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.
junio, 2022

Índice general

índice de figuras

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 utilizado y aprovechado por la mayor cantidad de personas posibles, y es por eso que el punto clave del 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 [2] 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 el.
- 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.

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. Fastify

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

Las principales caracteristicas con las que fastify fue creado son:

- Alto rendimiento: 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.
- Extensible: Fastify es completablemente extensible mediante el uso de hooks, puglins o decoradores
- Basado en esquemas: incluso si no es obligatorio, recomendamos usar JSON Schema para validar sus rutas y serializar sus salidas, internamente

Fastify compila el esquema en una función de alto rendimiento.

- Registro:
- Amigable para los desarrolladores:
- Listo para usar con Typescript:

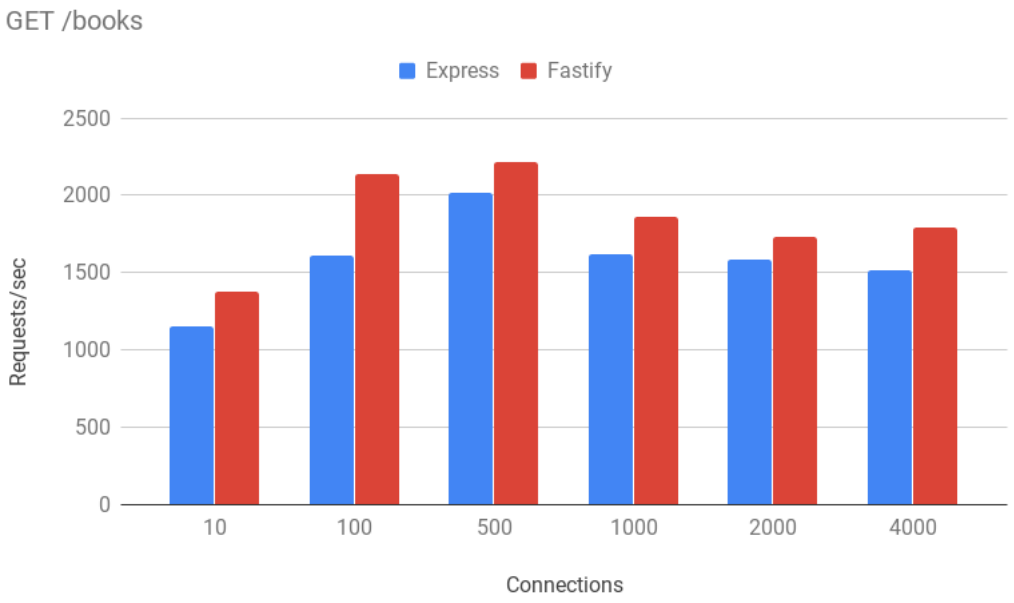

Figura 3.1: *Número de peticiones por segundo para distinta cantidad de conexiones*

3.3.2. MongoDB

MongoDB es una sistema 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.

Uno de los principales beneficios del uso de base de datos NoSQL antes que SQL es la facilidad de manejo de los datos, ya que solo se consultan documentos sin necesidad de realizar operaciones SQL complejas que requieren

conocimiento previo de la estructura de la base de datos. Ademas, debido a la similitud entre la especificación BSON y JSON, las consultas a la base de datos resultan transparentes en comparación a su contraparte SQL, el cual necesita de un algoritmo para convertir los datos tabulares en algo que pueda ser comprendido por el lenguaje de programación usado en el servidor backend.

Como se puede observar en la imagen 3.2, mongoDB es actualmente la opción mas usada entre la categoria de bases de datos NoSQL, ocupando el puesto numero 5 en el ranking de bases de datos. Esto nos proporciona suficiente seguridad de que existirá soporte para las aplicaciones que hagan uso de MongoDB por los proximos años.

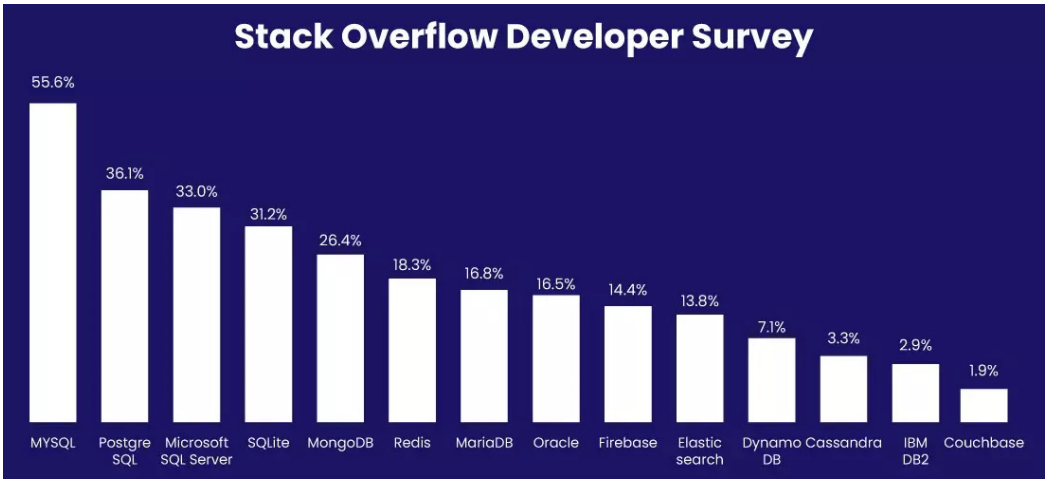

Figura 3.2: *Ranking de bases de datos mas populares de 2022*

Algunas de las características más importantes de MongoDB son:

- Consultas ad-hoc optimizadas para análisis en tiempo real
- Indexación adecuada para mejores ejecuciones de consultas
- Replicación para una mejor disponibilidad y estabilidad de los datos
- Fragmentación
- Balanceo de carga

Para realizar operaciones en la base de datos MongoDB es necesario de instalar un MongoDB Driver compatible con NodeJS, el cual permite establecer una conexión con la base de datos, y 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.

```
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 usa 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 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 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 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 pagina. Cuando una página es cargada en el explorador,

su código javascript se ejecuta y hace la página totalmente interactiva. A este proceso de 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 creado con el objetivo de proporcionar una forma sencilla a los desarrolladores de aplicar los principios de Material Design [1] 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 tambien destaca por su gran cantidad de contribuyentes, lo que asegura la longevidad y mantenibilidad del framework.

Entre las principales caracteristicas que posee Material UI se encuentran:

- 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.
- Construido usando la estrategia móvil primero, en la cual se crea el código primero para los dispotivos móviles, y despues se escalan usando las reglas de CSS necesarias.
- 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 monstrará 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>
<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 enfasis que se le quiera dar al botón. Además, tambien 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 comunmente se encuentra envuelto en un compontente 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 customizar los sliders de mútiples formas, tales como usar valores discretos, cambiar el tamaño del slider, o utilizar marcas personalizadas para el slider, lo que permite dar contexto al usuario sobre los valores mostrados en 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.

TODO: 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.

4.4.1. Toolforge

- 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`

4.4.2. Histropedia Timeline

- URL del proyecto `http://histropedia.com/timeline/`

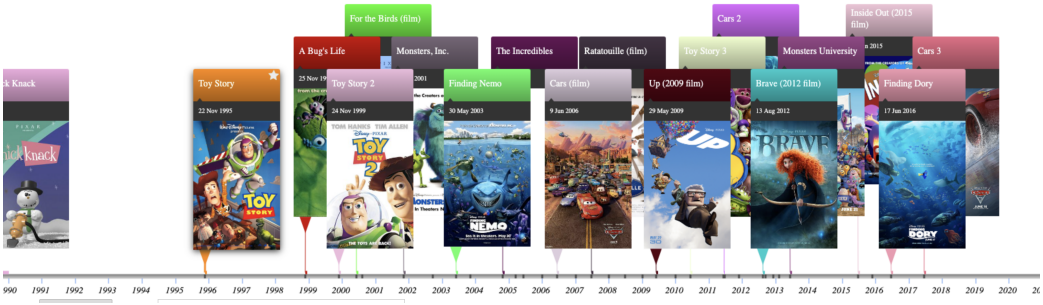

Figura 4.1: *Histropedia Timeline*

`https://en.wikipedia.org/wiki/Wikipedia:Tools`

`apersonbot.toolforge.org` Este set de herramientas esta enfocado en supervisar usuarios y sus contribuciones a la wikipedia

- Candidate Search - Aadminscore - Articles for Creation Review History

4.4.3. xtools

- URL del proyecto `https://xtools.wmflabs.org/`

4.4.4. Sigma - ArticleInfo

- URL del proyecto `https://sigma.toolforge.org/articleinfo.py`

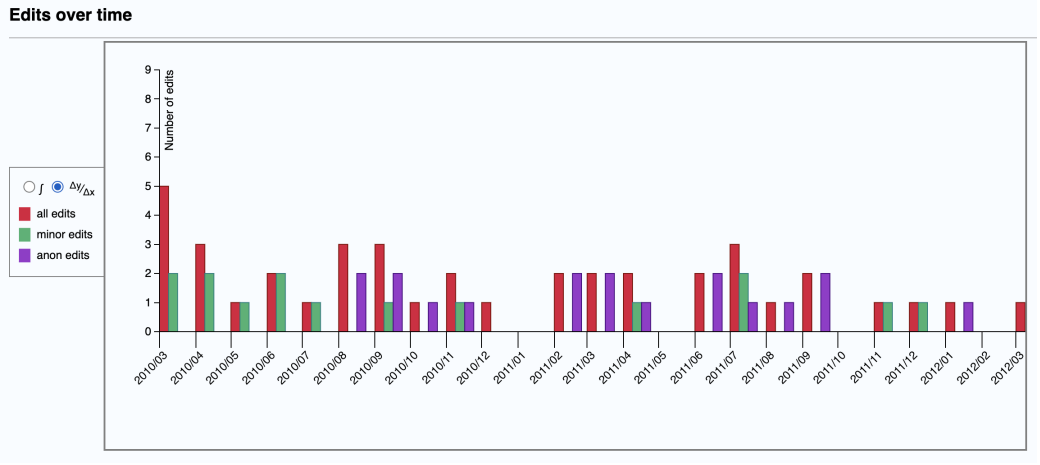

Figura 4.2: *Sigma Article Info*

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.

Figura 4.3: *Interfaz de wikireplay*

`https://de.wikipedia.org/wiki/Benutzer:Atlasowa/edit_history_visualization`

4.4.5. wikihistoryflow

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

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

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)
- MongoDB
-

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 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 aplicacion web.
6. La aplicacion tiene que contar con una vista principal que incluya grafica e informacion general de cada uno de los wiki del watchlist.

7. Permitir al usuario crear una visualizacion 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 4.4: *Prototipo de página de perfil de watcher*

4.7.2. Página principal \home

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

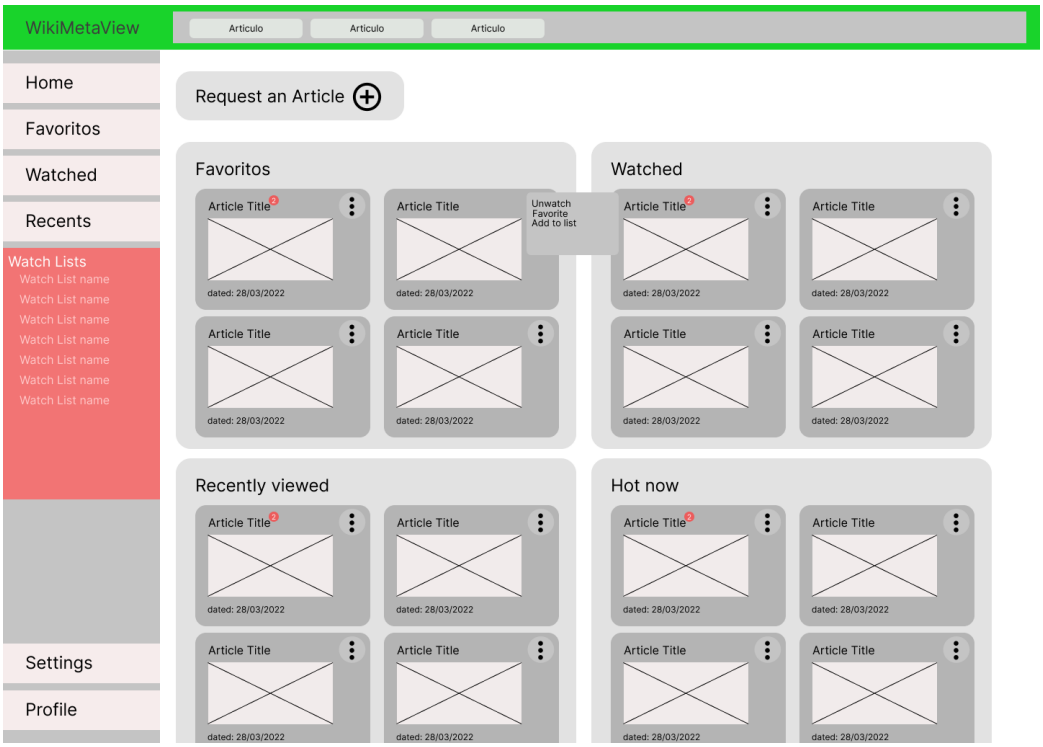

Figura 4.5: *Prototipo de vista principal de un usuario autenticado*

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 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

4.7.3. 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

4.7.4. Página de configuración \settings

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

4.7.5. Página de artículo \article

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

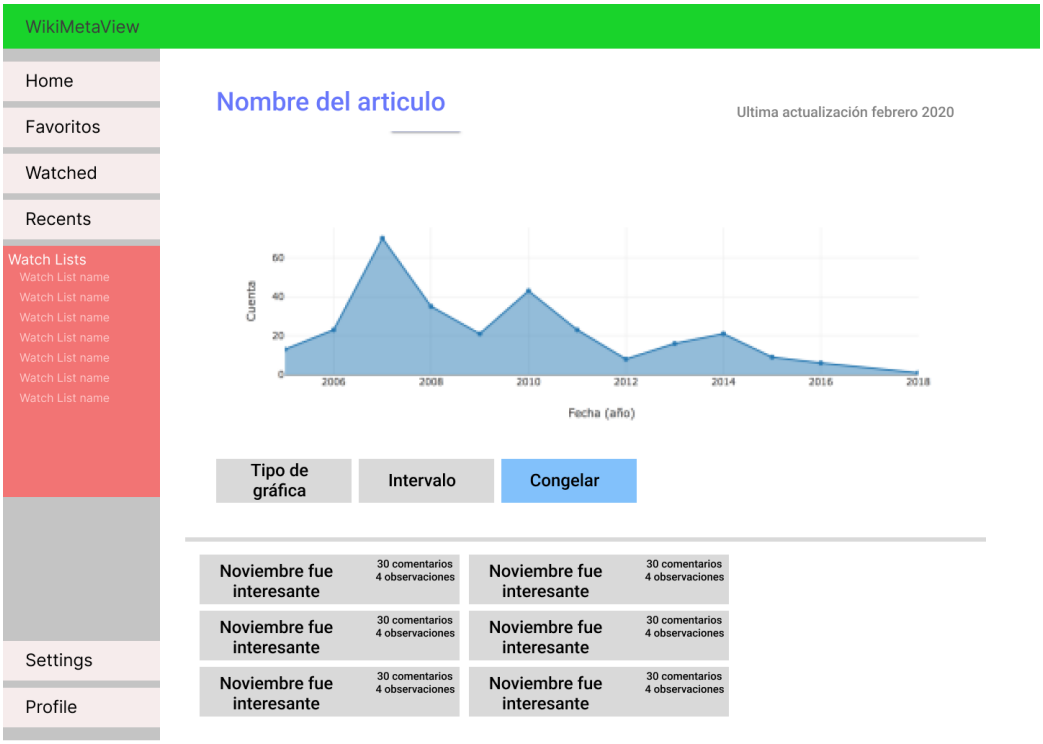

Figura 4.6: *Prototipo de página de artículos*

4.7.6. Sobre nosotros \about

Breve resumen del proyecto, incluye el documento de tesis y documento de seminario asi 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 esta en el url. Presenta la modal de autenticación

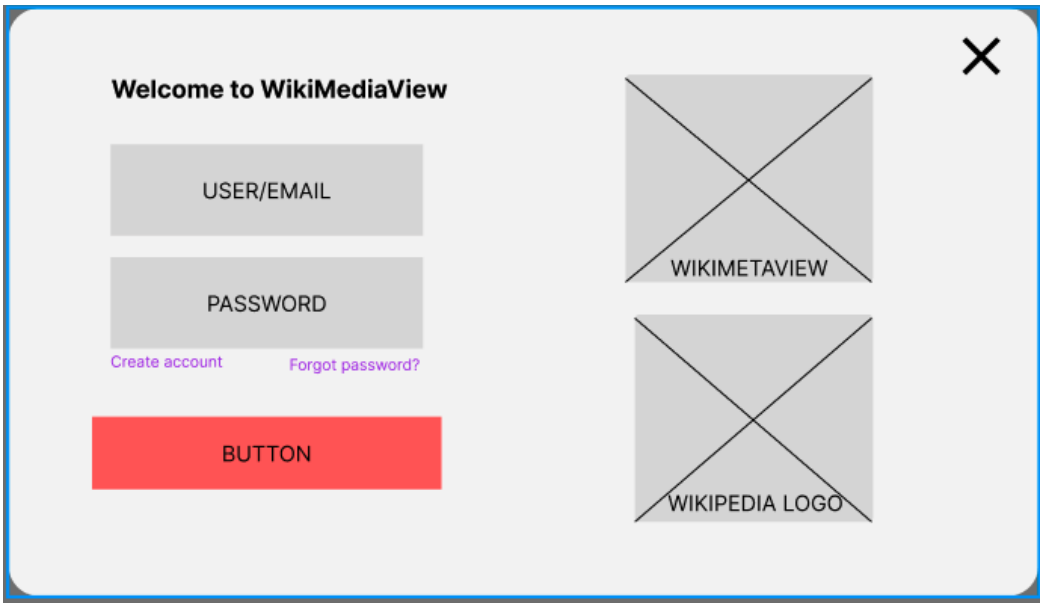

Figura 4.7: *Modal de autenticación*

?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] *Material Design Principles*. Google. URL: https://material.io/design/introduction#principles.

[2] Morville Petter. *UX Factors*. Semantic Studios. 2004. URL: http://semanticstudios.com/user_experience_design/.