

República Bolivariana de Venezuela  
Universidad Central de Venezuela  
Facultad de Ciencias  
Escuela de Computación



**Desarrollo de un editor de visualizaciones de  
propiedades de historiales de wikis**  
**Universidad Central de Venezuela**

Tutor Prof. Eugenio Scalise  
Adrian J. Mejias O. y Jose E. Tirado S.  
septiembre, 2022

# Índice general

<b>1. Introducción</b>	<b>4</b>
<b>2. Tecnologías para la visualización de datos en web</b>	<b>5</b>
2.1. Librerías o frameworks para aplicaciones intensivas de frontend	5
2.2. Librerías para la visualización de datos . . . . .	5
2.2.1. Data Driven Documents (D3) . . . . .	6
2.2.2. echarts . . . . .	6
2.2.3. Comparación de bibliotecas . . . . .	6
2.3. Proyectos alternos . . . . .	7
2.3.1. echarts for react . . . . .	7
<b>3. Tecnologías para el desarrollo web</b>	<b>8</b>
3.1. Arquitectura . . . . .	9
3.2. Tecnologías para el desarrollo . . . . .	10
3.3. Servidor . . . . .	10
3.3.1. Fastify . . . . .	10
3.3.2. MongoDB . . . . .	11
3.4. Cliente . . . . .	13
3.4.1. ReactJS . . . . .	13
3.4.2. Next.js . . . . .	15
3.4.3. Material UI . . . . .	15
<b>4. Propuesta de Trabajo Especial de Grado</b>	<b>19</b>
4.1. Motivación e identificación del problema . . . . .	19
4.2. Objetivos del trabajo . . . . .	20
4.2.1. Objetivo General . . . . .	20
4.2.2. Objetivos Específicos . . . . .	20

4.3.	Estrategia de solución y método de desarrollo ágil a utilizar . . . . .	20
4.3.1.	Desarrollo Rápido de Aplicaciones (RAD) . . . . .	20
4.4.	Trabajos similares, diferencias y ventajas de la solución a desarrollar	22
4.4.1.	Toolforge . . . . .	22
4.4.2.	Sigma - Summary . . . . .	23
4.4.3.	Histropedia Timeline . . . . .	24
4.4.4.	Xtools . . . . .	25
4.4.5.	Wiki Replay . . . . .	25
4.4.6.	IBM History Flow tool . . . . .	26
4.4.7.	Wiki History Flow . . . . .	29
4.5.	Herramientas . . . . .	30
4.5.1.	Tecnologías para el ambiente de desarrollo . . . . .	30
4.5.2.	Tecnologías o librerías web a utilizar . . . . .	30
4.6.	Requerimientos . . . . .	31
4.7.	Prototipo de interfaz . . . . .	31
4.7.1.	Perfil de Watcher \watcher\watcherId . . . . .	31
4.7.2.	Página principal \home . . . . .	32
4.7.3.	Landing page \ . . . . .	34
4.7.4.	Página de configuración \settings . . . . .	34
4.7.5.	Página de artículo \article . . . . .	34
4.7.6.	Sobre nosotros \about . . . . .	35
4.8.	Modales . . . . .	35
4.8.1.	Modal de autenticación . . . . .	35
4.9.	Planificación de las actividades . . . . .	36

# índice de figuras

3.1.	Número de peticiones por segundo para distinta cantidad de conexiones . . . . .	11
3.2.	Ranking de bases de datos más populares de 2022 . . . . .	12
4.1.	Fases del Desarrollo Rápido de Aplicaciones (enfoque de James Martin) . . . . .	21
4.2.	Sigma Summary . . . . .	23
4.3.	Línea de tiempo que muestra la colección de arte del Museo del Prado . . . . .	24
4.4.	Histropedia Timeline . . . . .	25
4.5.	Interfaz de wikireplay . . . . .	26
4.6.	Explicación del mecanismo de visualización de History Flow .	28
4.7.	La página "Brazil" mostrando un crecimiento abrupto y pocas contribuciones anónimas . . . . .	29
4.8.	History Flow creado por el proyecto . . . . .	30
4.9.	Prototipo de página de perfil de watcher . . . . .	32
4.10.	Prototipo de vista principal de un usuario autenticado . . . . .	33
4.11.	Prototipo de página de artículos . . . . .	34
4.12.	Modal de autenticación . . . . .	35

# Capítulo 1

## Introducción

Un wiki es un sitio web que permite a sus usuarios colaborar en su estructura y contenido. Esta versatilidad que provee el concepto de Wiki es lo que lo convierte en una de las herramientas más usadas en la actualidad para compartir información. La enciclopedia Wikipedia es el sitio web más popular basado en wiki.

El principal problema que maneja Wikipedia en cuanto a moderación de contenido viene como resultado de su propia filosofía “todos pueden editar”, lo que conlleva a múltiples problemas tales como: vandalismo, escritura pobre, una mala estructura de página, peleas de edición, entre otras cosas. Por esta razón no existe una solución única para acabar con la existencia de “mal” contenido en Wikipedia, y es indispensable el uso de participación humana en procesos de moderación que implican complejos desafíos técnicos y éticos.

En la actualidad, gracias a la evolución del internet, la información es considerada virtualmente ubicua y en constante cambio, y lo que realmente ofrece valor es la capacidad individual de sintetizar esa información y relacionarla. Como resultado de esto surge la filosofía wiki, en donde la información se comparte, y el conocimiento no se crea, sino se co-crea de forma colaborativa.

El concepto de la filosofía de wiki y el software utilizado para crear estos sitios web están intrínsecamente relacionados, y no se podría poner en práctica lo primero sin lo segundo. Esto es así debido a que el software debe proporcionar el medio para que pueda existir esa construcción colectiva de conocimiento, que es indispensable en la filosofía wiki.

# **Capítulo 2**

## **Tecnologías para la visualización de datos en web**

Para el análisis de datos, el paso fundamental es la visualización de datos, sin este paso los analistas de datos simplemente no podrían comparar grupos de datos eficientemente. Esta tarea además de importante es compleja y necesita de agrupaciones que se encarguen de trabajar en ella constantemente. Afortunadamente estas agrupaciones publican sus esfuerzos en plataformas como Github para así poder aportar a la investigación, discutir soluciones, revisar problemas y atraer nuevos colaboradores.

### **2.1. Librerías o frameworks para aplicaciones intensivas de frontend**

### **2.2. Librerías para la visualización de datos**

Pero no cualquier librería Para la selección de librería se consideran los siguientes factores:

1. Debe ser un proyecto open source.
2. De ser posible, debe tener bindings para ReactJS para facilidad en el desarrollo.

3. Debe ser extensible para poder implementar aquellas visualizaciones que sean muy específicas.
4. Deben ser longevas y tener cierta garantía de que será mantenida en el tiempo, asegurando así que para futuras mejoras de este TEG sigan disponibles.
5. Debe ser fácil de utilizar

### **2.2.1. Data Driven Documents (D3)**

- URL de repositorio <https://github.com/d3/d3>
- Fecha de última versión: 11 de Abril de 2022
- Mantenida por la organización homónima

### **2.2.2. echarts**

- URL de repositorio <https://github.com/apache/echarts>
- Fecha de última versión
- Mantenida por Apache

### **2.2.3. Comparación de bibliotecas**

Después de comparar estas librerías, sus APIs y leer otras investigaciones la decisión final es utilizar echarts

La base de esta conclusión es la simplicidad [3], mientras que d3 permite una gran flexibilidad, la curva de aprendizaje tiene una inclinación pronunciada. En comparación, echarts se ve como una herramienta plug and play que requiere mínima configuración y ahorra al desarrollador días de esfuerzo.

El tipo de gráficas a utilizar son las siguientes

- Número
- Línea
- Barra
- Área

- Torta
- Dispersión
- History flow

Para estas gráficas echarts provee soluciones pre-hechas y si es el caso de gráficos de interés en nichos específicos, como es el caso del History Flow, permite que fácilmente crees gráficas completamente nuevas. También provee gráficos de Sankey y Stacked Area Chart que son bastante parecidos y podrían ser personalizados para parecerse al History Flow.

## 2.3. Proyectos alternos

Estos dos gigantes en el mundo de la visualización de datos por si solos son opciones excelentes, pero trabajan sobre javascript vainilla y no están adaptadas para trabajar directamente sobre tecnologías como ReactJS. Para facilidad del desarrollo de la aplicación se utilizará una librería que se encargue de adaptar echarts a ReactJS.

### 2.3.1. echarts for react

- URL del repositorio <https://github.com/hustcc/echarts-for-react>

# Capítulo 3

## Tecnologías para el desarrollo web

Así como el propósito de un libro es ser leído y su éxito se mide en la cantidad de personas que lo lean. El éxito de un sitio web informativo se mide en su crecimiento; vive para tener una relación simbiótica con sus usuarios, donde ellos proveen el contenido y con él atraen lectores que se convierten en editores y en nuevos creadores. El sitio web por su parte, aporta todas las herramientas necesarias para que todos puedan participar.

Con el tiempo, y si los usuarios tienen una experiencia positiva este "libro" web, se puede esperar que esté convertido en ecosistema de datos. Para garantizar una buena experiencia de usuario Petter Morville [9] propone tener en cuenta siguientes factores:

- Utilidad: el producto realizado debe ser de utilidad para el público objetivo.
- Usabilidad: debe ser un sistema que sea simple y fácil de usar para el usuario.
- Deseabilidad: cuando un producto es deseable, los usuarios se sienten atraídos a él.
- Encontrable: se debe construir un sistema que sea navegable e intuitivo al recorrer, de tal forma que los usuarios puedan encontrar lo que necesiten fácilmente.

- Accesibilidad: el sistema debe poder ser usado por la mayor cantidad de personas posibles.
- Credibilidad: el sistema debe proporcionar seguridad al usuario acerca de la credibilidad de su contenido
- Valor: una experiencia de usuario valiosa implica un buen uso de los demás factores.

En este capítulo presentaremos todas las herramientas que nos facilitarán lograr la mejor experiencia de usuario.

### 3.1. Arquitectura

Para que una aplicación sea encontrable — y así usada — por internautas es fundamental que tenga una buena relación con los motores de búsqueda.

Sin embargo también para asegurar la larga vida y mantenibilidad de la aplicación y la facilidad de desarrollo se debe tomar en cuenta herramientas extensamente empleadas contemporáneamente como Angular, React y Vue.

El problema entonces recae en que estas tecnologías por si solas son meramente para aplicaciones de una sola página. Los motores de búsqueda hacen lo que pueden pero el HTML estático manejado por las SPA es mínimo, resultando en que los motores de búsqueda no puedan inferir de qué trata la página, afectando su habilidad de ser encontrada e indexada.

Como remedio surge un nuevo paradigma con el nombre de Server Side Rendering o simplemente SSR [7]; donde se utilizan tecnologías SPA como motor de plantillas, para así generar HTML digerible por los motores de búsqueda. Una vez el HTML estático llega al cliente, este atraviesa un proceso conocido como *hydration*, donde retoma sus funcionalidades dinámicas características de SPA.

Así se logra el perfecto balance de herramientas actuales y fáciles de usar, que también cumplen con los requerimientos de los motores de búsqueda para indexar páginas web.

## 3.2. Tecnologías para el desarrollo

Actualmente existe un catálogo muy amplio de tecnologías de desarrollo web, por lo que escoger las debidas herramientas y analizar sus compatibilidades es clave para obtener el mejor conjunto de herramientas.

El este capítulo hablaremos de las herramientas de desarrollo web usadas para llevar a cabo el proyecto, y las descompondremos en 2 categorías: Servidor y Cliente.

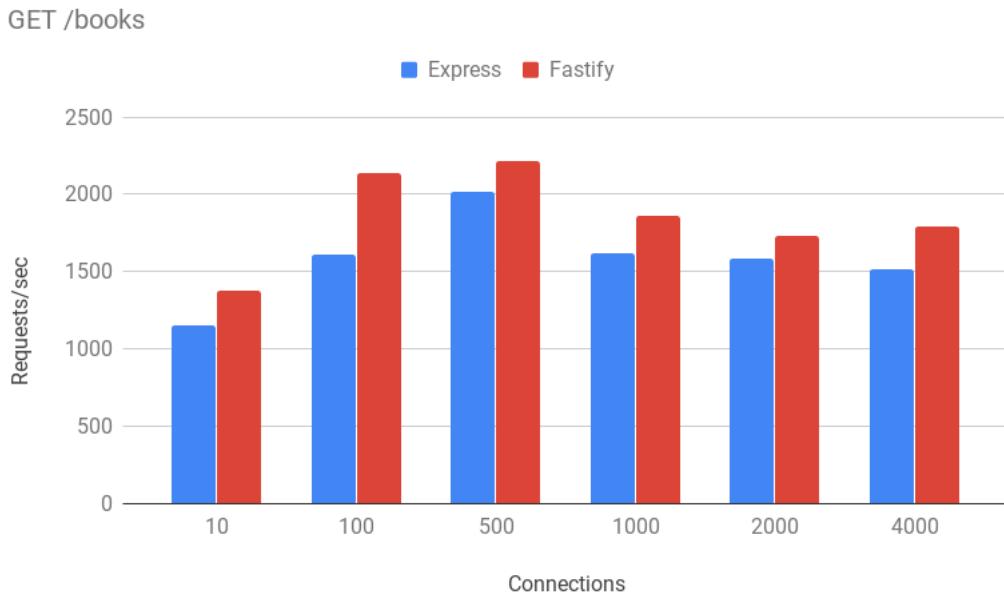
## 3.3. Servidor

### 3.3.1. Fastify

Fastify es un framework web para NodeJS de código abierto concentrado en proporcionar el mejor rendimiento, y una arquitectura flexible.

Las principales características de Fastify son [1]:

- Alto rendimiento: si se compara la velocidad de Fastify con otros frameworks web como Express, tal como se aprecia en la figura 3.1, notamos que Fastify es aproximadamente un 20 % más rápido que Express.
- Extensible: Fastify es completamente extensible mediante el uso de *hooks*, plugins o decoradores.
- Basado en esquemas: a pesar de no ser obligatorio, es recomendado usar el esquema JSON para validar las rutas y serializar las salidas, internamente Fastify compila el esquema en una función de alto rendimiento.
- Registro: Usualmente el guardar un registro puede ser costoso, por esa razón Fastify hace uso de un logger eficiente que casi remueve ese costo.
- Amigable para los desarrolladores: este framework está desarrollado para ser bastante expresivo y ayudar a los desarrolladores en su uso, sin sacrificar rendimiento o seguridad.
- Listo para usar con Typescript: la instalación inicial del framework provee compatibilidad predeterminada con Typescript



**Figura 3.1:** Número de peticiones por segundo para distinta cantidad de conexiones

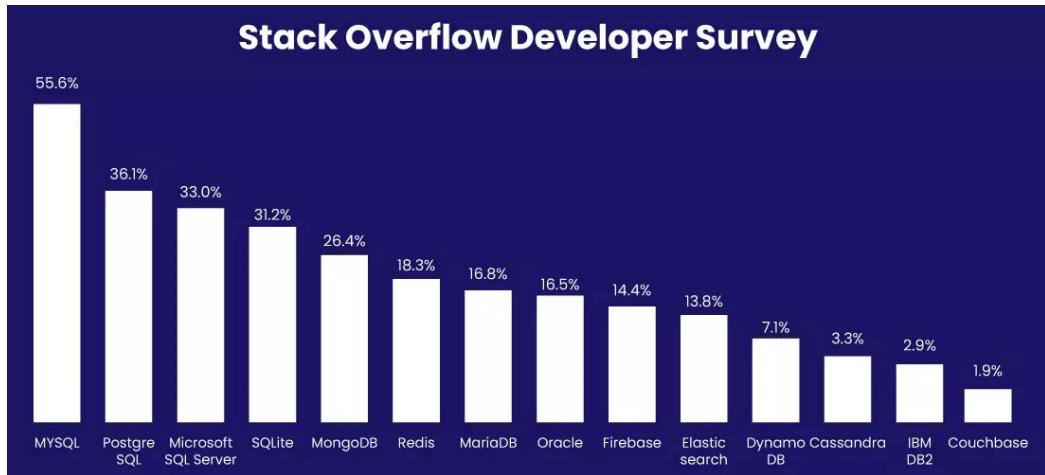
### 3.3.2. MongoDB

MongoDB es un sistema manejador de base de datos de documentos NoSQL escalable y flexible diseñado para lidiar con los conflictos que poseen las bases de datos relacionales y las limitaciones de otras soluciones NoSQL. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, lo que facilita la integración de los datos.

No tener restricciones al estructurar los datos facilita el desarrollo, siempre y cuando el desarrollador sea consciente de la redundancia que puede causar en la base de datos. Los documentos BSON almacenados en Colecciones dentro de MongoDB, estos son similares al conocidísimo estándar JSON. Las consultas y mutaciones en MongoDB no son hechas en un lenguaje de consulta como SQL sino serializadas en JSON.

Como se puede observar en la imagen 3.2, mongoDB es actualmente la opción más usada entre las bases de datos NoSQL, ocupando el puesto

número 5 en el ranking de bases de datos propuesto por Stack Overflow [10]. Esto proporciona suficiente certeza de que MongoDB será mantenido por años en el futuro.



**Figura 3.2:** Ranking de bases de datos más populares de 2022

Algunas de las características más importantes de MongoDB son:

- Consultas ad-hoc optimizadas para análisis en tiempo real
- Indexación adecuada para mejores tiempos de ejecución de consultas
- Replicación para una mejor disponibilidad y estabilidad de los datos
- *Sharding*<sup>1</sup>
- Balanceo de carga

Para realizar operaciones sobre la base de datos MongoDB usando un servidor NodeJS es necesario usar un MongoDB Driver compatible con NodeJS, el cual permite establecer una conexión con la base de datos, y que provee una API que facilita la interacción con la misma. A continuación se mostrará un código básico de conexión y consulta de datos usando el NodeJS MongoDB Driver.

---

<sup>1</sup> *Sharding*: proceso de distribuir un conjunto de datos en múltiples bases de datos, que después pueden ser almacenados en múltiples ordenadores.

```

import { MongoClient } from "mongodb";

const uri = "<connection string uri>";

const client = new MongoClient(uri);

async function run() {

    await client.connect();

    const database = client.db("sample_mflix");

    const movies = database.collection("movies");

    const query = { title: "The Room" };

    const options = {};

    const movie = await movies.findOne(query, options);

}

```

En el código se puede observar como se utiliza el cliente de MongoDB para conectarse a la base de datos «sample\_mflix» y después consultar todas las películas cuyo título sea igual a «The Room».

## 3.4. Cliente

Para este proyecto se usarán las siguientes tecnologías:

### 3.4.1. ReactJS

ReactJS es una librería de JavaScript de código abierto desarrollada por Facebook para facilitar la creación de componentes interactivos y reutilizables, para desarrollo de interfaces de usuario, especialmente aplicaciones de una sola página.

React maneja el concepto de “programación reactiva” [2] haciendo uso de un DOM Virtual, lo que le permite determinar qué partes del DOM han cambiado comparando contenidos entre la versión nueva y la almacenada en el

DOM virtual, para así propagar los datos generando cambios en la aplicación, es decir, la aplicación “reacciona” a cambios en los datos ejecutando una serie de eventos y re-renderizando solo aquellos componentes que lo ameriten. Esto es el secreto de porqué React es altamente eficiente.

Otras características que destacan en React son:

- **Componentes**

El código de React es hecho con entidades llamadas componentes. Los componentes pueden ser renderizados en elementos particulares del DOM usando la librería de React DOM. Estos componentes son capaces de recibir parámetros conocidos como ”propiedades del componente” de la siguiente forma:

```
ReactDOM.render(<Greeter greeting="Hello World!" />, document.  
getElementById('myReactApp'));
```

Las 2 formas de declarar componentes en React es mediante el uso de funciones o clases, y generalmente se usa una de las dos opciones de forma situacional.

- **JSX**

JSX, también llamado Javascript XML, es una extensión a la sintaxis del lenguaje javascript. Este provee una forma de estructurar componentes usando una sintaxis familiar para muchos desarrolladores. Los componentes de React son usualmente escritos en JSX, aunque también pueden ser escritos usando Javascript puro.

Un ejemplo de código JSX:

```
class App extends React.Component {  
  render() {  
    return (  
      <div>  
        <p>Header</p>  
        <p>Content</p>  
        <p>Footer</p>  
      </div>  
    );  
  }  
}
```

## ■ Hooks

Los hooks son funciones que permiten a los desarrolladores “engancharse” a los estados de React y a ciertos puntos dentro del ciclo de vida de los componentes.

React proporciona algunos hooks integrados tales como: useState, useContext, useReducer, useMemo y useEffect son los más usados y permiten controlar los estados y manejar eventos.

### 3.4.2. Next.js

Next.js es un framework desarrollado sobre NodeJS que permite a las aplicaciones de React usar funcionalidades como el renderizado del lado servidor y la generación de páginas web estáticas.

Por defecto, Next.js pre-renderiza cada página. Esto significa que Next.js genera HTML para cada página en adelanto, en vez de hacerse con Javascript del lado del cliente. Pre-renderizado puede resultar en mejor rendimiento y SEO.

Cada HTML generado es asociado con el mínimo código Javascript necesario para que funcione la página. Cuando una página es cargada en el explorador, su código javascript se ejecuta y hace la página totalmente interactiva. A este proceso le conoce como “hydration”

Next.js ofrece 2 formas de pre-renderizado:

- Generación estática: El HTML es pre-generado y puede ser servido estáticamente.
- Renderizado del lado servidor: El HTML es generado al recibir una petición del cliente, necesita un servidor que procese las peticiones.

### 3.4.3. Material UI

Material UI es un framework de React creado con el objetivo de proporcionar una forma sencilla a los desarrolladores de aplicar los principios de Material Design [5] usando componentes de React. Este framework se destaca por la gran variedad de componentes disponibles, que van desde elementos visuales sencillos como botones o selectores, hasta complejos como modales o drawers.

Además de su variedad de componentes, Material UI también destaca por su gran cantidad de contribuyentes, lo que asegura la longevidad y mantenibilidad del framework.

Entre las principales características que posee Material UI se encuentran:

- Posibilidad de crear temas, lo que permite personalizar los estilos predeterminados de los componentes de la librería de forma global, o acotada.
- Construido usando la estrategia Mobile First [6], en la cual se crea el código primero para dispositivos móviles, y después se escalan usando las reglas de CSS pertinentes.
- Los componentes de Material UI se consideran autosuficientes, por lo que solo utilizan los estilos CSS necesarios para mostrar el componente en pantalla, es decir, no dependen de hojas de estilos globales como en el caso de normalize.css o bootstrap.

A continuación se mostrará el uso de algunos componentes que ofrece la librería:

- Botones

```
<Button variant="text">Text</Button>
<Button variant="contained">Contained</Button>
<Button variant="outlined">Outlined</Button>
<Button color="secondary">Secondary</Button>
<Button size="small">Small</Button>
<IconButton aria-label="delete">
    <DeleteIcon />
</IconButton>
```

Material UI viene incluido con 3 variantes visuales de botones, los cuales son usados dependiendo de la importancia o énfasis que se le quiera dar al botón. Además, también posee variaciones de color y tamaño predeterminados y la posibilidad de usar íconos que funcionan como botones.

- Selectores

```
<FormControl fullWidth>
    <InputLabel id="demo-simple-select-label">Age</InputLabel>
    <Select>
```

```

labelId="demo-simple-select-label"
id="demo-simple-select"
value={age}
label="Age"
onChange={handleChange}
>
<MenuItem value={10}>Ten</MenuItem>
<MenuItem value={20}>Twenty</MenuItem>
<MenuItem value={30}>Thirty</MenuItem>
</Select>
</FormControl>

```

El selector comúnmente se encuentra envuelto en un componente de FormControl, el cual provee contexto para hacer uso de campos requeridos, manejo de errores y uso de propiedades como "seleccionado." o "campo lleno".

- Sliders

```

const marks = [
  {
    value: 0,
    label: "0km"
  },
  {
    value: 20,
    label: "20km"
  },
  {
    value: 37,
    label: "37km"
  },
  {
    value: 100,
    label: "100km"
  },
];
<Slider aria-label="Volume" value={25} onChange={handleChange}>
<Slider defaultValue={30} step={10} marks min={10} max={110} disabled />
<Slider
  aria-label="Custom marks"
  defaultValue={20}
  getAriaValueText={valuetext}>

```

```
step={10}  
valueLabelDisplay="auto"  
marks={marks}  
/>>
```

Material UI permite personalizar los Sliders de múltiples maneras, tales como usar valores discretos, cambiar el tamaño del Slider, o utilizar íconos personalizados en los extremos o en la perilla de este, lo que contextualiza al usuario sobre lo que representa el valor del Slider.

# **Capítulo 4**

## **Propuesta de Trabajo Especial de Grado**

### **4.1. Motivación e identificación del problema**

El orgullo de Wikipedia como la mayor fuente de información (en forma de artículos sobre eventos, personajes, definiciones, conceptos, ubicaciones y entre otros...) se debe a su extensa red de colaboradores.

Cada vez que estos ejecutan una acción en la Wikipedia -ya sea si aportan artículo complejo o corrigen una tilde- dejan una huella llamada metadata. Estos rastros”, conformados por fechas, número de líneas editadas, direcciones IP y demás, son de suma importancia para analistas de datos.

Estos, si tienen la experiencia, la práctica y tienen el tiempo para dedicarse a estudiar la API de Wikipedia; pueden tomar la metadata, procesarla y generar gráficos para buscar patrones de interés.

El proceso es largo, y su complejidad limita quien puede formar parte. Si se tuviera una solución donde aquellos interesados ”no técnicos” puedan ayudar y donde los interesados experimentados no tengan que dedicarle exhaustivas cantidades de tiempo para poder aportar.

Este trabajo final de grado propone entonces una herramienta que facilita esta laboriosa tarea para todo tipo de usuarios. Dándoles una interfaz fácil de usar y flexible para generar gráficos

## **4.2. Objetivos del trabajo**

### **4.2.1. Objetivo General**

Crear una nueva versión del front-end de wikimetrics y extender con funcionalidades pertinentes para fomentar discusión sobre los artículos

### **4.2.2. Objetivos Específicos**

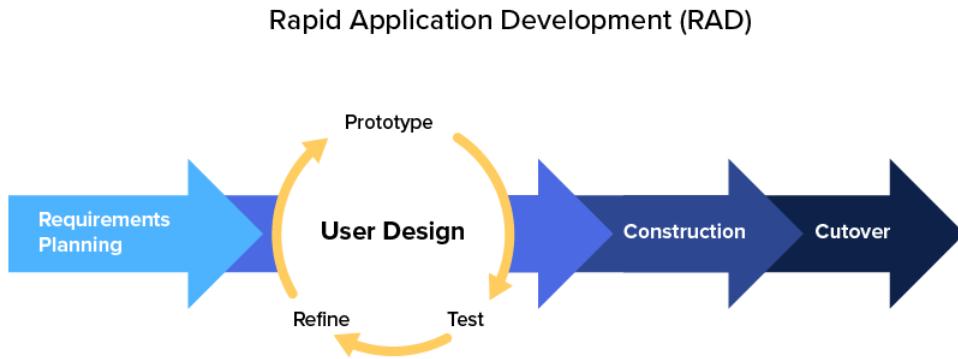
- Implementar una aplicación web responsive que ofrezca las funcionalidades requeridas por un watcher de un wiki y que pueda ser reconocida por los motores de búsqueda.
- Consumir y extender la API de wikimetrics para desarrollar una aplicación web que habilite a sus usuarios construir y visualizar gráficas
- Definir los requerimientos de la aplicación
- Utilizar un método ágil para el desarrollo de la aplicación.
- Realizar el despliegue y puesta en producción de la aplicación

## **4.3. Estrategia de solución y método de desarrollo ágil a utilizar**

### **4.3.1. Desarrollo Rápido de Aplicaciones (RAD)**

El desarrollo rápido de aplicaciones es una metodología de desarrollo que prioriza la creación rápida y prototipos y la retroalimentación rápida sobre ciclos prolongados de desarrollo y prueba. Con el uso de RAD, los desarrolladores pueden realizar múltiples iteraciones y actualizaciones al software rápidamente sin la necesidad de iniciar un cronograma de desarrollo desde cero cada vez.

Esta metodología de desarrollo fue creada por James Martin en el año 1980 en IBM y fue formalizada con la publicación del libro *Rapid Application Development* en 1991.



**Figura 4.1:** Fases del Desarrollo Rápido de Aplicaciones (enfoque de James Martin)

### Fases del Desarrollo Rápido de Aplicaciones

Según el enfoque de James Martin [4] el Desarrollo Rápido de Aplicaciones se divide en las siguientes fases:

- **Fase de planificación:** Esta fase es equivalente a una reunión de alcance del proyecto. Aunque la fase de planificación se condensa en comparación con otras metodologías de gestión de proyectos.

Durante esta etapa, los desarrolladores, los clientes (usuarios de software) y los miembros del equipo se comunican para determinar los objetivos y las expectativas del proyecto, así como los problemas actuales y potenciales que deben abordarse durante la construcción.

- **Fase de diseño:** Durante esta fase, los usuarios interactúan con analistas de sistemas y desarrollan modelos y prototipos que representan todos los procesos, entradas y salidas del sistema.

Todos los errores y problemas se resuelven en un proceso iterativo. El desarrollador diseña un prototipo, el cliente (usuario) lo prueba y luego se reúnen para comunicar qué funcionó y qué no.

- **Fase de construcción:** Esta etapa toma los prototipos y el sistema

en su fase beta resultado de la etapa de diseño y lo convierte en un modelo funcional.

Debido a que la mayoría de los problemas y cambios se abordaron durante la minuciosa fase de diseño iterativo, los desarrolladores pueden construir el modelo de trabajo final más rápidamente que siguiendo un enfoque tradicional de gestión de proyectos.

- **Fase de transición:** Esta es la fase de implementación donde el producto terminado va al lanzamiento. Incluye conversión de datos, pruebas y cambio al nuevo sistema, así como capacitación de usuarios.

Todos los cambios finales se realizan mientras los programadores y los clientes continúan buscando errores en el sistema.

## 4.4. Trabajos similares, diferencias y ventajas de la solución a desarrollar

Siendo Wikimedia y Wikipedia las más grandes comunidades y organizaciones dedicadas a recopilar datos, es lógico pensar que tiene consigo una abundante cantidad de seguidores capacitados y apasionados por aportar lo que puedan. A continuación se detallará el estado actual de las herramientas existentes para explorar la Wikipedia.

Estas herramientas cumplen diferentes objetivos:

1. Identificar posibles candidatos para ser administrador.
2. Identificar peleas de edición.
3. Proveer una forma visual de ver el desarrollo general de los artículos (crecimiento de artículo, detección de actividad maliciosa, entre otras cosas).
4. Hacer uso de la información para proveer interfaces educativas tales como cronologías.

### 4.4.1. Toolforge

Es una suite de herramientas estadísticas para las páginas, usuarios, wikis de Wikimedia. Entre las herramientas hosteadas más populares de toolforge

se encuentra

- URL del proyecto <https://wikitech.wikimedia.org/wiki/Portal:Toolforge>

Comenzando con el proyecto madre de Wikimedia, este gigante sirve de plataforma para que colaboradores técnicos puedan usar servidores de Wikipedia para desarrollo y para poder levantar aplicaciones que mejoren la experiencia de todos los colaboradores de Wikipedia.

Este proyecto es particularmente interesante porque podría ser un medio por el cual la aplicación WikiMetaView y sus dependencias podrían ser servidas al público.

[https://www.wikidata.org/wiki/Wikidata:Tools/Visualize\\_data](https://www.wikidata.org/wiki/Wikidata:Tools/Visualize_data)

#### 4.4.2. Sigma - Summary

Esta herramienta creada por el usuario SigmaΣ permite obtener el historial de contribuciones de un usuario y filtrar las ediciones por una palabra clave. Tal como se puede observar en la imagen 4.2, el resultado de buscar el usuario “Clarityfiend” y filtrar por la palabra “space” es similar al que se puede encontrar en la lista de contribuciones por usuario de Wikipedia [12], con el beneficio de poder filtrar por palabras claves.

- URL del proyecto <https://sigma.toolforge.org/summary.py>

**Edit summary search**

This tool searches through a user's contribution history and returns edits made by that user if the edit summary contains the specified string.

Username: Clarityfiend Search: space Max pages: 500  
Database: enwiki Advanced options | Submit

(328 remaining)

- 06:02, 03 July 2022 (diff | hist) . (+1) . m Model 98a rifle (\* History 'γ-space')
- 06:56, 12 May 2022 (diff | hist) . (+566) . Wikipedia:Reference desk/Humanities (\* Using female pronouns to refer to starships/spaceships \*)
- 02:10, 02 May 2022 (diff | hist) . (-276) . Chink (disambiguation) /Reverted [[WP:AGF|good faith]] edits by [[Special:Contributions/75.61.99.105|75.61.99.105|talk]]: A disambiguation page is for navigation to mainspace articles, not a dictionary.)
- 05:06, 03 April 2022 (diff | hist) . (+1) . m Hawker Hurricane (\* Phony War γ+space)
- 22:41, 31 March 2022 (diff | hist) . (+3) . m List of people known as the Great (\* Aristocrats γ space after 'c.')
- 09:55, 18 March 2022 (diff | hist) . (+1) . m Ghobalan Salami (\* top γ-blank space)
- 21:02, 14 March 2022 (diff | hist) . (+2) . m Suma-class cruiser (\* Ships in class γ-spaces)
- 08:58, 20 February 2022 (diff | hist) . (+1) . m Bee bearding (\* Bee-bearding records γ-space)
- 00:11, 18 February 2022 (diff | hist) . (+170) . Wikipedia:Reference desk/Science (\* Did anyone NASA or any space agency save Earth from major calamity? \*)
- 07:01, 17 February 2022 (diff | hist) . (+192) . Wikipedia:Articles for deletion/Examples of vector spaces (\* Examples of vector spaces γ comment)
- 10:19, 13 February 2022 (diff | hist) . (+4) . m The Pied Piper of Cleveland (\* top γ removed odd spaces)
- 22:12, 12 January 2022 (diff | hist) . (+26) . Wikipedia:Reference desk/Miscellaneous (\* Longest straight flight only in a single nation's airspace plus international waters. \*)
- 22:09, 12 January 2022 (diff | hist) . (+245) . Wikipedia:Reference desk/Miscellaneous (\* Longest straight flight only in a single nation's airspace plus international waters. \*)

Figura 4.2: Sigma Summary

#### 4.4.3. Histropedia Timeline

- URL del proyecto <http://histropedia.com/timeline/>

Es una herramienta que permite visualizar diferentes eventos en forma de línea de tiempo interactiva, usando el servicio de Wikidata para consultas Sparql [11]. Es usada como herramienta educativa por distintas entidades como el Museo del Prado para que los usuarios puedan visualizar de forma sencilla el arte que se expone, tal como se puede observar en la imagen 4.3. Además, la línea de tiempo permite asociar cada elemento de la colección con un evento histórico importante, de tal forma que se facilite el aprendizaje y se haga más dinámico.

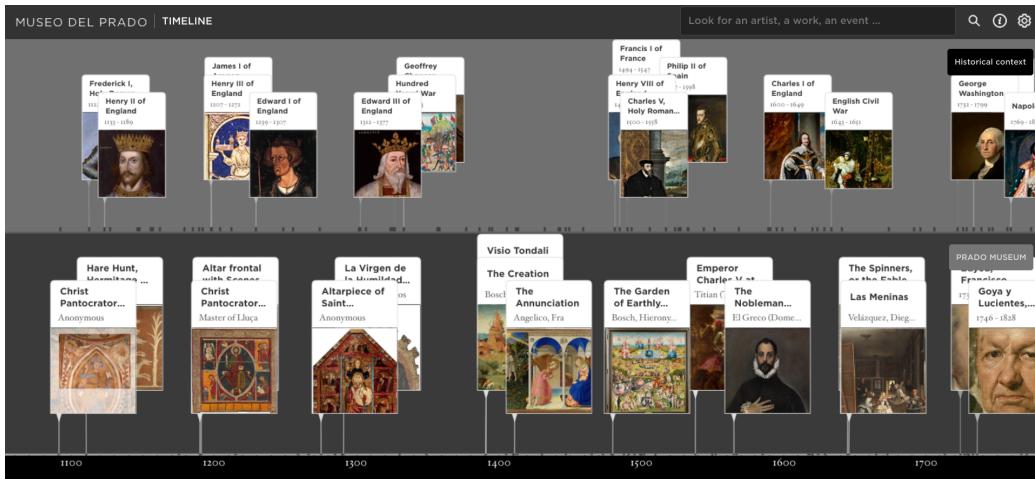
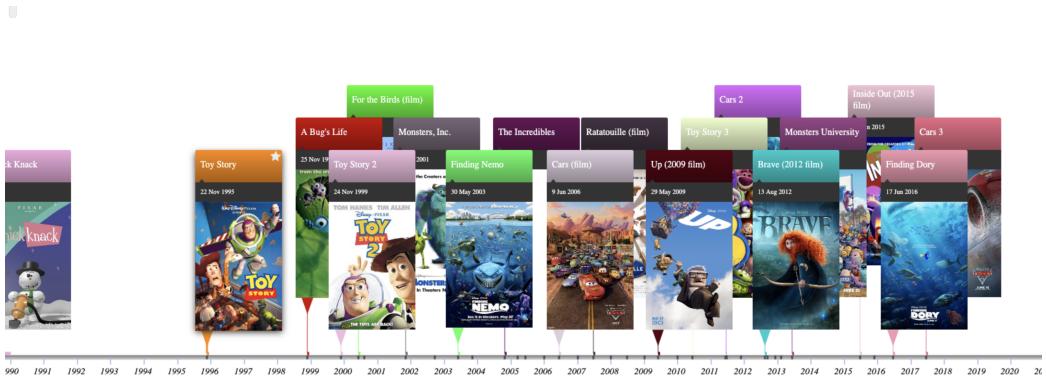


Figura 4.3: Línea de tiempo que muestra la colección de arte del Museo del Prado



**Figura 4.4: Histropedia Timeline**

<https://en.wikipedia.org/wiki/Wikipedia:Tools>

[apersonbot.toolforge.org](https://apersonbot.toolforge.org) Este set de herramientas está enfocado en supervisar usuarios y sus contribuciones a la Wikipedia

- Candidate Search - Adminscore - Articles for Creation Review History

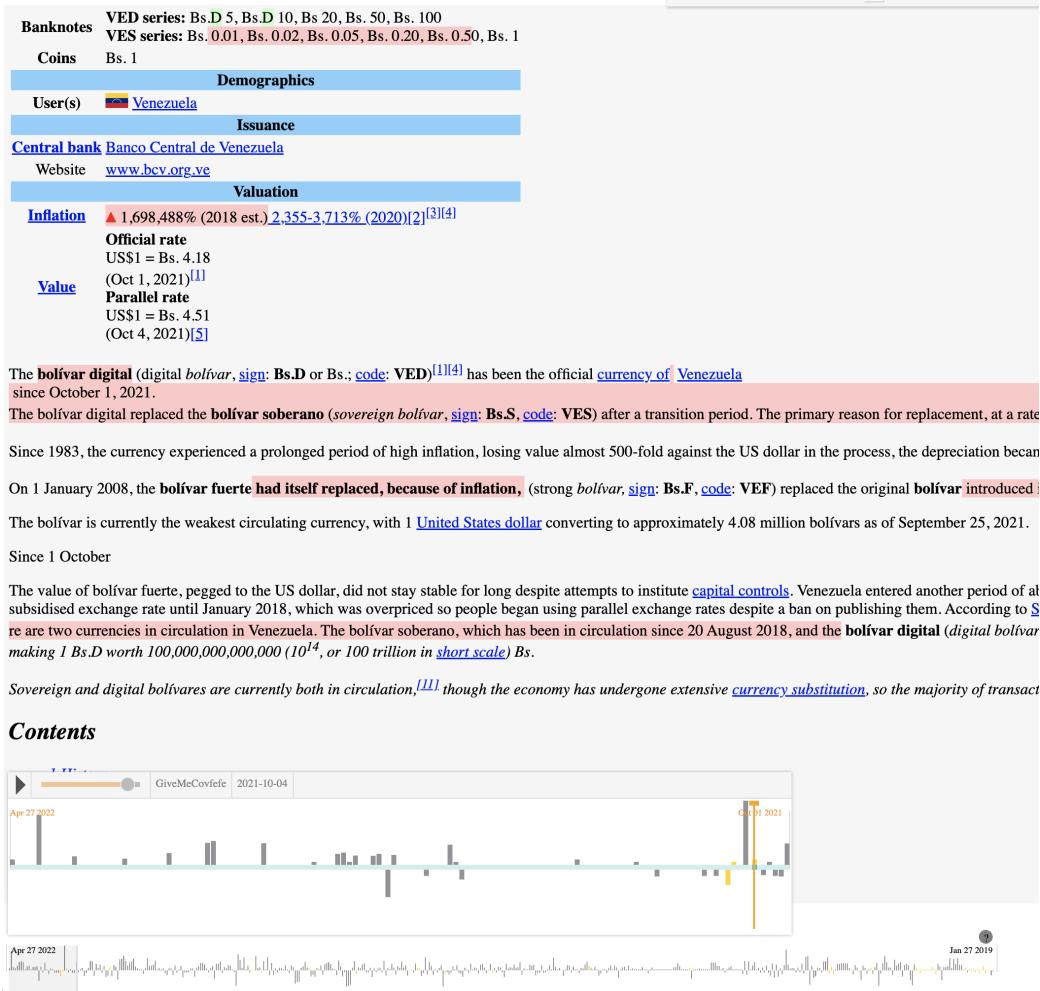
#### 4.4.4. Xtools

- URL del proyecto <https://xtools.wmflabs.org/>

#### 4.4.5. Wiki Replay

- URL del proyecto <https://cosmiclattes.github.io/wikireplay/player>

Se encarga de mostrar una línea de tiempo con los cambios en un artículo. Esta línea de tiempo tiene el comportamiento de un reproductor de video y permite que el usuario presione un botón de reproducir y así ser guiado por como el contenido de un artículo ha ido evolucionando en el tiempo.



**Figura 4.5:** Interfaz de wikireplay

[https://de.wikipedia.org/wiki/Benutzer:Atlasowa/edit\\_history\\_visualization](https://de.wikipedia.org/wiki/Benutzer:Atlasowa/edit_history_visualization)

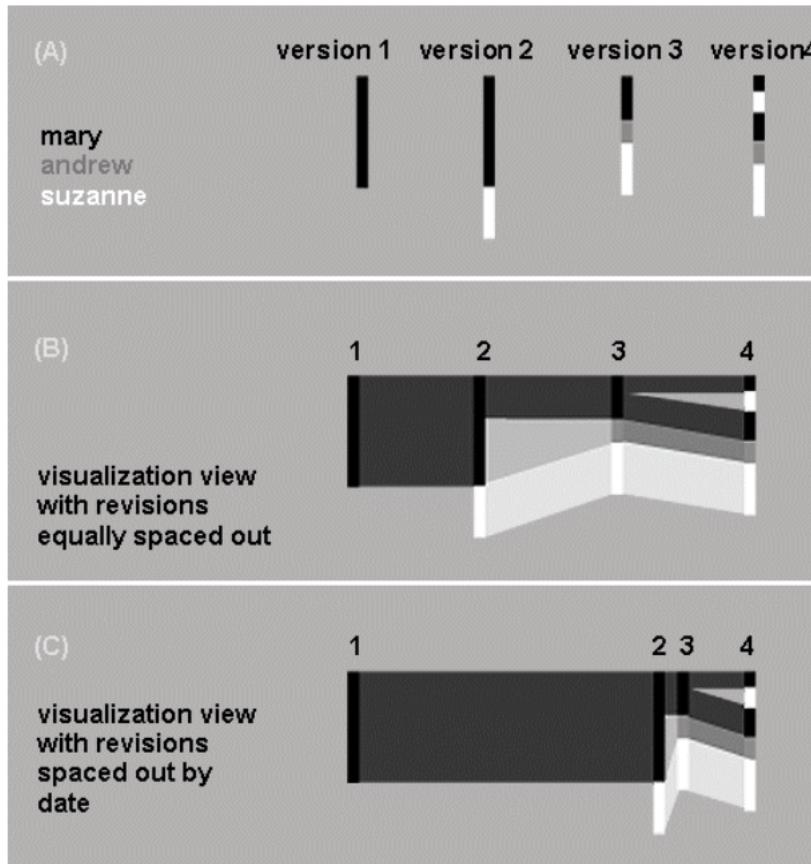
#### 4.4.6. IBM History Flow tool

- URL del proyecto [http://alumni.media.mit.edu/~fviegas/papers/history\\_flow.pdf](http://alumni.media.mit.edu/~fviegas/papers/history_flow.pdf)

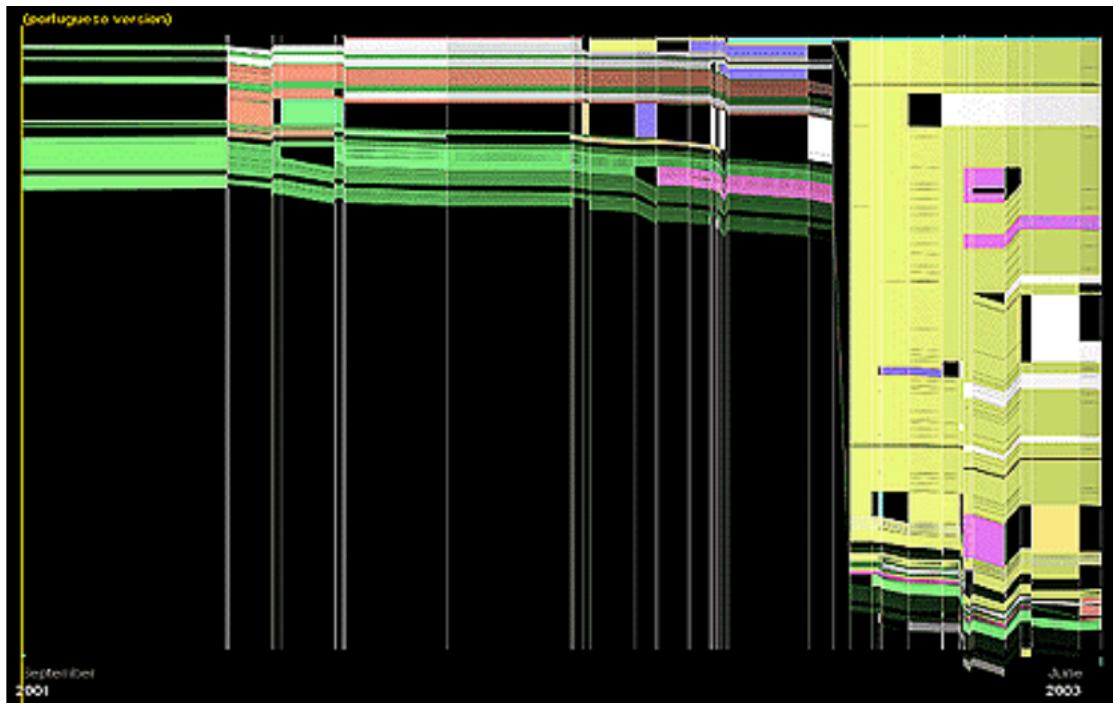
Es una herramienta de análisis de datos exploratorio, la cual utiliza el

registro de ediciones que posee Wikipedia para mostrar relaciones entre distintas versiones de un documento. El propósito de esta herramienta es mostrar tendencias o patrones generales del documento, pero al mismo tiempo preservar los detalles para una examinación cercana.

Como se puede observar en la imagen 4.6 cada barra vertical representa una versión del documento, siendo el largo proporcional a la cantidad de palabras que posee el documento. El color de la versión en toda su extensión permite asociar cada palabra con un usuario, identificando el autor de dicha versión. Por último, se utiliza la distancia entre versiones para separarlas de acuerdo a su fecha y así obtener información extra sobre la frecuencia de edición de cada palabra. Al realizar el procedimiento en la página “Brazil” 4.7 se puede observar un evidente incremento en la longitud de las barras verticales, lo que implica un crecimiento drástico del contenido de la página.



**Figura 4.6:** Explicación del mecanismo de visualización de History Flow

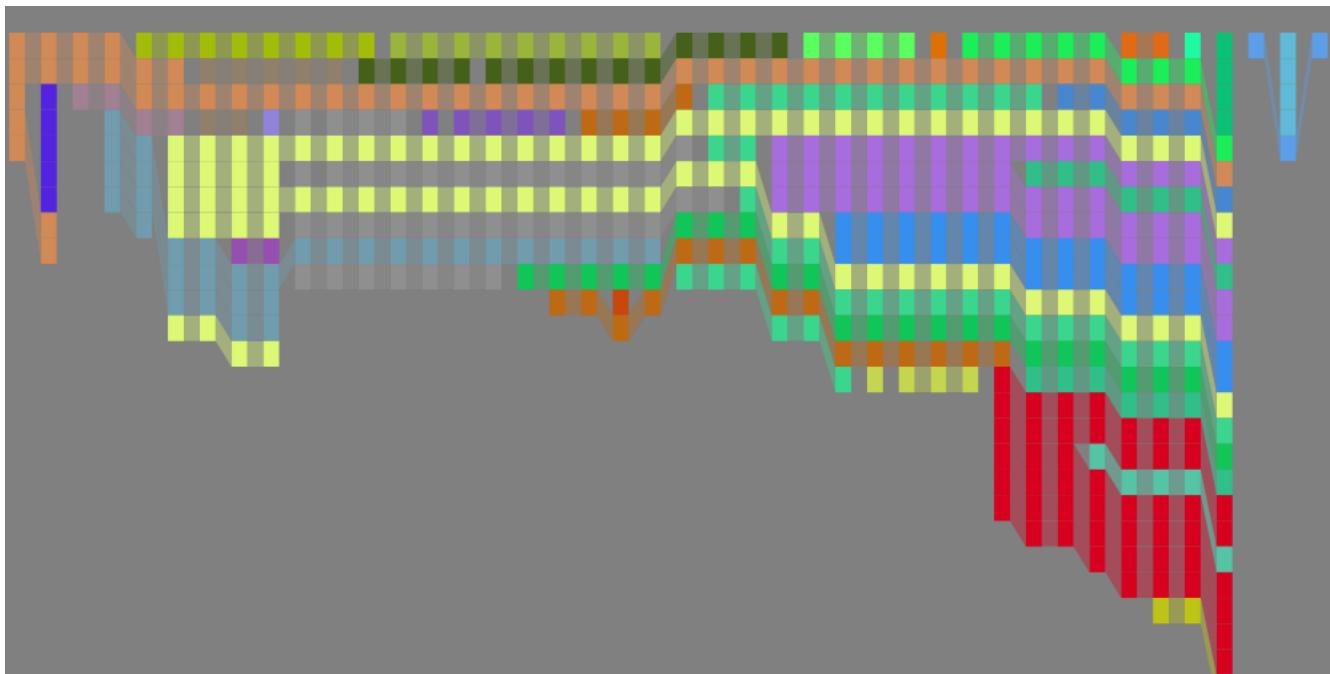


**Figura 4.7:** La página "Brazil"mostrando un crecimiento abrupto y pocas contribuciones anónimas

#### 4.4.7. Wiki History Flow

- URL del proyecto <https://github.com/rdmpage/wikihistoryflow>
- 

Es un repositorio que posee varios archivos de PHP que permiten crear el history flow de una página en formato SVG. Este proyecto obtiene las revisiones directamente de Wikipedia haciendo uso de web scrapping y calculando la diferencia entre textos con el paquete de Pear Text\_Diff [8], para después graficar las diferencias en forma de history flow.



**Figura 4.8:** *History Flow* creado por el proyecto

## 4.5. Herramientas

### 4.5.1. Tecnologías para el ambiente de desarrollo

- Visual Studio Code.
- Git, como control de versiones.
- Navegadores web (Google Chrome, Firefox, etc)
- Base de Datos (MongoDB)
- Servidor (Nodejs)
- Manejador de paquetes de NPM (pnpm)

### 4.5.2. Tecnologías o librerías web a utilizar

- Material UI

- Next.js + ReactJS
- Node
- MongoDB
- Fastify

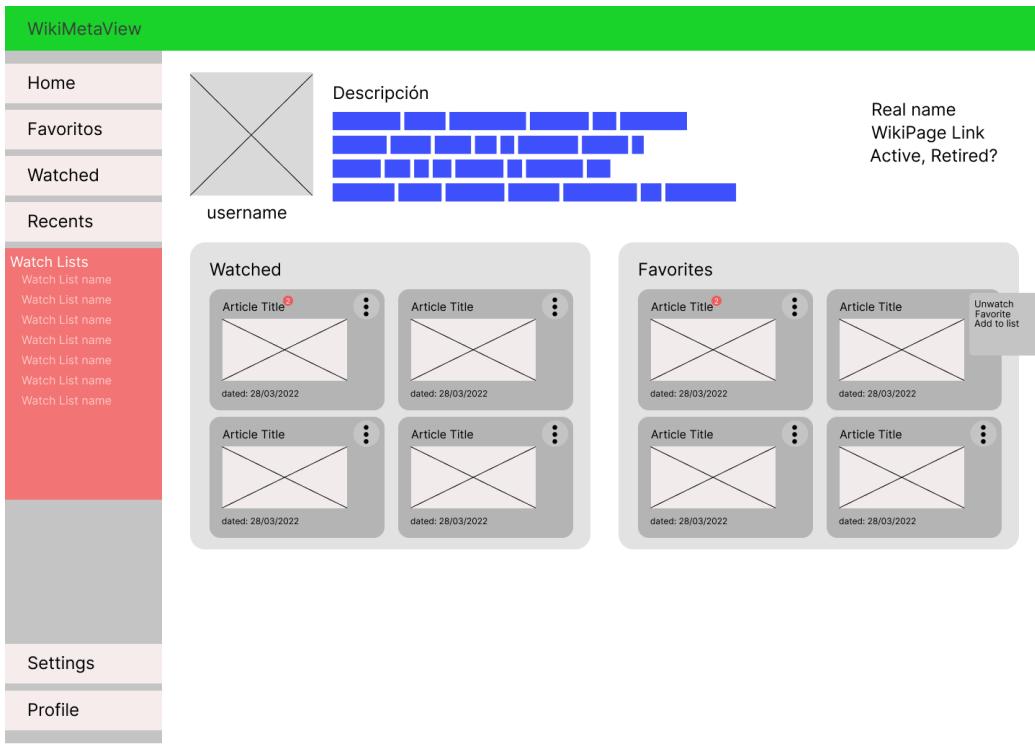
## 4.6. Requerimientos

1. Una herramienta capaz de visualizar gráficas sobre propiedades de edición de un wiki.
2. Los wikis a visualizar son extraídas del watchlist del usuario autenticado a través de Wikipedia (usando API de MediaWiki).
3. Permitir editar las visualizaciones (alternar tipo de gráfica, cambiar los tipos de datos a visualizar).
4. Los datos a representar en las visualizaciones son proporcionados a través del API Wikimetrics 2.0.
5. La herramienta tiene que ser una aplicación web.
6. La aplicación tiene que contar con una vista principal que incluya gráfica e información general de cada uno de los wikis del watchlist.
7. Permitir al usuario crear una visualización nueva y guardarla de forma permanente, para este caso es necesario el uso de una base de datos (preferiblemente MongoDB)
8. La aplicación tiene que estar alojada en un servidor, para poder ser accedida de manera remota.

## 4.7. Prototipo de interfaz

### 4.7.1. Perfil de Watcher \watcher\watcherId

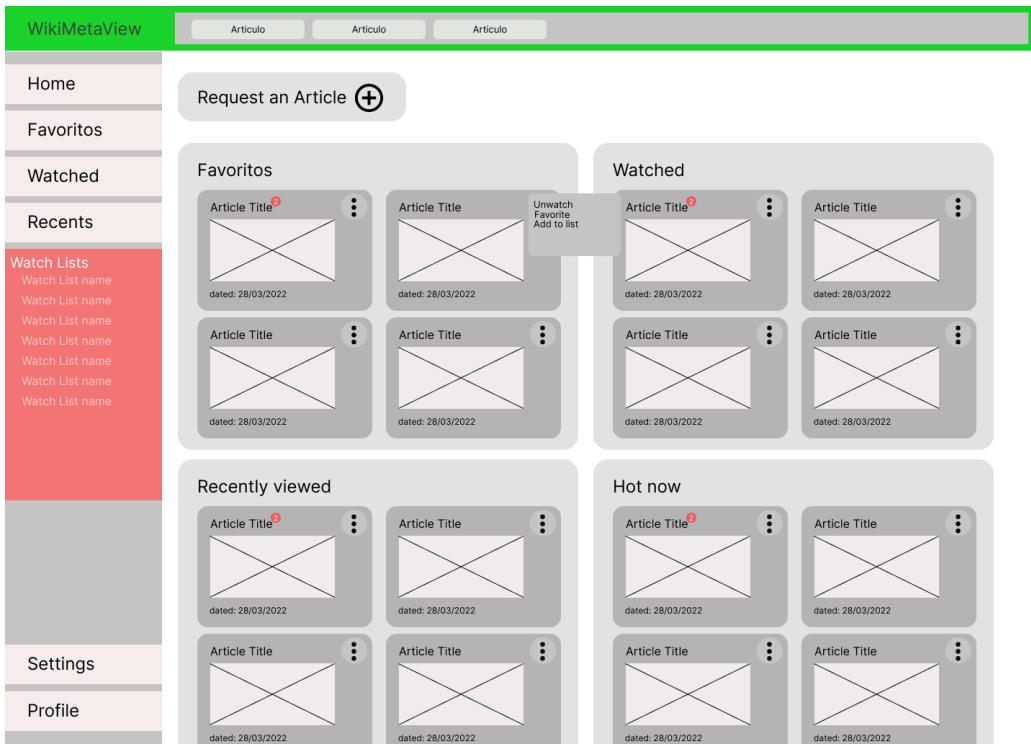
Para cualquier usuario muestra el perfil de un watcher, en él puedes ver todas las contribuciones de este junto con la lista de visualizaciones que ha creado. Se permitirá un mínimo de personalización. Y se podrá enlazar el perfil de Wikipedia



**Figura 4.9:** Prototipo de página de perfil de watcher

#### 4.7.2. Página principal \home

Para usuarios autenticados, les permite ver diferentes secciones relevantes para ellos



**Figura 4.10:** Prototipo de vista principal de un usuario autenticado

### Secciones del home

#### 1. Watched

Muestra una lista y un pequeño abstracto de los artículos que el usuario hace watching y tienen una visualización actualizada

#### 2. Queue

Muestra una lista de los artículos que el usuario solicito para hacer una visualización pero que el server no ha podido solicitar

#### 3. Controversial

Muestra una lista de aquellas visualizaciones que provocan discusión en la comunidad, caracterizado por muchos comentarios

#### 4.7.3. Landing page \

TODO: poner imagen Esta página se encarga de explicar las motivaciones de la App y dar una noción básica del funcionamiento para los watchers. Tiene el objetivo de cap

#### 4.7.4. Página de configuración \settings

TODO: Terminar sección de settings Acá el usuario puede bla bla

#### 4.7.5. Página de artículo \article

/article?title=<title>&url=<url> Muestra un artículo junto con su metadata y las visualizaciones creadas por los usuarios.



Figura 4.11: Prototipo de página de artículos

#### 4.7.6. Sobre nosotros \about

Breve resumen del proyecto, incluye el documento de tesis y documento de seminario así como datos de contacto y repositorios de github para futuros contribuyentes.

### 4.8. Modales

Las modales pueden sobreponerse a cualquier página en la aplicación y permiten al usuario efectuar una acción.

Estas modales serán controladas por parámetros de consulta agregados al URL de cada página web CleanArquitecture

#### 4.8.1. Modal de autenticación

Si este parámetro está en el URL. Presenta la modal de autenticación

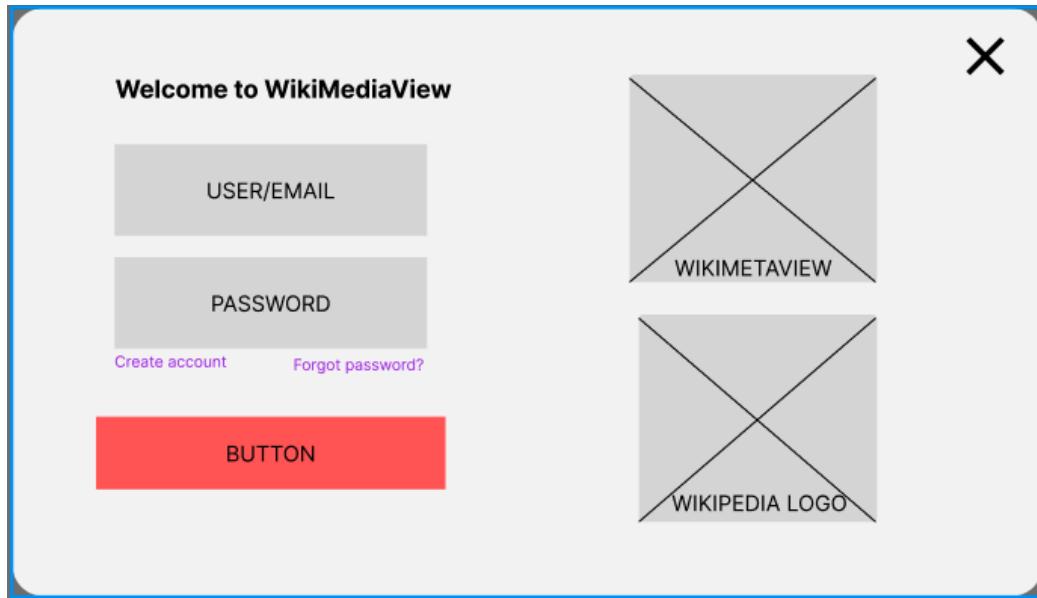


Figura 4.12: Modal de autenticación

?auth=true

## 4.9. Planificación de las actividades

TODO: UNA TABLA DE DOS COLUMNAS

ACTIVIDAD, TIEMPO ESTIMADO.

Preparar el entorno de desarrollo 1/2 semana
Estudiar API Wikimetrics 2.0 (Back-end) 1/2 semana
Estudiar API MediaWiki 1/2 semana
Integracion del API Wikimetrics 2.0 (Back-end) 1/2 semana
Integracion del API MediaWiki 1/2 semana

# Bibliografía

- [1] Fastify. *Fastify core features*. Fastify. URL: <https://www.fastify.io/#:~:text=featured%20here%3F-,Core%20features,-These%20are%20the.>
- [2] Emily Jiang Grace Jansen. *Defining the term “reactive”*. IBM. URL: <https://developer.ibm.com/articles/defining-the-term-reactive/>.
- [3] Clement Ho. *Why we chose echarts*. Gitlab. URL: <https://about.gitlab.com/blog/2019/09/30/why-we-chose-echarts/>.
- [4] James Martin. *Rapid Application Development*. IBM. 1991. URL: <https://archive.org/details/rapidapplication00mart>.
- [5] *Material Design Principles*. Google. URL: <https://material.io/design/introduction#principles>.
- [6] Mozilla. *Mobile first*. Mozilla. 2022. URL: [https://developer.mozilla.org/en-US/docs/Web/Progressive\\_web\\_apps/Responsive/Mobile\\_first](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Responsive/Mobile_first).
- [7] Next.js. *Server side rendering*. Vercel. URL: <https://nextjs.org/docs/basic-features/pages#server-side-rendering>.
- [8] Pear. *Pear library Text\_Diff*. Pear. URL: <https://pear.php.net/package/Text\Diff>.
- [9] Morville Petter. *UX Factors*. Semantic Studios. 2004. URL: [http://semanticstudios.com/user\\_experience\\_design/](http://semanticstudios.com/user_experience_design/).
- [10] *Stack Overflow 2022 Developer Survey*. Stack Overflow. 2022. URL: <https://survey.stackoverflow.co/2022/#databases>.

- [11] Wikidata. *Wikidata Sparql*. Wikidata. URL: [https://www.wikidata.org/wiki/Wikidata:SPARQL\\_tutorial](https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial).
- [12] Wikipedia. *Lista de contribuciones del usuario Clarityfiend*. Wikipedia. URL: <https://en.wikipedia.org/wiki/Special:Contributions/Clarityfiend>.