
Especificación Técnica de la Aplicación

Ingeniería de Aplicaciones Web - Universidad Nacional de La Plata

Índice

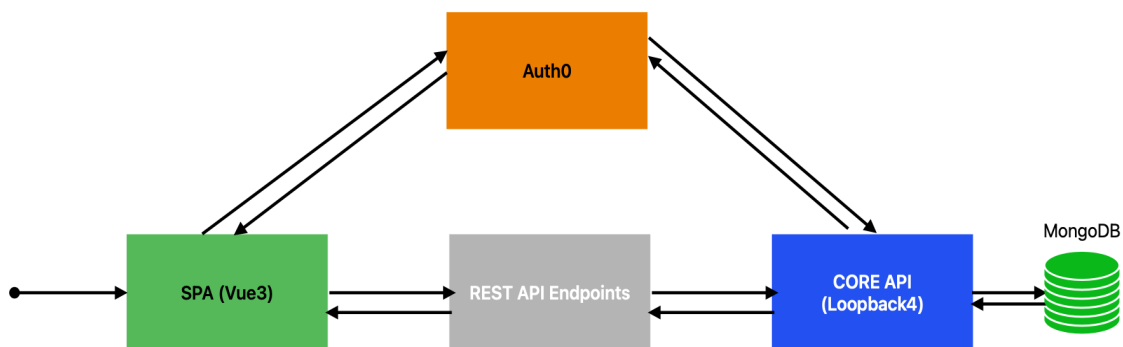
Infraestructura de la Aplicación.....	2
I. Descripción.....	2
II. Modelo de infraestructura.....	2
Modelo de Datos.....	3
I. Modelado.....	3
II. Descripción complementaria.....	3
Tecnologías Utilizadas.....	5
I. Loopback 4.....	5
II. Vue 3.....	5
III. Typescript.....	6
IV. Auth0.....	6
V. Docker.....	6
Consideraciones Técnicas.....	7
I. Identificación de los sitios que pertenecen a un usuario.....	7
II. UUIDS.....	7
III. Pérdida de autenticación en navegador Safari (cookies de terceros).....	7
IV. Servicio de verificación de estado de conexión (ping).....	8
V. Diseño del Crawler.....	8
VI. Frecuencia de ejecución del algoritmo “crawler”.....	9
VII. Implementación algorítmica.....	9
1. El método doWork.....	9
2. El algoritmo de scraping y la librería Cheerio.....	9
Ejecución de la Aplicación.....	11
I. Acceso a la aplicación:.....	11
Casos de Uso.....	12
I. Login.....	12
II. Crear Sitio.....	12
III. Listar Sitios.....	13
IV. Listado de Snapshots.....	13
Código Fuente del Proyecto.....	14

Infraestructura de la Aplicación

I. Descripción

El sistema consta de una API REST desarrollada en Loopback 4, una SPA construida en Vue 3 y para la persistencia, se optó por utilizar MongoDB. Además la aplicación se encuentra totalmente dockerizada para facilitar su despliegue.

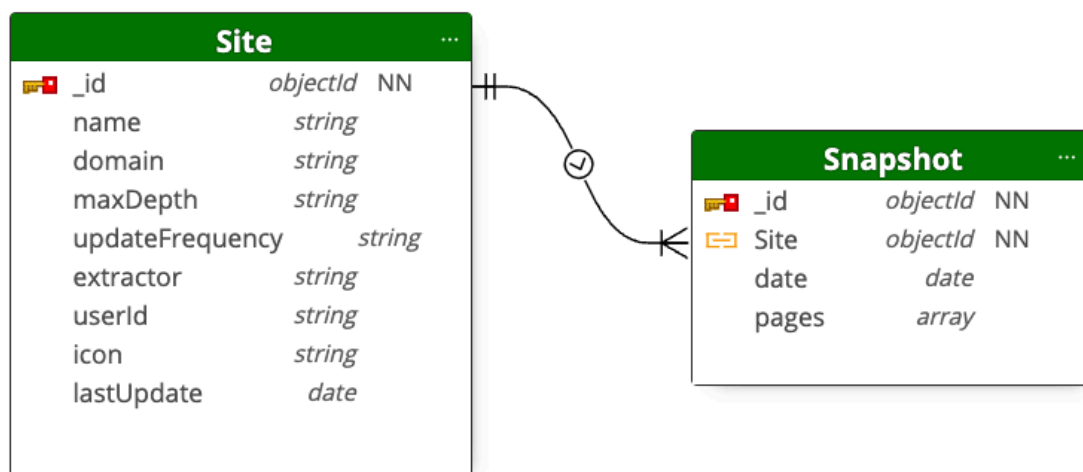
II. Modelo de infraestructura



Modelo de Datos

Como se mencionó anteriormente, para la persistencia se utilizó MongoDB como DBMS. En el siguiente gráfico se especifica el modelado NoSQL.

I. Modelado



II. Descripción complementaria

- **domain:** representa el dominio del sitio.
- **maxDepth:** representa la profundidad en páginas que el algoritmo "crawler" buscará a partir del dominio base.
- **icon:** almacena un string que representa un color, por ejemplo "red", "blue", etc. El color es generado de manera aleatoria cuando un sitio es creado.
- **lastUpdate:** representa la fecha de la última vez que el sitio fue actualizado, es decir se capturó una snapshot del mismo.
- **pages:** es un arreglo de objetos que representa la información que se recopiló en cada página (cada link visitado).

III. Ejemplo concreto de la base de datos

Colección "Site"

```
{
  "_id": "1816019d-f764-452b-977c-c9f1b78a3d1b",
  "name": "Google",
  "domain": "http://google.com.ar",
  "maxDepth": "2",
  "updateFrequency": "Desactivado",
  "extractor": "(cheerio) => {\n return {title:
cheerio(\"title\").text()} \n}",
  "userId": "google-oauth2|111108902090947385878",
  "icon": "orange",
  "lastUpdate": {
    "$date": "2024-01-09T16:12:10.218Z"
  }
}
```

Colección "Snapshot"

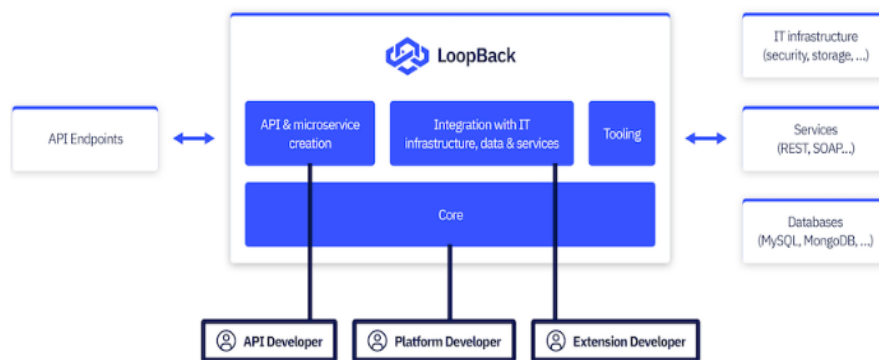
```
{
  "_id": {
    "$oid": "659d705a67475f5298828b52"
  },
  "date": {
    "$date": "2024-01-09T16:12:10.212Z"
  },
  "siteId": "1816019d-f764-452b-977c-c9f1b78a3d1b",
  "pages": [
    {
      "title": "Google",
      "url": "http://google.com.ar"
    },
    {
      "title": "Imágenes de Google",
      "url": "https://www.google.com.ar/imghp?hl=es-419&tab=wi"
    },
    {
      "title": " Google Maps ",
      "url": "http://maps.google.com.ar/maps?hl=es-419&tab=w1"
    }
  ]
}
```

Tecnologías Utilizadas

Especificación de tecnologías utilizadas por cada componente de la aplicación:

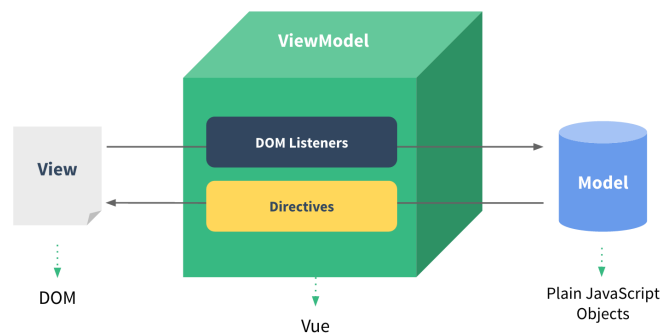
I. Loopback 4

[LoopBack](#) es un framework de Node.js altamente extensible y de código abierto basado en Express que permite crear rápidamente API y microservicios compuestos desde el backend sistemas como bases de datos y servicios REST.



II. Vue 3

[Vue JS](#) es un framework de JavaScript para el desarrollo de interfaces de usuario y aplicaciones de una sola página (SPAs). La aplicación utiliza su versión 3 y se hizo uso principalmente de [Composition API](#) para el desarrollo.

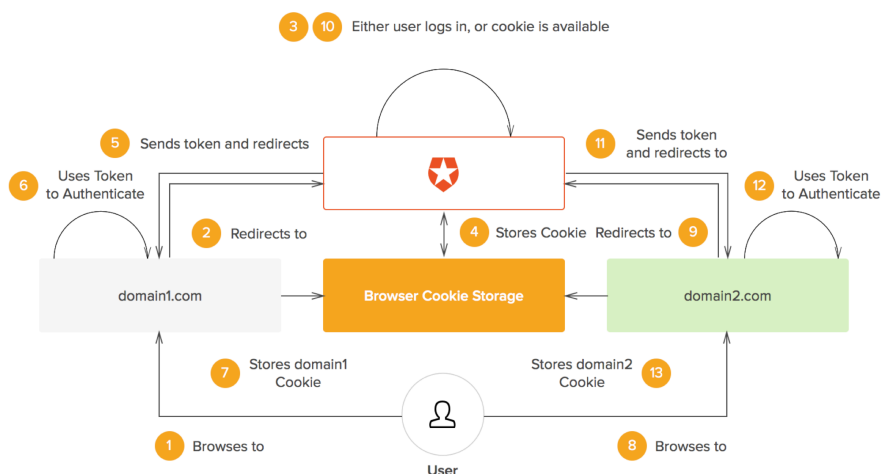


III. Typescript

[Typescript](#) es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Se decidió hacer uso de esta tecnología ya que al aportar tipos reduce sustancialmente la frecuencia de errores en tiempo de ejecución y compilación. Esto aporta seguridad y robustez al código, agiliza el desarrollo y facilita el debugging. Tanto en el frontend como en el backend se utilizó typescript.

IV. Auth0

[Auth0](#) proporciona una plataforma para autenticar, autorizar y proteger el acceso para aplicaciones, dispositivos y usuarios. En esta aplicación se hace uso de esta tecnología para facilitar el acceso de usuarios a la misma, así como también para agilizar el desarrollo delegando la administración de usuarios a un tercero (Google en este caso).



V. Docker

[Docker](#) es la tecnología en contenedores que permite crear y usar contenedores Linux. Automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Todos los componentes de software de esta aplicación se encuentran en un clúster de contenedores, haciendo uso de [Docker Compose](#).

Consideraciones Técnicas

En este apartado se detallan algunos aspectos técnicos que tienen cierta relevancia así como también ciertas decisiones tomadas.

I. Identificación de los sitios que pertenecen a un usuario

Cada sitio, como ya se especificó en el Modelo de Datos, posee un siteld. Dado que la base de datos de la aplicación no es responsable de persistir los usuarios como tales (ya que esta tarea ha sido delegada en un servicio de terceros), es necesario el uso de alguna referencia para vincular un sitio con un usuario. Se optó por utilizar el campo "sub". En Auth0, el campo "sub" (sujeto) se utiliza para representar de manera única al usuario. Cuando un sitio es creado, se extrae del token que recibe el backend su sujeto y se vincula con el sitio.

II. UUIDS

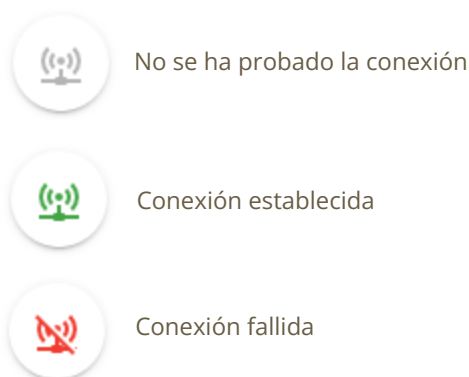
El sistema hace uso de UUIDS para la identificación de sus entidades con el objetivo de mitigar riesgos de seguridad debido a la predecibilidad como característica intrínseca de los ObjectIDs que provee MongoDB.

III. Pérdida de autenticación en navegador Safari (cookies de terceros)

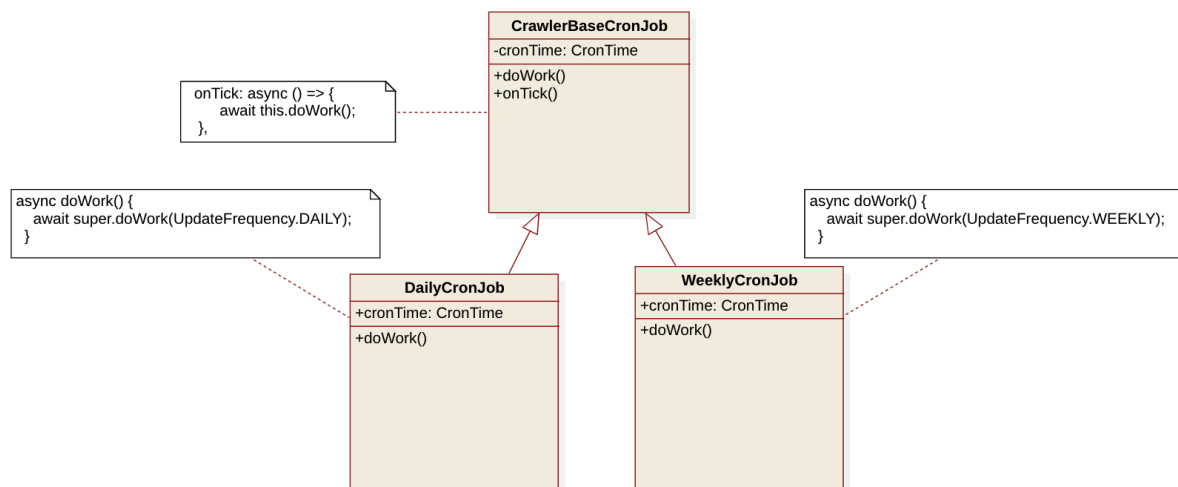
Una problemática a resolver es que en el navegador Safari, cada vez que se recarga la página, se muestra el prompt de auth0 nuevamente para iniciar sesión, es decir la misma se pierde. El problema se debe a que Auth0 hace uso de cookies de terceros y por otro lado Safari, en sus últimas versiones establece como política, el bloqueo por defecto de las mismas. Si bien los datos de la sesión además quedan persistidos en la Store de Piña y transitivamente en el LocalStorage del navegador, el problema persiste debido a que AuthGuard de auth0 no hace uso de la Store. Una posible solución sería crear una guarda personalizada que utilice los datos del LocalStorage.

IV. Servicio de verificación de estado de conexión (ping)

Este servicio que ofrece el backend en Loopback, permite determinar si el dominio ingresado en el formulario de la creación ha respondido satisfactoriamente a un ping (es decir se pudo establecer una conexión). El servicio está implementado en el `ServiceStatusController` y su url es <http://localhost:3000/api/ping/{url}>



V. Diseño del Crawler



El diseño utilizado, busca aportar cierta flexibilidad y adaptabilidad a los cambios. Por ejemplo, si se desea agregar una nueva frecuencia de actualización, lo único que necesitamos hacer es agregar una nueva clase por ejemplo "MonthlyCronJob", redefinir su variable de instancia `cronTime` y llamar al método `doWork()` heredado con `UpdateFrequency.MONTHLY`

VI. Frecuencia de ejecución del algoritmo “crawler”

Las frecuencias preestablecidas del algoritmo son una vez por día, y una vez por semana. Sin embargo, debido a fines prácticos y para que resulte fácil probar el sistema, las frecuencias implementadas son: diaria cada 45 segundos y semanal cada 60 segundos. Esto se puede modificar fácilmente en la variable `cronTime` de cada subclase.

VII. Implementación algorítmica

1. El método *doWork*

Cada job concreto (`DailyCronJob` y `WeeklyCronJob`) invocan a este método con el parámetro `UpdateFrequency.SuFrecuencia`. El método `doWork` solamente procesará los sitios cuya frecuencia coincide con la que fue enviada como parámetro.

```
const sites: Site[] = await this.siteRepository.find({
  where: {updateFrequency: updateFrequency},
});
```

Una vez filtrados los sitios que deben ser procesados en ese instante de tiempo, por cada sitio, se invoca al método **`processSite()`**, el cual se encargará de hacer scraping. Una vez terminado el scraping, se devuelve el control a `doWork()`, el mismo recibe ya las páginas extraídas por el crawler, crea una `Snapshot` con las mismas y la persiste.

2. El algoritmo de scraping y la librería *Cheerio*

El método **`processSite()`** es quien invoca al **`scrapeRecursive()`**. El primero define las siguientes variables que son importantes mencionar:

```
const pages: [] = []; // Almacenará la información de las páginas extraídas
const startUrl = site.domain; // URL inicial para el scraping
const depth = parseInt(site.maxDepth, 10); // Profundidad de búsqueda
const extractor = site.extractor;
await scrapeRecursive(startUrl, depth);
```

El método **scrapeRecursive()**:

```

async function scrapeRecursive(currentUrl: string, currentDepth: number) {
  if (currentDepth > 0) { // Caso Base de la recursividad
    const $ = await loadPage(currentUrl); // Se carga la página - $ contendrá el DOM
    const page = evaluateExtractor($); // Se aplica la función en el extractor al DOM ($)
    page.url = currentUrl;
    pages.push(page as never);

    /*
      Guardamos cada link <a href> encontrado en el DOM ($)
      Esto nos sirve para posteriormente visitar los links
      anidados. (si la profundidad recibida así lo especificara)
    */
    const links: string[] = [];
    $('a').each((i, element) => {
      const href = $(element).attr('href');
      if (href) {
        links.push(href);
      }
    });

    /*
      Creamos un link absoluto por cada link extraído.
      Desencadenamos un hilo de recursión por cada link y
      restamos en uno la profundidad actual para que en
      algún momento se alcance el caso base
    */
    for (const link of links) {
      const absoluteLink = new URL(link, currentUrl).href;
      await scrapeRecursive(absoluteLink, currentDepth - 1);
    }
  }
}

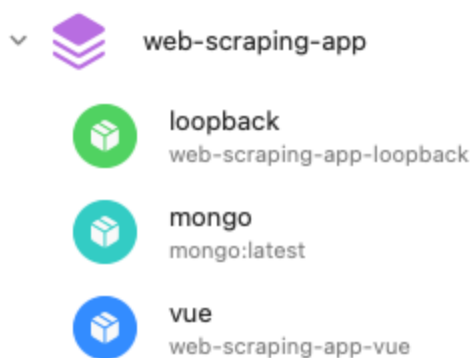
async function loadPage(url: string) {
  try {
    const response = await axios.get(url, {
      responseEncoding: 'latin1',
    });
    return cheerio.load(response.data); // Devuelve el DOM visitado
  } catch (error) {
    console.error('Cannot load url', url, 'Error:', error.message);
    return cheerio.load('');
  }
}

```

Ejecución de la Aplicación

La aplicación está completamente Dockerizada, por lo tanto lo único que se requiere es ubicarse en el directorio web-scraping-app y ejecutar el comando `docker compose up -d`

Se creará un clúster de contenedores que debería verse de la siguiente manera:



I. Acceso a la aplicación:


- Sitio: <http://localhost:5173/>
- API Docs: <http://localhost:3000/api/explorer/>




Casos de Uso

Se muestra un breve roadmap de cómo interactuar con la aplicación

I. Login

WEB SCRAPING PORTAL




¿Listo para comenzar?

Inicia sesión en tan solo un click

ACCEDER



Web Scraping Portal

Ingresá a nuestro portal

Email address


Password

[¿Olvidaste tu contraseña?](#)


Continue

No tenés una cuenta? [Registrate con Google](#)


OR

 Continue with Google

II. Crear Sitio

 **INFORMACIÓN DEL SITIO**

Nombre del sitio
Google

Dominio del sitio
http://google.com.ar 

Profundidad de búsqueda
2



Frecuencia de actualización
Diaria

```

javascript
(cheerio) => {
  return {title: cheerio("title")}
}

```

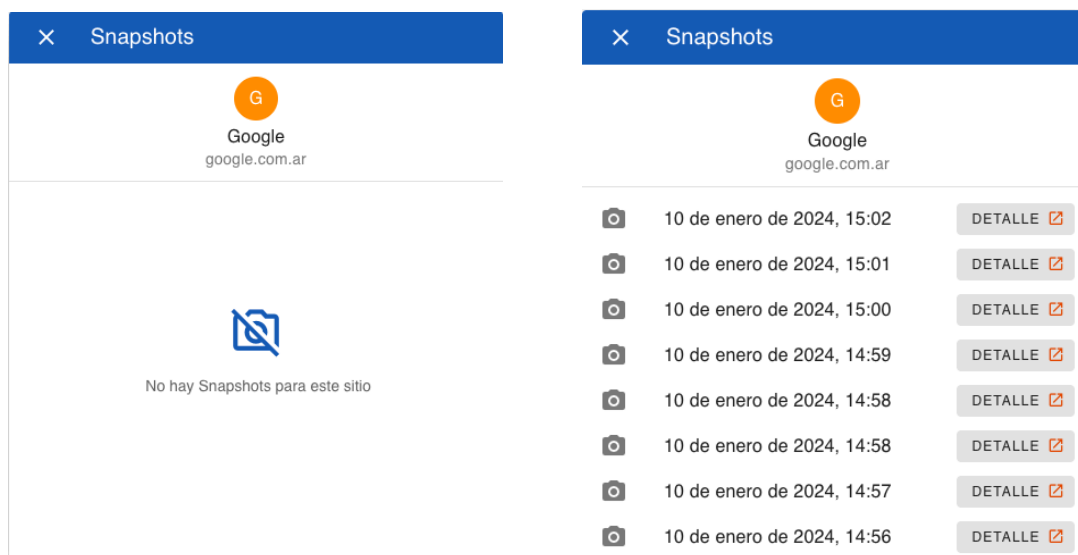
copy

LIMPIAR  **ACEPTAR** 

III. Listar Sitios



IV. Listado de Snapshots



V. Detalle de Snapshot

× Detalle de Snapshot →			
Página	Clave	URL Explorada	Información Recopilada
1	title	http://google.com.ar	Google
2	title	https://www.google.com.ar/imghp?hl=es-419&tab=wi	Imágenes de Google
3	title	http://maps.google.com.ar/maps?hl=es-419&tab=wI	Google Maps
4	title	https://play.google.com/?hl=es-419&tab=w8	Apps de Android en Google Play
5	title	https://www.youtube.com/?tab=w1	YouTube
6	title	https://news.google.com/?tab=wn	Google News
7	title	https://mail.google.com/mail/?tab=wm	Gmail
8	title	https://drive.google.com/?tab=wo	Google Drive: Sign-in

Código Fuente del Proyecto

<https://github.com/santidossantos/web-scraping>