

- **Owns the display**: look and feel standards are defined by the front end.
- **Componentized**: builds page “blocks” of its own choosing, can vary what to show or hide. Thus, it relies on composition and (to some extent) on decomposition in the backend.
- **Multisource**: can asynchronously fill component “blocks” from various content sources.
- **Emergent**: does not stand alone as a single piece of software but represents a sum of relationships with collective value. The sum is greater than the parts.
- **Guides the user journey**: the front end is the map followed by users to their goals or destinations.
- **Provides interaction**: being closest to the browser, it structures user interaction and its relationships to other software supports this.

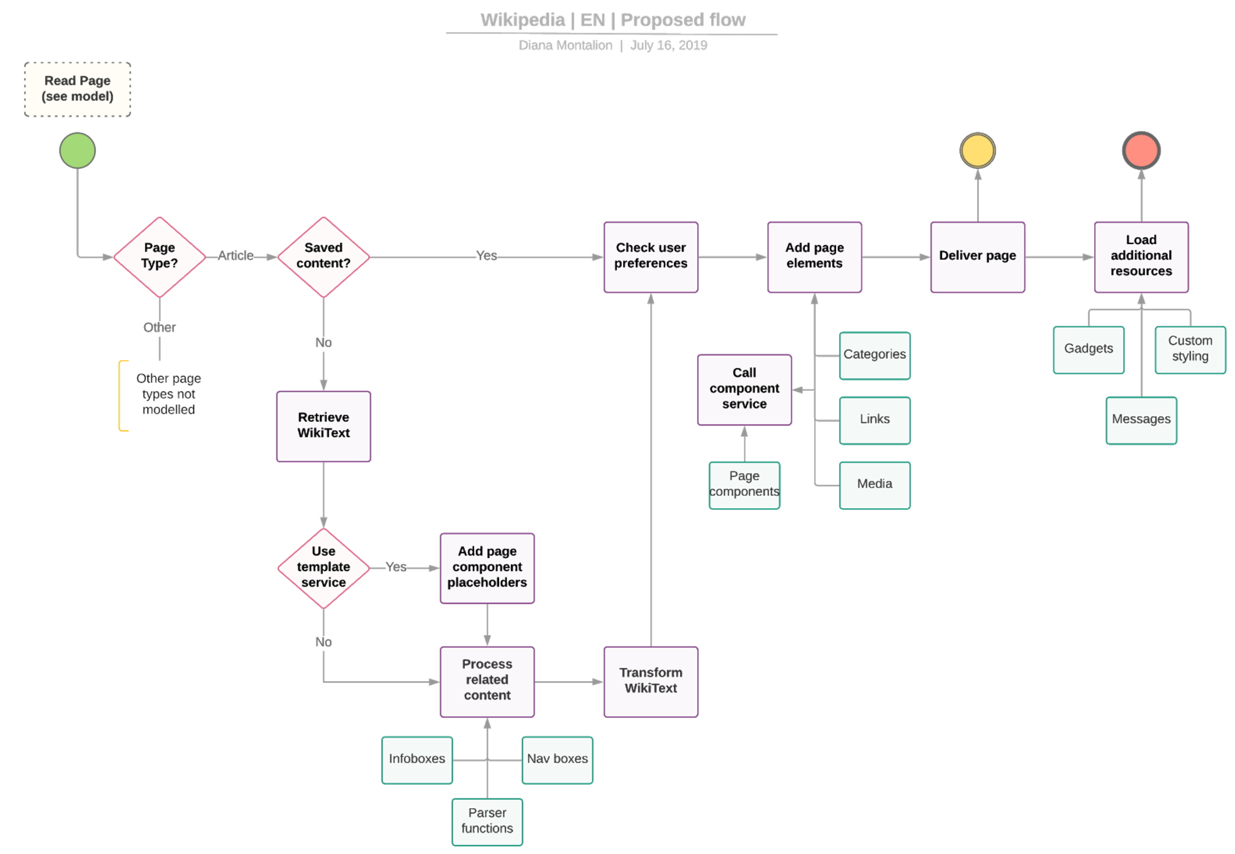
Benefits of focusing on abstracting the functionality of page components and content layout from templates

We've recommended focusing on the “boxes” of content that are added to the body of pages as a stepping stone to modularization. Here are some reasons why:

- Creates a mechanism to make templates equitable, sharable and localizable across wikis.
- Structures the right discussions - the architectural design exercise is valuable because

it's inherently modular. The team as a whole will be thinking in the right direction.
- Because they are valuable to most projects, the work can't be architected in a silo. The working group's work will encourage sociotechnical integration.
- Improves the architecture for all users (eventually) without impacting anyone initially.
- Users can continue to use the software as it is, adoption is optional.
- The stakeholders are primarily the "core" team.
- Establishes patterns that can be (potentially) used for further decoupling. Gadgets? Libraries?
- Cleans up "core" content.
- Cache purging can become more efficient and discreet.
- This creates the potential to move -some- composition up the stack, which is a linchpin for front end changes.

Here's a look at how it *might* work ...



[lucidchart link]

Process models

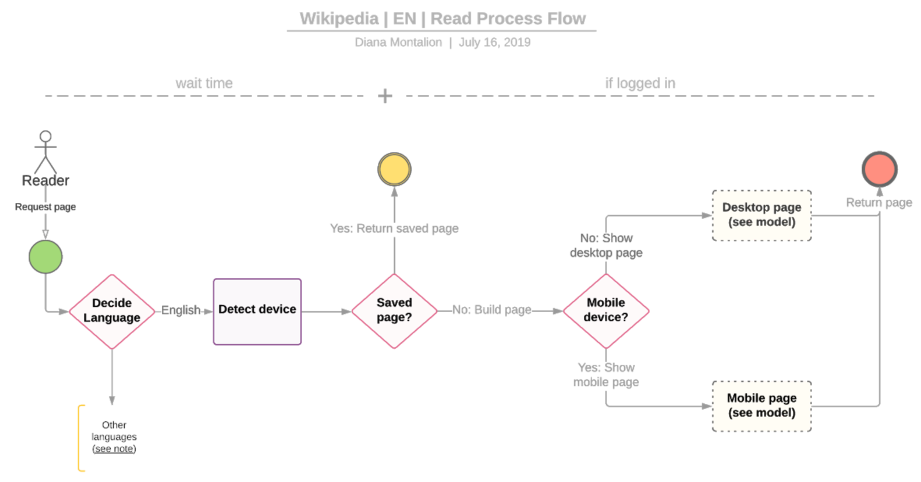
The following models show, at a high-level, how critical activities trigger technological decisions and interactions. There are many more process models that might be valuable to the Working Group. Here, we hope to establish core activities and a foundational practice for modeling that -- above all - enables cross-functional discussion about changes. We've focused specifically on Wikipedia for illustrative purposes.

Predominantly, we are interested in differences. Reading a page on a phone is different from a desktop. Reading an English page is different from a German page, which is significantly different from a Chinese page, from the system's point of view. The goal is to call out these differences in ubiquitous language because wherever the flow of business logic branches, or relies on different workflows, is where the challenge to change lives.

The models shown below are static versions. Click the lucidchart links for a bigger, clickable versions.

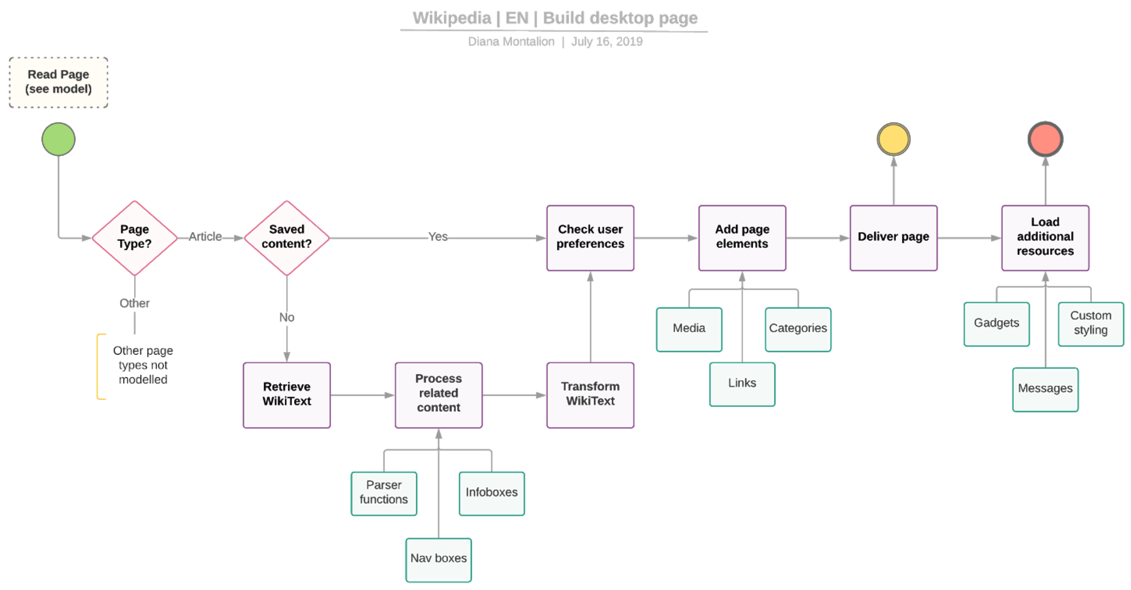
Read page

Read an English Wikipedia page



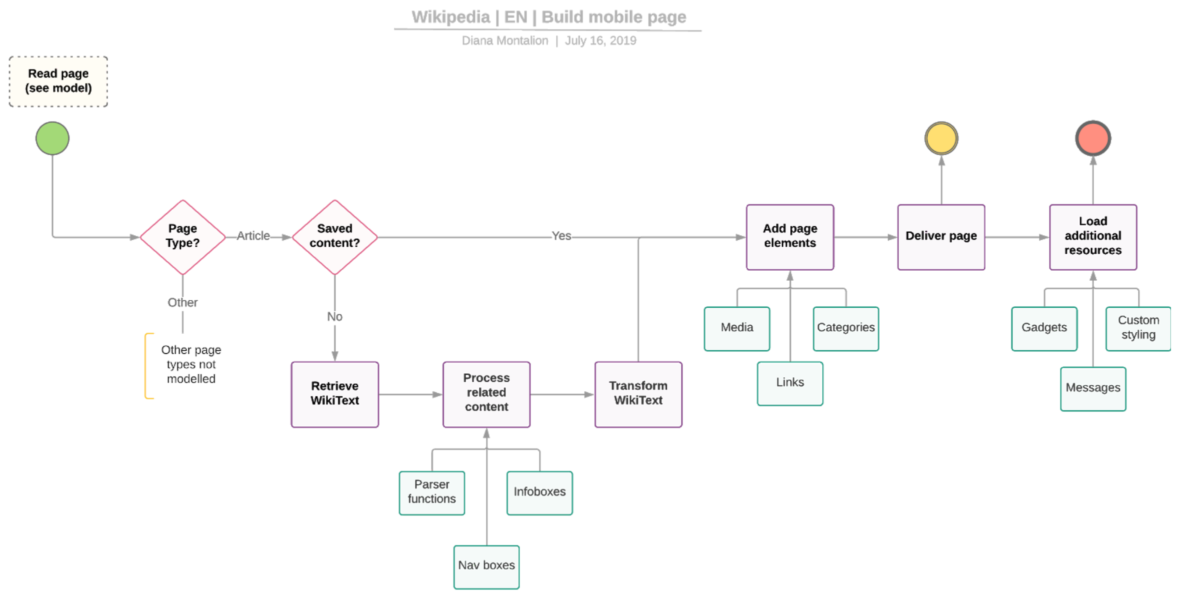
lucidchart link

Read an English Wikipedia page on desktop



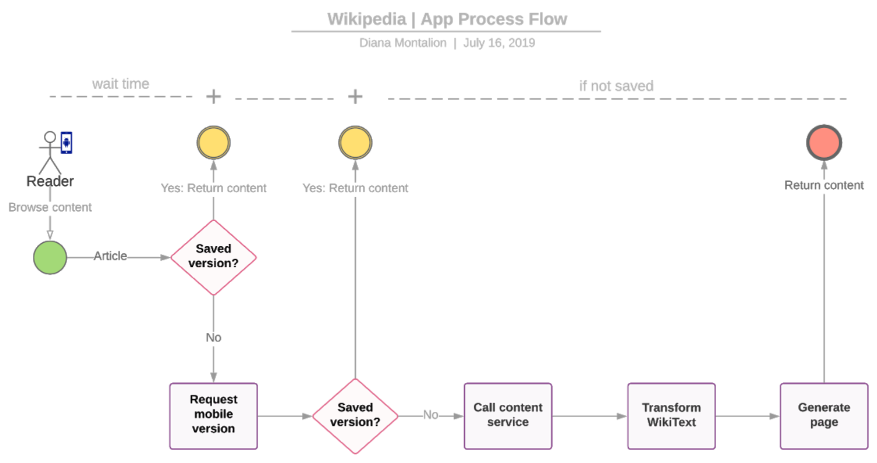
lucidchart link

Read an English Wikipedia page on mobile



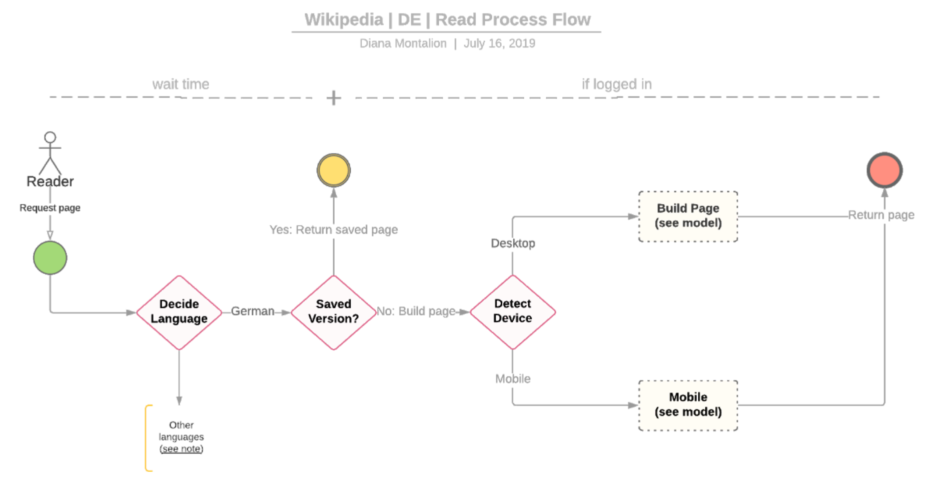
Lucidchart link

Read an English Wikipedia page in the app (android)



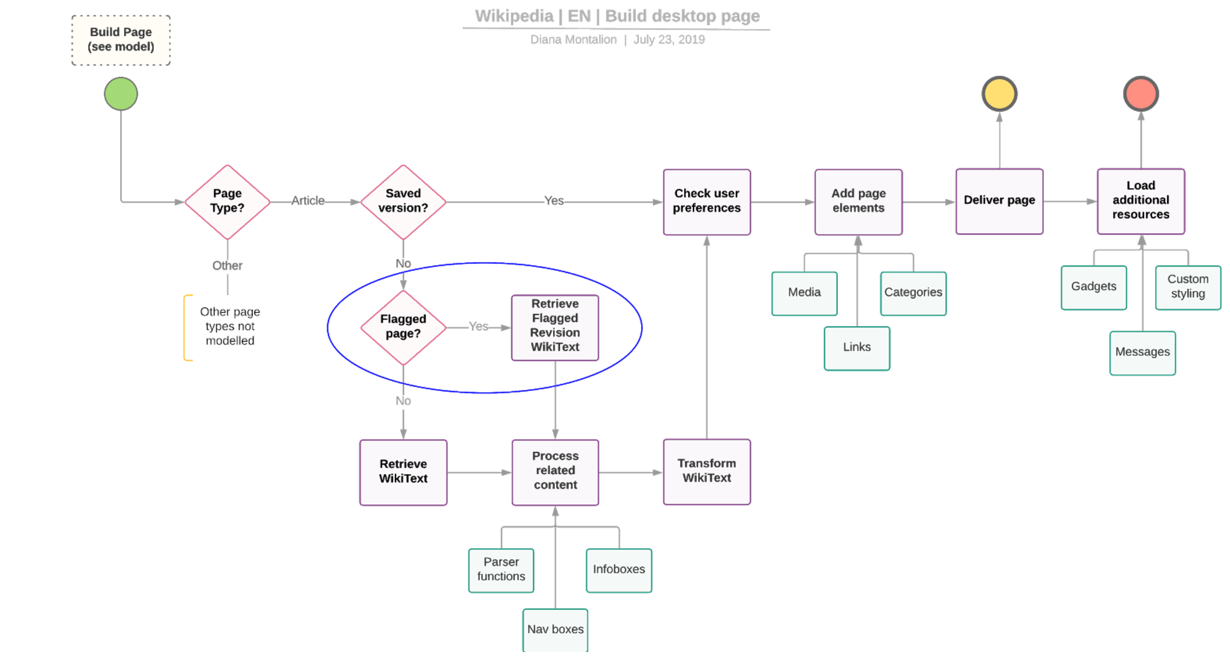
lucidchart link

Other languages: Reading a German Wikipedia page

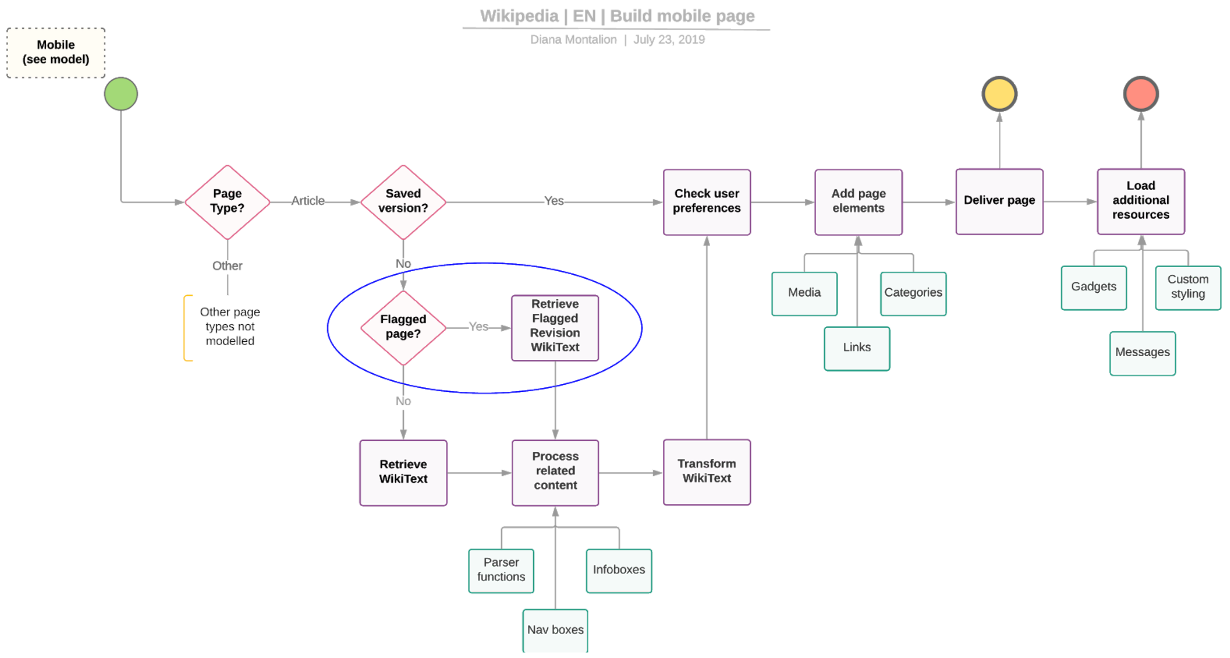


lucidchart link

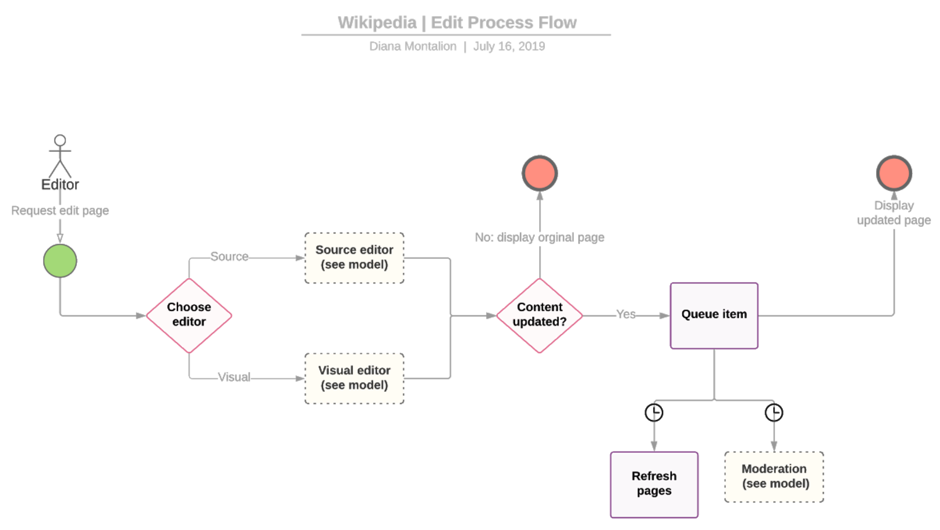
Read German desktop page



Read German mobile page

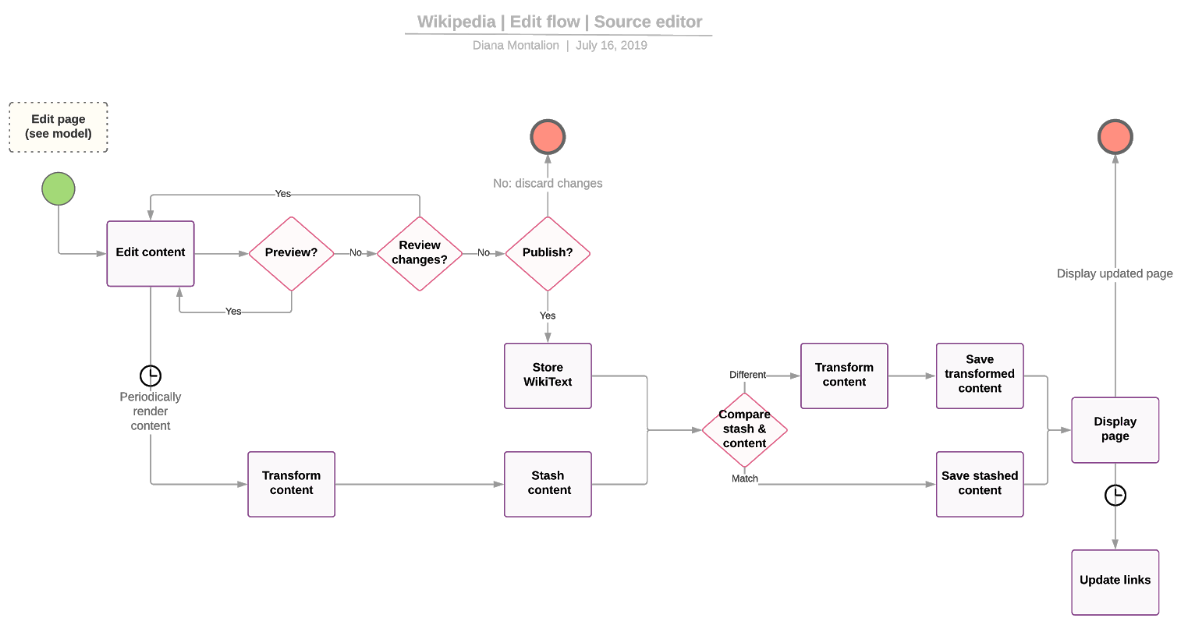


Edit page



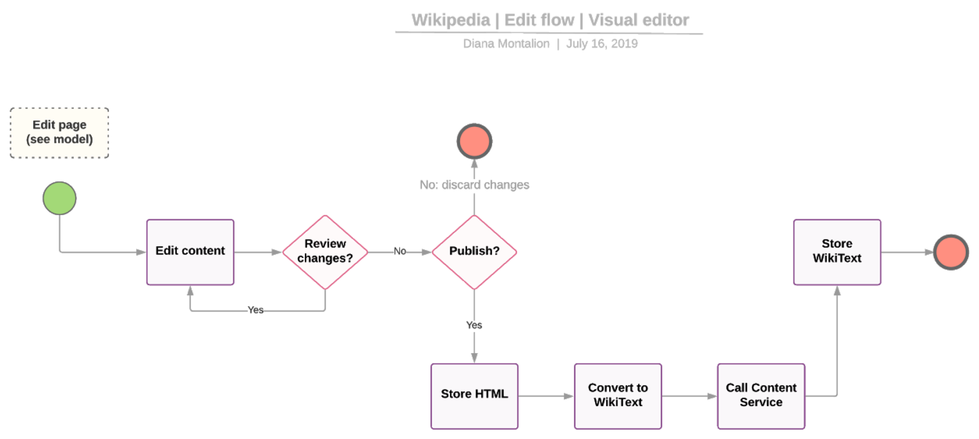
lucidchart link

Edit an English Wikipedia page using the edit tab



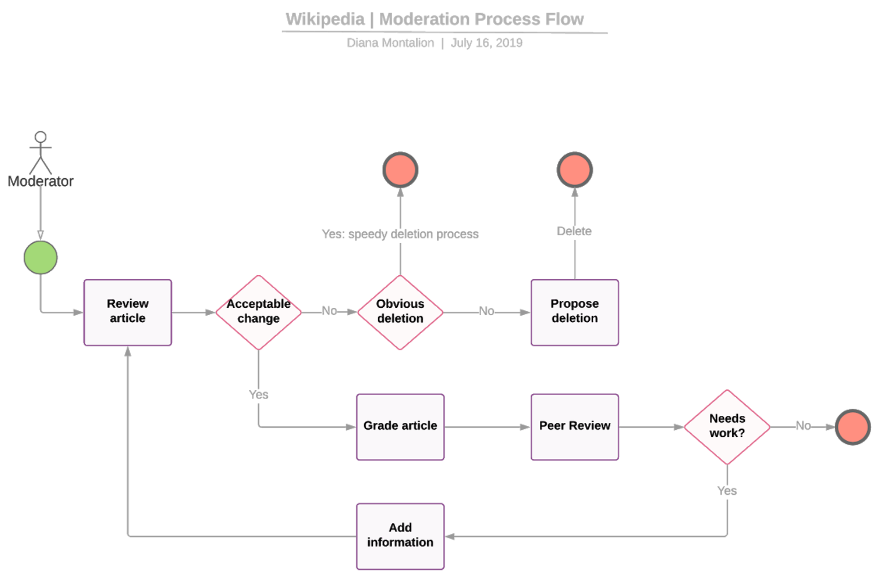
lucidchart link

Edit an English Wikipedia page using Visual Editor



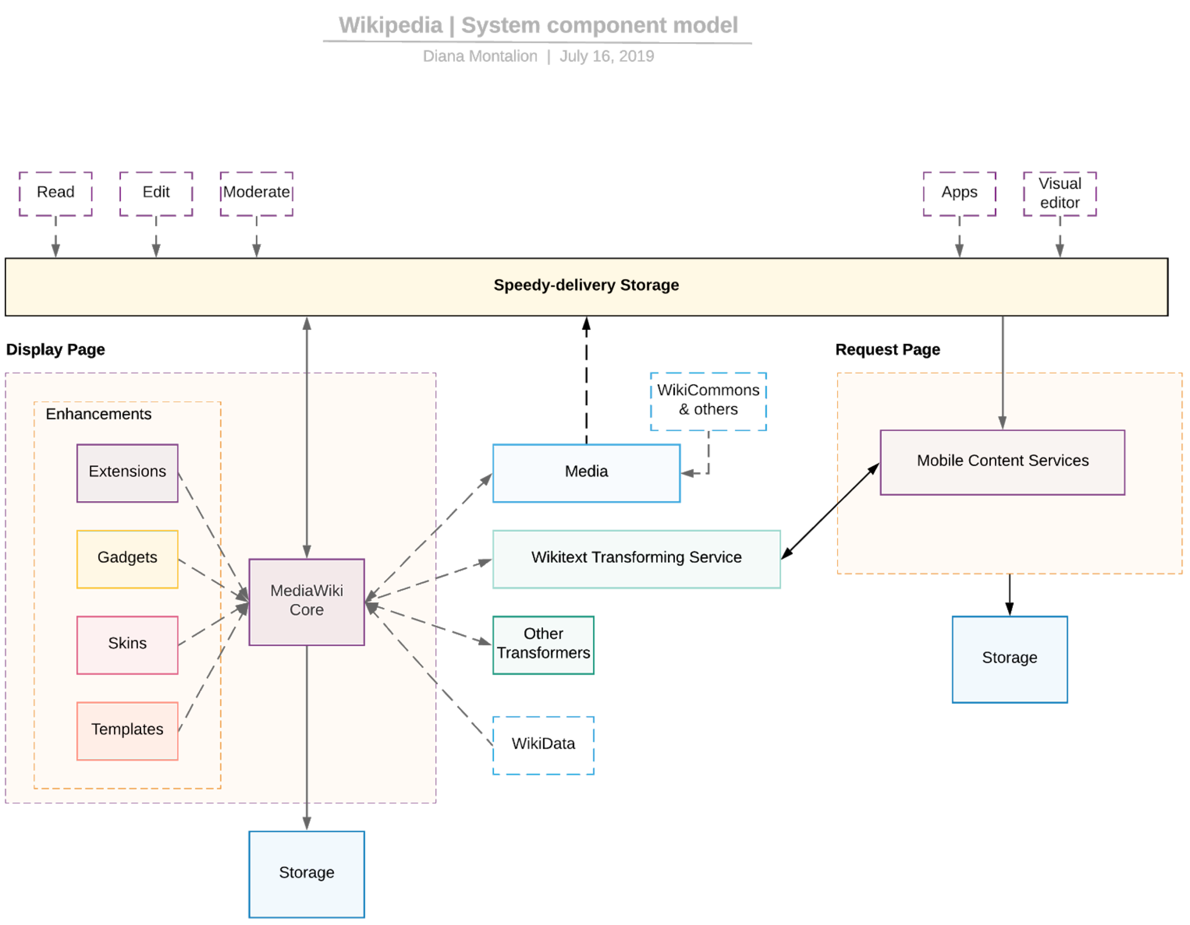
lucidchart link

Moderate an English Wikipedia page



Lucidchart link

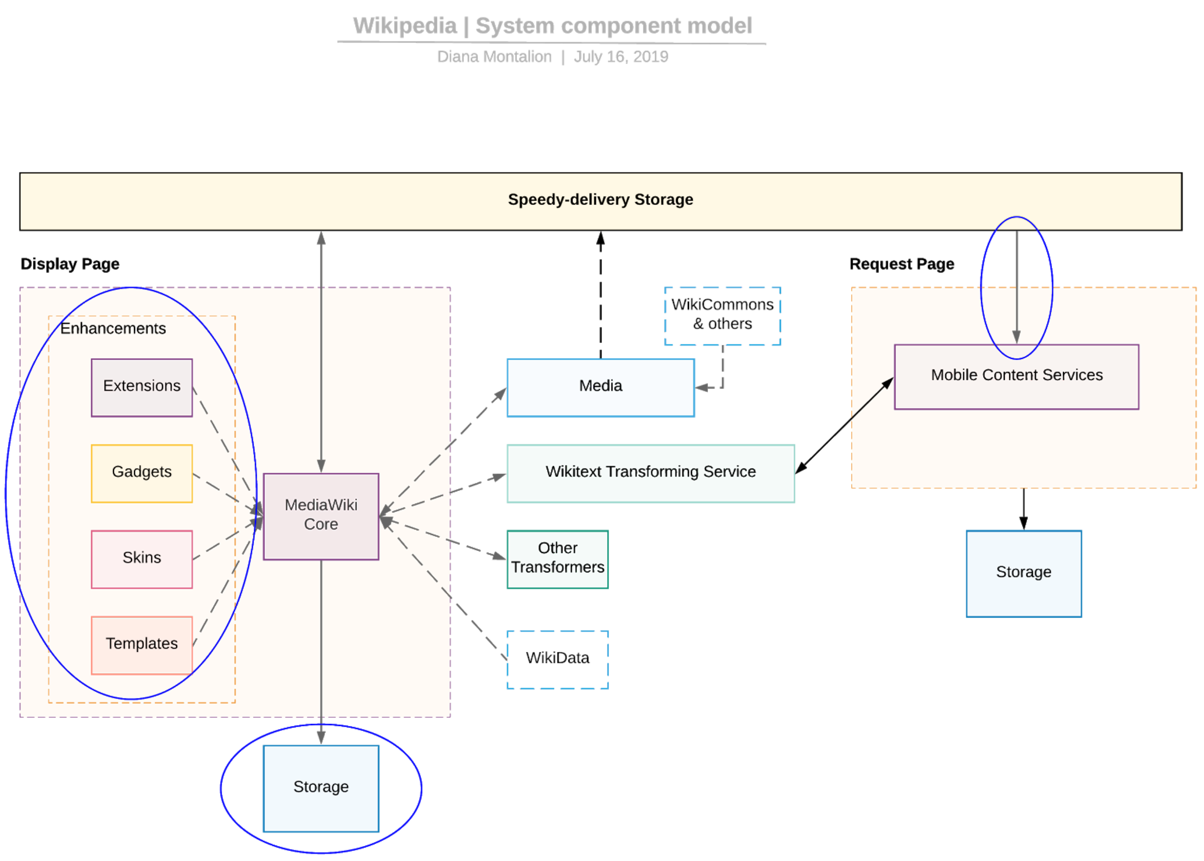
How system components interact to create pages (all cases)



Lucidchart link

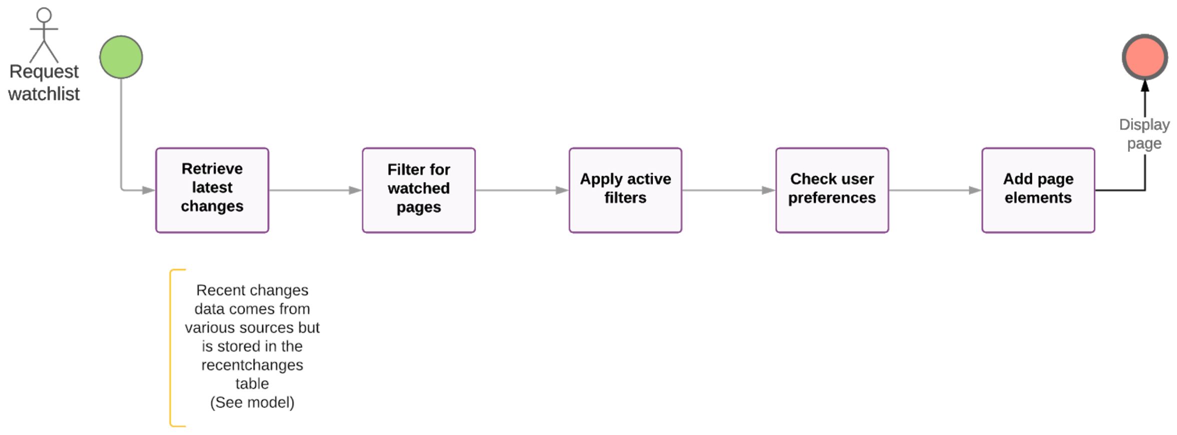
Different language wikis

Each language has its own Wiki, in that all the configurations, content and adaptations are unique to it. The differences between English and German are highlighted. See the differences in components here.



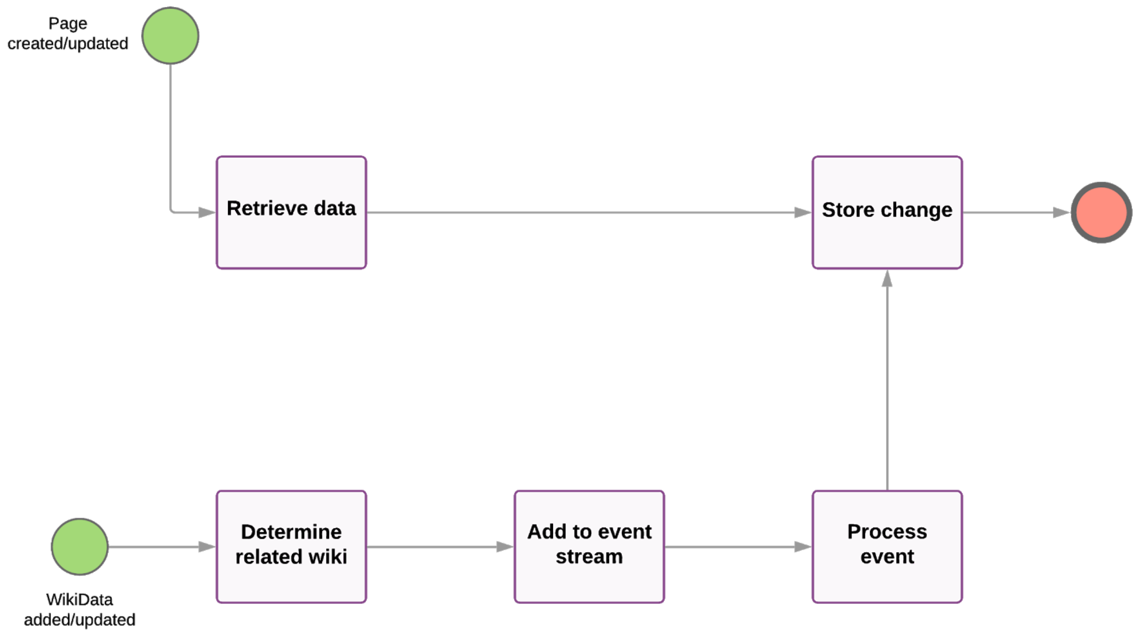
Non-wikitext pages

Watchlist page



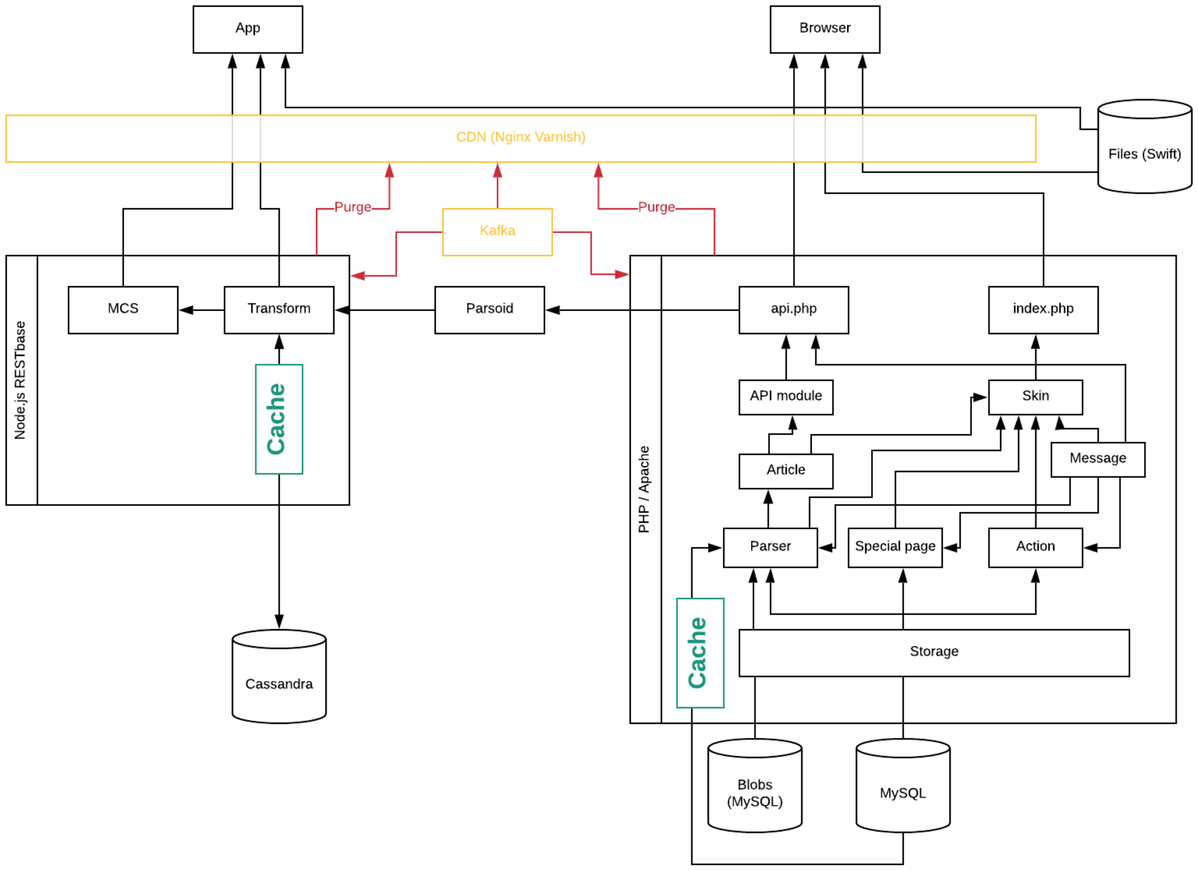
Lucidchart link

Recent changes data



Lucidchart link

Page rendering process (aka Daniel's whiteboard)



lucidchart link