

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.  
mayo, 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) . . . . .	5
2.2.2. echarts . . . . .	6
2.3. Proyectos alternos . . . . .	6
2.3.1. recharts . . . . .	6
<b>3. Tecnologías para el desarrollo web</b>	<b>7</b>
3.1. Arquitectura . . . . .	9
3.2. Tecnologías para el desarrollo . . . . .	9
3.3. Servidor . . . . .	10
3.3.1. MongoDB . . . . .	10
3.3.2. Fastify . . . . .	10
3.4. Cliente . . . . .	10
3.4.1. ReactJS . . . . .	10
3.4.2. Next.js . . . . .	13
3.4.3. Material UI . . . . .	13
<b>4. Propuesta de Trabajo Especial de Grado</b>	<b>15</b>
4.1. Motivación e identificación del problema . . . . .	15
4.2. Objetivos del trabajo . . . . .	15
4.2.1. Objetivo General . . . . .	15
4.2.2. Objetivos Específicos . . . . .	15
4.3. Estrategia de solución y método de desarrollo ágil a utilizar . .	16

4.4.	Trabajos similares, diferencias y ventajas de la solución a desarrollar	16
4.5.	xtools	17
4.5.1.	wikireplay	17
4.5.2.	wikihistoryflow	17
4.6.	Herramientas	19
4.6.1.	Tecnologías para el ambiente de desarrollo	19
4.6.2.	Tecnologías o librerías web a utilizar	19
4.7.	Requerimientos	19
4.8.	Prototipo de interfaz	20
4.8.1.	Perfil de Watcher \watcher\:watcherId	20
4.8.2.	Página principal \home	20
4.8.3.	Landing page \	23
4.8.4.	Página de configuración \settings	23
4.8.5.	Página de artículo \article	23
4.8.6.	Sobre nosotros \about	24
4.9.	Modales	24
4.10.	Planificación de las actividades	24

# índice de figuras

3.1. Número de peticiones por segundo para distinta cantidad de conexiones . . . . .	11
4.1. Interfaz de wikireplay . . . . .	18
4.2. Prototipo de página de perfil de watcher . . . . .	21
4.3. Prototipo de vista principal de un usuario autenticado . . . . .	22
4.4. Prototipo de página de artículos . . . . .	23

# 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 mas 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

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

### 2.2. Librerías para la visualización de datos

Para la selección de librería se consideran los siguiente 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 sera mantenida en el tiempo, asegurando así.

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

- URL de repositorio <https://github.com/d3/d3>
- Fecha de ultima versión: 11 de Abril de 2022

### **2.2.2. echarts**

- URL de repositorio <https://github.com/apache/echarts>
- Fecha de ultima versión

## **2.3. Proyectos alternos**

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

### **2.3.1. recharts**

- URL de repositorio <https://github.com/recharts/recharts>
- Fecha de última versión: 1 de Abril de 2022

## Capítulo 3

# Tecnologías para el desarrollo web

El propósito de todo sitio web es ser utilizada y aprovechada por la mayor cantidad de personas posibles, y es por eso que el punto clave de desarrollo web se encuentra en cómo los desarrolladores proporcionan un sitio web que cumpla con ese propósito. Para conseguir dicho propósito se deben tener múltiples consideraciones que abarca desde la experiencia de usuario hasta la calidad de la infraestructura y el código realizado. Cuando hablamos de experiencia de usuario nos referimos al resultado de la interacción del usuario con los distintos elementos que posee el sitio web, de tal forma que el usuario pueda tener una experiencia positiva al hacer uso de dichos elementos. Petter Morville [**UXFactors**] propone que para lograr una buena experiencia de usuario se debe tener en cuenta el correcto uso de los 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 y fácil de encontrar, 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.

Sin embargo, para que sea posible cumplir con todos los factores que permiten una buena experiencia de usuario, es indispensable realizar un código de calidad que proporcione al sitio web un buen rendimiento, permitiendo al sitio web desempeñar su función correctamente.

En este capítulo presentaremos las herramientas que nos facilitaran el trabajo para lidiar con las problemáticas mencionadas. Brindar una buena experiencia de usuario en una aplicación puede llegar a ser un gran desafío donde existen varios elementos que afectan directamente, como el diseño, ya que es importante jugar un buen estilo, de carácter agradable, y con animaciones amigables. El rendimiento, donde la aplicación debe desempeñar una buena fluidez de respuesta y correctitud, que se contemple una interfaz que tenga sentido con la funcionalidad de la misma.

Cuando entramos en el área web, nuestro caso una aplicación web, llegan algunos desafíos adicionales como aplicar soporte de diseño para variedad de densidades de pantallas y versiones de navegadores web, esto se debe a que nuestra aplicación estará expuesta en la web, donde puede ser accedida desde tabletas, móviles, computadores de escritorio, televisores, y cualquier dispositivo que tenga conexión a internet y un navegador. También el peso de la aplicación, incluyendo códigos, fuentes e imágenes impacta considerablemente en el tiempo de espera para conexiones lentas a internet. En este capítulo presentaremos herramientas que nos facilitaran el trabajo para lidiar las problemáticas mencionadas segmentando la aplicación por: arquitectura, diseño y utilidades extras a emplear. Cabe acotar que para el manejo de estas herramientas es necesario tener conocimiento sobre JavaScript, TypeScript<sup>10</sup> (subconjunto de JavaScript con un sistema de tipos más robusto), CSS y HTML.

## 3.1. Arquitectura

Para que una aplicación sea descubierta y 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 son meramente para SPA. Lo que implica entonces que no existe una noción real”de seo - En las SPA el enrutamiento ocurre del lado del cliente usando javascript, y en consecuencia los crawlers de los motores de búsqueda no saben interpretar estas paginas.

Como remedio surge un nuevo paradigma, que es el que vamos a usar para esta aplicación, conocido como Server Side Rendering; donde se utiliza estas tecnologías SPA como un motor de plantillas para retornar un HTML que los motores de búsqueda puedan entender, y después por un proceso conocido como hydration, las aplicaciones en el lado del cliente dejan de comportarse como HTML plano y retoman sus funcionalidades de SPA.

Asi entonces llegamos al perfecto balance en el que tenemos herramientas actuales y fáciles de usar, que también cumplen con los requerimientos de los motores de búsqueda para indexar nuestras paginas.

## 3.2. Tecnologías para el desarrollo

Actualmente existe un catalogo muy amplio de tecnologías de desarrollo web, por lo que escoger las debidas herramientas y sobre todo analizar las compatibilidades entre estas herramientas pueden ser el punto clave para obtener la mejor eficiencia y experiencia de usuario.

Para nuestro proyecto haremos uso de un servidor cuyo propósito será obtener los datos y transformarlos de forma que el cliente los pueda usar directamente sin necesidad usar mucho computo, esto con el propósito de proporcionar una mejor experiencia de usuario.

El este capitulo 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. MongoDB**

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. 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, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

### **3.3.2. Fastify**

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

Si comparamos la velocidad de Fastify con otros frameworks web como Express, tal como podemos ver en la figura 3.1, notamos que Fastify es aproximadamente un 20 % más rápido que Express.

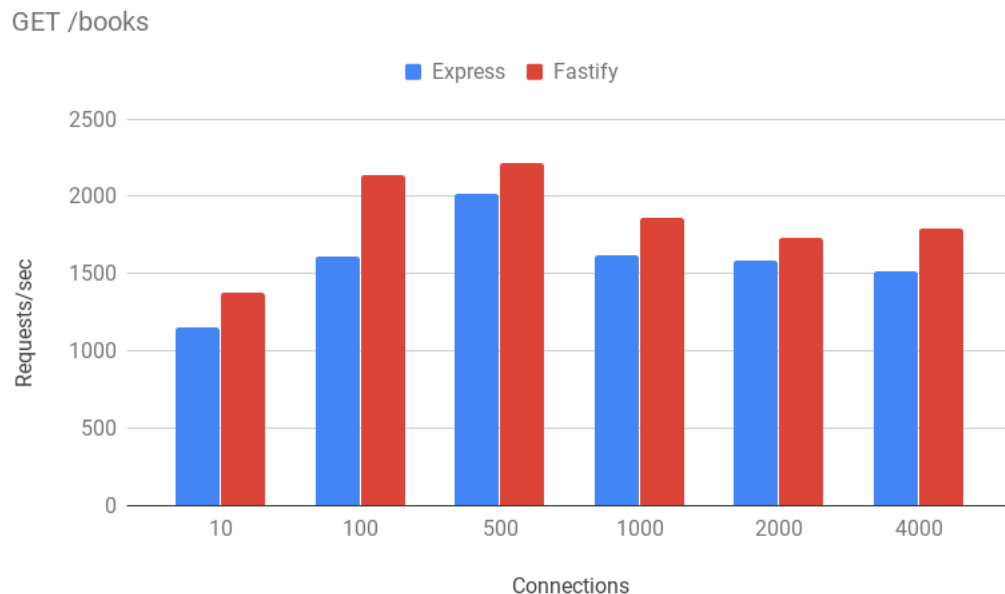
## **3.4. Cliente**

Para la realización de 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, reutilizables, para desarrollos de interfaces de usuario, especialmente aplicaciones de una sola página.

React maneja el concepto de “programación reactiva” haciendo uso de un DOM Virtual, lo le permite determinar qué partes del DOM han cambiado



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

comparando contenidos entre la versión nueva y la almacenada den el DOM virtual, para así propagar los datos generando cambios en la aplicación, es decir, los datos “reaccionan” ejecutando una serie de eventos.

Este concepto de reactividad es lo que hace a la librería altamente eficiente, ya que limita la actualización del DOM solamente a los elementos que han cambiado.

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 extension 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 usando 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, los cuales son los mas usados y permiten controlar los estados y eventos respectivamente.

### 3.4.2. Next.js

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

Por defecto, Next.js pre-renderiza cada pagina. Esto significa que Next.js genera HTML para cada pagina en adelante, 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 pagina. Cuando una página es cargada en el explorador, su código javascript se ejecuta y hace la página totalmente interactiva. A este proceso se le conoce como “hydration”

Next.js ofrece 2 formas de pre-renderizado:

- Generación estática: El HTML es generado a tiempo de ejecución y será reutilizado en cada petición.
- Renderizado lado servidor: El HTML es generado en cada petición

### 3.4.3. Material UI

Material UI es un framework de React que sigue los principios de Material Design ??, el cual fue creado por Google. Su objetivo es facilitar la creación de interfaces gráficas con el uso de componentes de React que ya fueron probados y aseguran una buena experiencia de usuario. Material UI se destaca ante otros frameworks UI por ser uno de los que posee mayor mantenibilidad (mayor cantidad de contribuyentes) y una gran variedad de componentes.

Material UI también posee la posibilidad de crear temas, lo que permite modificar los estilos de los componentes predeterminados de la librería de forma global en caso de necesitarse un diseño más personalizado.

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

- Botones

```
<Button variant="text">Text</Button>
<Button variant="contained">Contained</Button>
<Button variant="outlined">Outlined</Button>
```

donde el parametro variant representa una variante visual del boton, las cuales son usadas dependiendo del proposito/importancia del botón.

- 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 componente selector debe estar dentro de un componente de FormControl para que sea posible obtener sus datos al momento de enviar el formulario.

- Sliders

```
<Slider aria-label="Volume" value={25} onChange={handleChange}
/>
```

Donde su parametro value es el valor por default del slider y el parámetro onChange recibe una funcion que se ejecuta al mover el slider.

## Capítulo 4

# Propuesta de Trabajo Especial de Grado

### 4.1. Motivación e identificación del problema

### 4.2. Objetivos del trabajo

#### 4.2.1. Objetivo General

Crear una nueva version 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

TODO: DECIDIR ENTRE

- Kanban: Tiene como objetivo la mejora continua, la flexibilidad en la gestión de tareas y un flujo de trabajo mejorado. Con este enfoque ilustrativo, el progreso de todo el proyecto se puede comprender fácilmente de un vistazo. Para esto hace uso del tablero Kanban, que es una herramienta que visualiza todo el proyecto para rastrear el flujo de su proyecto. A través de este enfoque gráfico de los tableros Kanban, un miembro nuevo o una entidad externa puede comprender lo que está sucediendo en este momento, las tareas completadas y las tareas futuras.
- Rapid application development (RAD): es una forma de metodología de desarrollo de software ágil que prioriza las versiones e iteraciones rápidas de prototipos. A diferencia del método Waterfall, RAD enfatiza el uso de software y los comentarios de los usuarios sobre la planificación estricta y el registro de requisitos.

ELABORAR Y PONER DIBUJITOS ILUSTRATIVOS

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

Siendo wikimedia y wikipedia las mas 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 detallara el estado actual de las herramientas existentes para explorar la wikipedia.

Estas herramientas cumplen diferentes objetivos,

1. Identificar posibles candidatos para ser administrador
2. Identifica
3. Etc.

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

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

- Candidate Search - Aadminsore - Articles for Creation Review History

## 4.5. xtools

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

<https://sigma.toolforge.org/>

<https://interaction-timeline.toolforge.org/>

### 4.5.1. wikireplay

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

Se encarga de mostrar una linea de tiempo con los cambios en un articulo. Esta linea 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.

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

### 4.5.2. wikihistoryflow

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

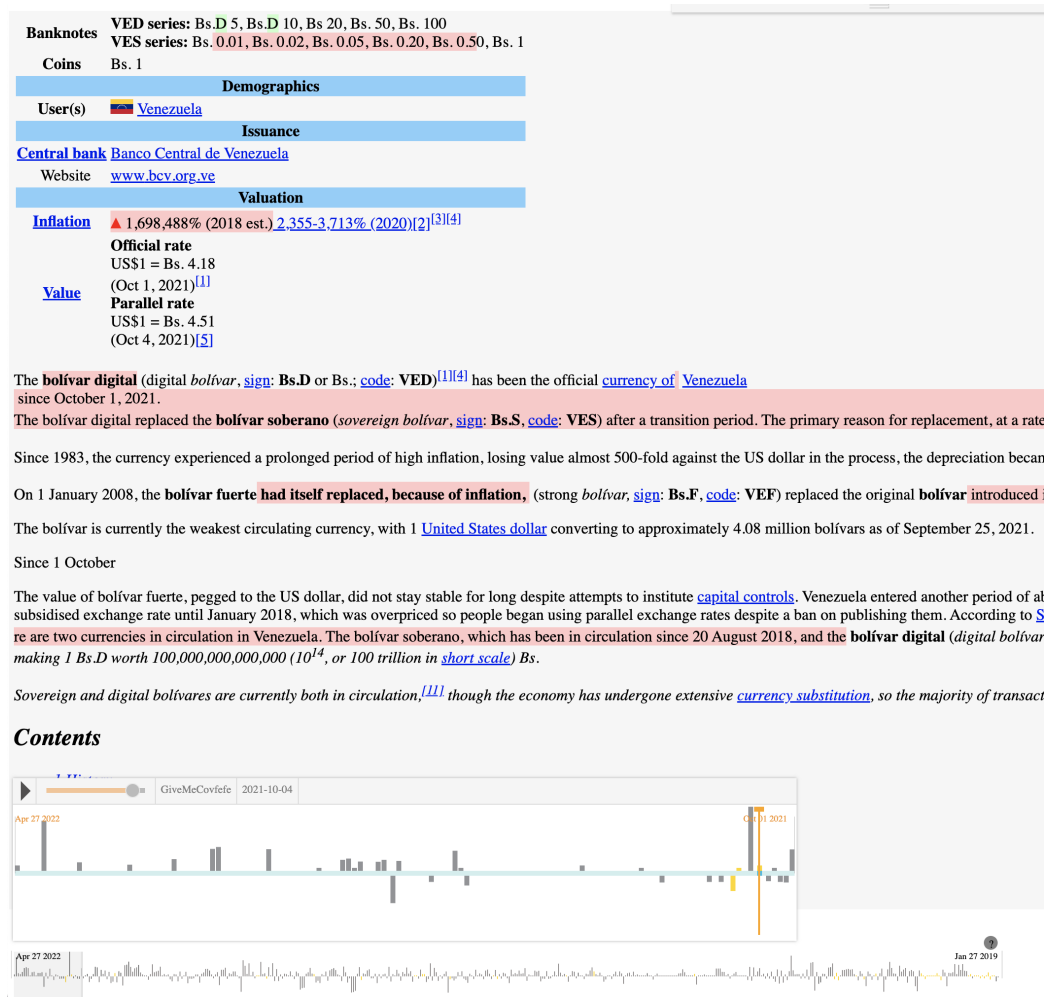


Figura 4.1: *Interfaz de wikireplay*

- 

Este proyecto genera una gráfica de history flow dado el url de un articulo

## 4.6. Herramientas

### 4.6.1. Tecnologías para el ambiente de desarrollo

- Visual Studio Code.
- Git, como control de versiones.
- Navegadores web (Google Chrome, Firefox, etc)
- MongoDB
- 

### 4.6.2. Tecnologías o librerías web a utilizar

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

## 4.7. Requerimientos

1. Una herramienta capaz de visualizar graficas sobre propiedades de edicion de un wiki.
2. Los wikis a visualizar son extraídas del watchlist del usuario autenticado a traves de Wikipedia (usando API de MediaWiki).
3. Permitir editar las visualizaciones (alternar tipo de grafica, cambiar los tipos de datos a visualizar).
4. Los datos a representar en las visualizaciones son proporcionados a traves 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 wiki 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.

## 4.8. Prototipo de interfaz

### 4.8.1. Perfil de Watcher \watcher\:watcherId

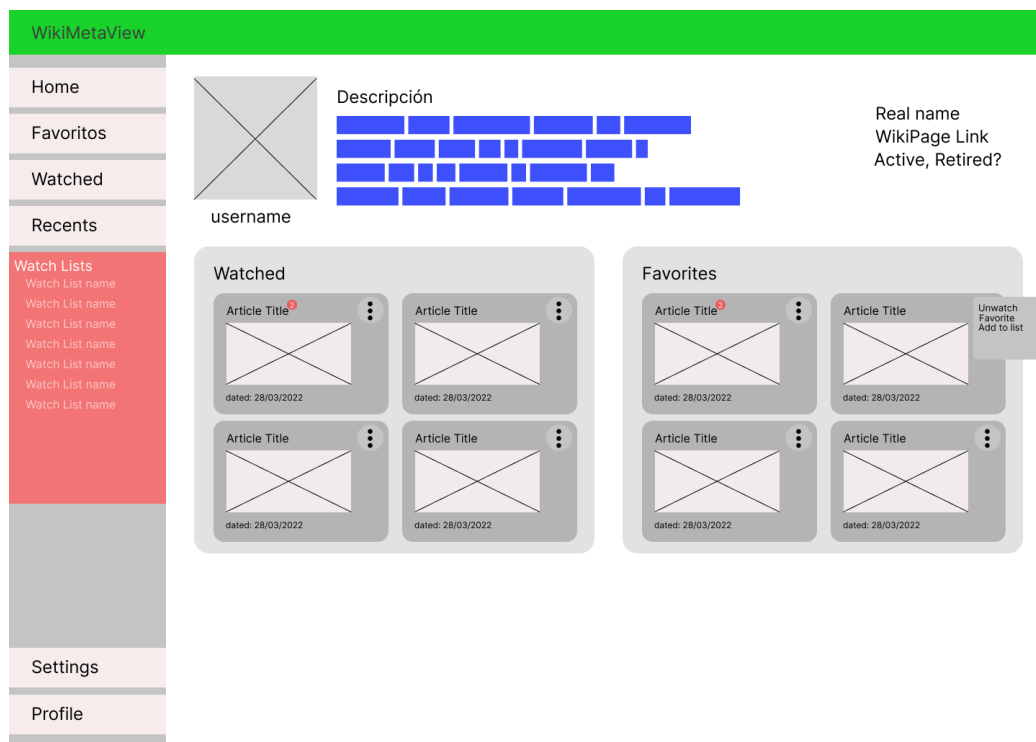
Para cualquier usuario muestra el perfil de un watcher, en él puedes ver todas las contribuciones de este junto con la lista de visualizaciones que ha creado. Se permitirá un mínimo de personalización. Y se podrá enlazar el perfil de wikipedia

### 4.8.2. Página principal \home

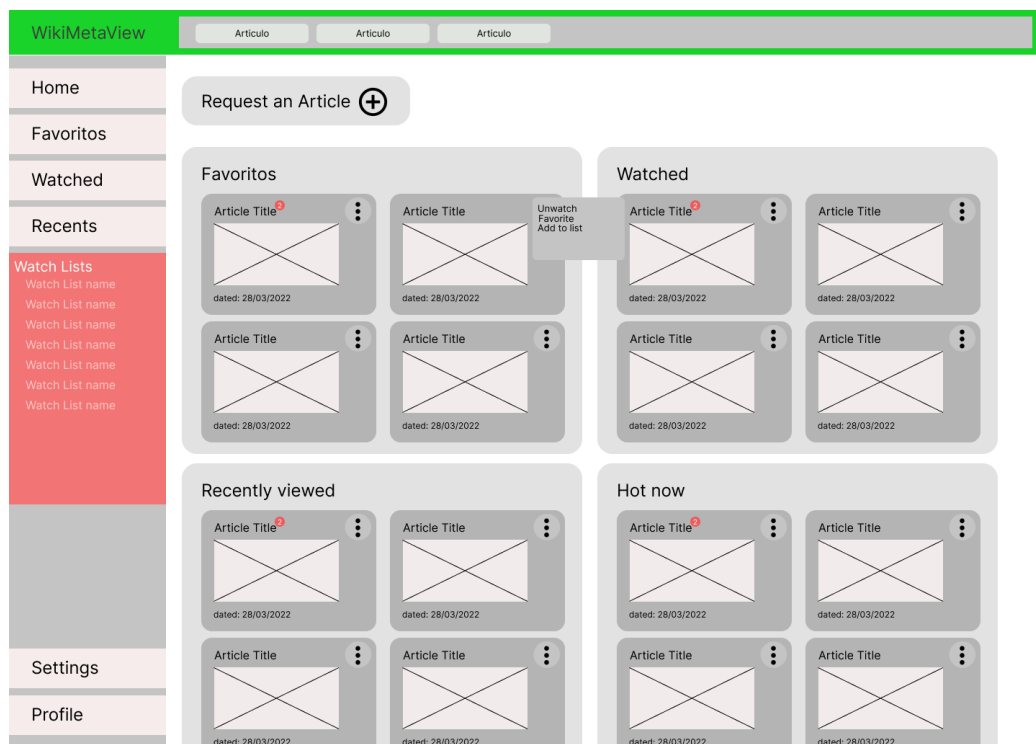
Para usuarios autenticados, les permite ver diferentes secciones relevantes para ellos

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 le 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



**Figura 4.2:** Prototipo de página de perfil de watcher



**Figura 4.3:** *Prototipo de vista principal de un usuario autenticado*



Figura 4.4: Prototipo de página de artículos

#### 4.8.3. Landing page \

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

#### 4.8.4. Página de configuración \settings

TODO: Terminar seccion de settings Aquí el usuario puede bla bla

#### 4.8.5. Página de artículo \article

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



#### 4.8.6. 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.

#### 4.9. Modales

Query Params Globales

?auth

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

?

#### 4.10. 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
Integración del API Wikimetrics 2.0 (Back—end) 1/2 semana
Integración del API MediaWiki 1/2 semana