Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.20196936
Vol.6, No. 3, Juni 2019, hlm. 337-334 p-ISSN: 2355-7699
Akreditasi KEMRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

IMPLEMENTASI WEB PUSH NOTIFICATION PADA SISTEM INFORMASI MANAJEMEN ARSIP MENGGUNAKAN PUSHJS

Alam Rahmatulloh[1], Andi Nur Rachman[2], Fahmi Anwar[3]

[1,2,3]Jurusan Informatika, Fakultas Teknik, Universitas Siliwangi
Email: [1]alam@unsil.ac.id, [2]andy.rachman@unsil.ac.id, [3]fahwar95@gmail.com

(Naskah masuk: 17 Juli 2018, diterima untuk diterbitkan: 13 Februari 2019)

Abstrak

Teknologi terus menerus berkembang, berbagai jenis teknologi terus bermunculan seperti sistem informasi manajemen arsip, masalahnya para pekerja kadang melakukan pekerjaan lain di komputer sehingga arsip tidak terkontrol. Penerapan *Web Push Notification* dapat menampilkan pemberitahuan berbasis *website* meskipun tidak membuka *web browser* secara langsung atau dalam kondisi *minimize*. *Web Push Notification* merupakan mekanisme pemberitahuan menggunakan *Javascript* pada *web browser*. Fitur ini tersedia dalam *Push API HTML5* dengan menggunakan *Push Service* atau *Messaging server* yang mengirim pemberitahuan ke *web browser* yang telah berlangganan tanpa membuka *website* sehingga dapat melakukan *broadcast message* dan *Notification API HTML5* tidak memerlukan *Push Service* atau *Messaging server* tetapi harus membuka *website*, tetapi belum didukung semua *web browser* sehingga pada makalah ini dibahas Implementasi *Web Push Notification* pada sistem informasi manajemen arsip menggunakan *PushJS*, metode pengembangan yang digunakan adalah *Rational Unified Proccess (RUP)*. Teknologi pemberitahuan yang cocok untuk sistem informasi manajemen arsip berbasis *web* yaitu *Notification API HTML5* karena tidak akan mengirim pemberitahuan yang sama ke semua pengguna. Namun tidak ada proses di belakang layar sehingga tidak akan dijalankan secara otomatis, masalah tersebut diatasi dengan menggunakan *AJAX* dengan mengambil *JSON* kemudian dijalankan berulang-ulang pada *web browser* dan meminimalisir bentrokan antara *script web push notification* di *multi tab window* atau *window web browser* diatasi menggunakan *localStorage* dari *WebStorage API HTML5*. Hasil uji menunjukan bahwa penerapan teknologi *Web Push Notification* pada Sistem Informasi Manajemen Arsip dapat membantu para pengguna dalam mengelola arsip yang banyak serta penggunaan *AJAX* berpengaruh terhadap kecepatan akses web.

Kata kunci: *ajax, json, notification api html5, pushjs, webstorage api html5*

APPLIED WEB PUSH NOTIFICATION USES PUSH JS IN THE ARCHIVE MANAGEMENT INFORMATION SYSTEM

Abstract

Technology continues to evolve, various types of technology continue to emerge such as records management information systems, the problem is that workers sometimes do other work on the computer so that the archive is not controlled. Web Push Notification application can display website-based notifications even if you don't open the web browser directly or in a minimized condition. Web Push Notification is a notification mechanism using Javascript in a web browser. This feature is available in the HTML5 Push API by using a Push Service or Messaging server that sends notifications to subscribed web browsers without opening the website so that it can broadcast and the HTML5 Notification API does not require a Push Service or Messaging server but must open a website, but not supported all web browsers so that this paper discusses Push Notification Web Implementation in archive management information systems using PushJS, the development method used is the Rational Unified Process (RUP). Notification technology that is suitable for web-based archive management information systems namely HTML5 Notification API because it will not send the same notification to all users. But there is no process behind the scenes so that it will not be run automatically, the problem is overcome by using AJAX by retrieving JSON and then running repeatedly on the web browser and minimizing clashes between web push notification scripts on multi tab windows or web browser windows resolved using localStorage from the HTML5 WebStorage API. The test results show that the application of Web Push Notification technology in the Archive Management Information System can help users manage many archives and use AJAX influences the speed of web access.

Keywords: *ajax, json, notification api html5, pushjs, webstorage api html5*

1. PENDAHULUAN

Pada umumnya sebuah aplikasi berbasis web tidak mempunyai fasilitas notifikasi atau pemberitahuan terbaru kepada pengguna, sehingga proses penyampaian informasi terutama informasi penting tidak diketahui secara langsung. Hal tersebut tentunya menjadi permasalahan, terutama pada sistem informasi manajemen arsip yang memerlukan penanganan yang cepat terhadap informasi arsip yang baru masuk. Jika tidak segera ditangani arsip terus bertumpuk dan semakin banyak, sementara pengguna aplikasi atau petugas tidak mungkin terus *standby* memperhatikan aplikasi manajemen arsip. Permasalahan tersebut dapat diatasi dengan menerapkan sistem notifikasi secara otomatis tanpa harus membuka atau memperhatikan aplikasi terus menerus. Teknik-teknik dalam penyajian data secara *realtime* yang telah dilakukan pada aplikasi web diantaranya dapat menggunakan *Asynchronous Javascript and XML (AJAX)* dan *Web Socket* (Husen, 2018).

Bentuk pemberitahuan saat ini yang sedang tren yaitu pemberitahuan pada *header* halaman *web* (Bahtiar & Mulwinda, 2016) dengan menggunakan AJAX atau sejenisnya namun pemberitahuan ini hanya dapat terlihat ketika membuka halaman web secara langsung. Pemberitahuan pada perangkat Android menggunakan Google *Cloud Messaging* (GCM) dengan menerapkan *Calendar Provider*. *Calendar Provider merupakan* jenis data yang dapat digunakan pada aplikasi kalender di perangkat Android (Setiawan, 2015) namun jenis pemberitahuan ini hanya berjalan pada perangkat Android dan GCM jika diterapkan pada web hanya dapat digunakan pada Push API dengan jenis pesan yang dikirim secara masal apabila telah berlangganan. Pemberitahuan pada perangkat Android dengan konsep *alarm* jadwal perkuliahan (Ramadhan & Utomo, 2014). Pemberitahuan monitoring menggunakan *SMS Gateway* (Fiade, et al., 2013) atau sistem pemilihan umum menggunakan SMS Gateway (Satrio, dkk., 2018) namun pemberitahuannya menggunakan biaya *SMS*. Adapun pemberitahuan menggunakan *email* (Asri, 2014).

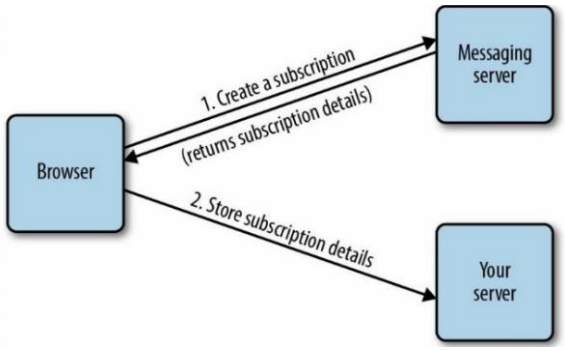
Sementara pada penelitian Rosidin menyatakan bahwa*Web Push Notification* merupakan teknologi pemberitahuan yang harus diterapkan pada *web* masa kini (Rosidin, 2017). Sehingga fokus penelitian ini, untukmendukung teknologi informasi dalam pengiriman informasi pemberitahuan yang terkontrol dibutuhkan implementasi *Web Push Notification* pada Sistem Informasi Manajemen Arsip menggunakan PushJS, karena fitur teknologi *Push API HTML5* dan *Notification API HTML5* belum didukung oleh semua *web browser* sehingga menggunakan PushJS sebagai *library* pada lintas platform (Nickerson, 2017).

2. *WEB PUSH NOTIFICATION*

Web Push Notification merupakan pemberitahuan yang dapat dikirim ke pengguna melalui *web desktop* dan *web* seluler. Pemberitahuan ini merupakan pesan gaya lansiran yang tampil pada sudut kanan atas atau bawah layar *desktop*, tergantung pada sistem operasi, atau muncul di perangkat seluler dengan cara yang hampir identik dengan *Push Notification* yang dikirim dari aplikasi. *Web Push Notification* dikirimkan di *desktop* atau layar seluler pengguna kapan pun ketika *web browser* dijalankan meskipun pengguna membuka halaman *web* atau tidak (Urbanairship, 2018).

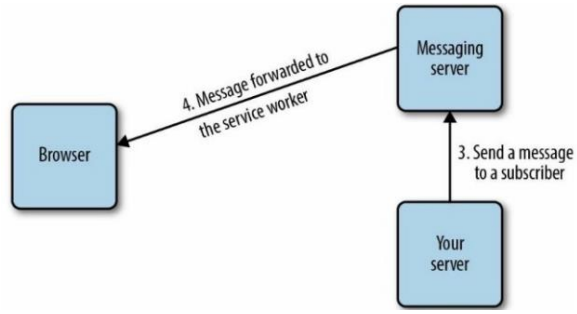
2.1 *Push APIHTML5*

Push APIHTML5 merupakan jenis notifikasi pada *web browser* yang menggunakan *Push Service* atau *Messaging server* sebagai tempat *broadcast messaging* dengan syarat *web browser* telah berlangganan (Ater, 2017). *Push APIHTML5* memungkinkan pengiriman pesan *push* ke *webapp* melalui layanan *push*. *Server* dapat mengirim pesan *push* kapan saja, bahkan ketika *webapp* atau *user agent* tidak aktif. Layanan push memastikan pengiriman yang andal dan efisien kepada agen pengguna. Pesan *push* dikirim ke *Service Worker* yang berjalan di *webapp*, yang dapat menggunakan informasi dalam pesan untuk memperbarui status lokal atau menampilkan pemberitahuan kepada pengguna. Push API memungkinkan *webapp* untuk berkomunikasi dengan *user agent* secara asinkron. Ini memungkinkan server aplikasi untuk memberikan informasi kapan tanpa membuka *webapp* (W3C, 2017).



Gambar 1. Alur Berlangganan Push API (Ater, 2017)

Gambar 1 merupakan alur dalam penggunaan pertama *Push API* dengan memanggil *subscriber()*. Pusat server perngiriman pesan akan menyimpan rincian langganan baru ini dan mengembalikan ke halaman, selanjutnya halaman dapat mengirim rincian berlangganan ke *server* dimana data disimpan untuk digunakan.



Gambar 2. Alur Pengiriman pesan Push API (Ater, 2017)

Gambar 2 merupakan alur pengiriman *Push API* dengan memanfaatkan rincian langganan dengan mengirim pesan ke *message server* kemudian meneruskan ke *web browser* pengguna, pada saat itu *ServiceWorker* bekerja apabila telah terdaftar di *web browser*.

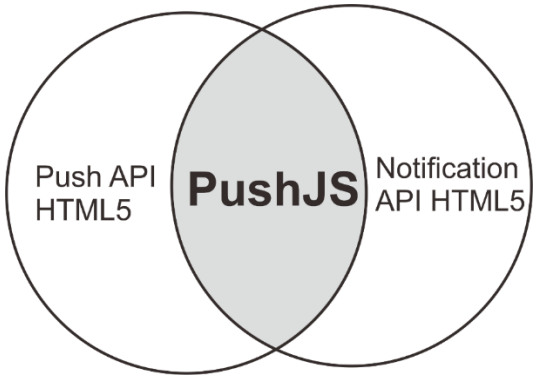
2.2 *Notification API HTML5*

Notification API HTML5 memungkinkan halaman web atau pekerja layanan (*service worker*) untuk membuat dan mengontrol tampilan pemberitahuan sistem. Pemberitahuan ini ditampilkan di luar web browser (di antarmuka perangkat) dan dengan demikian ada di luar konteks jendela atau *tab web browser* tunggal. Karena tidak bergantung pada jendela atau tab peramban apa pun, mereka dapat dibuat bahkan setelah pengguna meninggalkan situs. Sebelum dapat menampilkan pemberitahuan kepada pengguna, pengguna harus terlebih dahulu meminta izin pengguna. Keseluruhan prosesnya relatif sederhana dan lugas (Ater, 2017).

Notification API untuk menampilkan pemberitahuan untuk memperingatkan pengguna di luar konteks halaman web. Pemberitahuan ini mengacu pada menampilkan pemberitahuan pada "*desktop*". ini dirancang agar kompatibel dengan platform pemberitahuan yang ada sebanyak mungkin, tetapi juga menjadi platform-independen. Karena platform umum tidak menyediakan fungsi yang sama, spek ini akan menunjukkan peristiwa apa yang dijamin dan mana yang tidak. Khususnya, pemberitahuan yang ditentukan di sini hanya dapat berisi konten teks dan ikon (W3C, 2015).

2.3 *PushJS*

PushJS adalah salahsatu cara untuk membangun dan menjalankan pemberitahuan *desktop Javascript*. Tambahan yang cukup baru untuk spesifikasi resmi, *NotificationAPI* memungkinkan browser *modern* seperti Chrome, Safari, Firefox, dan IE 9+ untuk mendorong pemberitahuan ke desktop pengguna. PushJS bertindak sebagai solusi lintas peramban untuk *API* ini, *fallback* menggunakan penerapan yang lebih lama jika peramban pengguna tidak mendukung *API* baru (Nickerson, 2017).



Gambar 3. Irisan antara Push API dengan Notification API menjadi PushJS

Gambar 3 merupakan perumpamaan bahwa *PushJS* memiliki kemampuan dari *Push API HTML5* dan *Notification API HTML5*. Fungsi utama notifikasi menggunakan *Notification API HTML5* dan menggunakan *serviceWorker* untuk menggunakan teknologi *Push API HTML5* sehingga tidak langsung menggunakan fasilitas dari *Web Browser*.

2.4 *WebStorage API HTML5*

Web Storage API HTML5 atau disebut DOMStorage merupakan sebuah *API* yang membuatnya mudah untuk menyimpan data di seluruh *web*. Sebelum ada *Web Storage API*, *remote server* diperlukan untuk menyimpan data apa pun yang bertahan dengan mengirimnya bolak-balik dari klien ke *server*. Dengan munculnya *Web Storage API*, pengembang sekarang dapat menyimpan data secara langsung di sisi klien pada *web browser* untuk akses berulang di seluruh permintaan atau untuk diambil secara lama setelah menutup *browser* sepenuhnya, sehingga mengurangi lalu lintas jaringan. *Web Storage* berbeda dari *cookie* dari bagaimana cara menyimpan dan mengambil data. Dengan menggunakan *API* sederhana ini, pengembang dapat menyimpan nilai dalam objek JavaScript yang mudah diambil yang bertahan di seluruh beban halaman. Dengan menggunakan *sessionStorage* atau *localStorage*, pengembang dapat memilih untuk membiarkan nilai-nilai tersebut bertahan baik di seluruh beban halaman dalam satu jendela atau *tab* atau di seluruh *browser* saat di *restart*. Data yang tersimpan tidak ditransmisikan di seluruh jaringan dan mudah diakses pada kunjungan kembali ke halaman (Ater, 2017).

2.5 PHP *Data Objects*

PDO (*PHP Data Objects*) merupakan fitur PHP yang telah ada mulai versi 5. PDO merupakan jenis salah satu teknik penggunaan database dengan satu *query* yang sama tanpa mengubah script hanya mengatur koneksi tujuan *database engine*, karena mendukung berbagai jenis *database*, dimana untuk untuk memanggil seluruh fungsi setiap database tidak

memperlukan perubahan *script* pada fungsi koneksi ke *database*. PDO juga mendukung *SQL injection*dengan menerapkan PDO *BlindParam statement parameter* ataumasukan dari pengguna secara otomatis telah di saringterlebih dahulu sebelum diproses. PDO menggunakan teknologi *data-access abstaction layer* untuk menghungkan *script* dengan *database.*PDO memiliki beberapa keuntungan, diantaranya (Abdul, 2008):

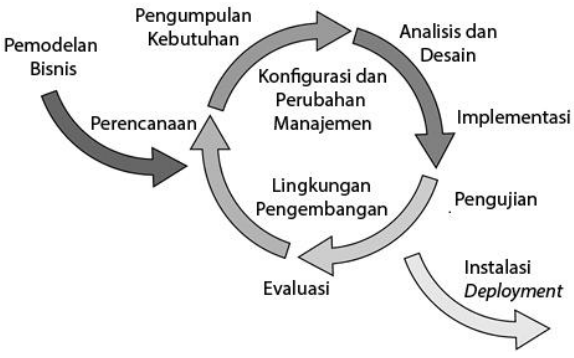
1. PDO mendukung banyak *database*, *database* yang didukung oleh PDO antara lain *MySQL*, *PostgreSQL*, Oracle, SQLite, IBM, *Firebird*, *DBLib* dan lain-lain.
2. PDO dapat melakukan migrasi *databaseengine* yang digunakan karena dalam PDO *query* setiap *database engine* tetap sama yang membedakannya hanyalah koneksi.

PDO juga memiliki beberapa kelemahan, diantaranya:

1. PDO tidak dapat menggunakan beberapa fitur canggih dari *database engine* yang digunakan, seperti dukungan eksekusi *Multiple Statements* di MySQL.
2. PDO tidak bisa menggunakan sintaks SQL yang spesifik untuk *database* tertentu, seperti CTE atau *Common Table Expression*, yang telah didukung *database*dari Oracle dan PostgreSQL.

3. METODE PENELITIAN

Metodologi pengembangan perangkat lunak pada penelitian ini menggunakan *RUP (Rational Unified Process)*mulai dari tahapan *Inception* (Permulaan), *Elaboration* (Perencanaan), *Contruction* (Kontruksi) dan *Transition* (Transisi).Proses pengulangan atau iteratif pada *RUP* secara global dapat dilihat pada Gambar 4.



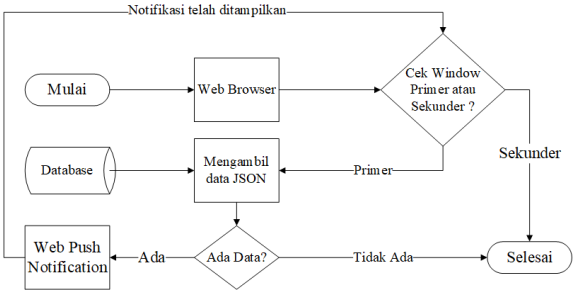
Gambar 4. Proses Iterasi RUP (Sukamto & Shalahuddin, 2013)

4. HASIL DAN PEMBAHASAN

4.1. Tahap *Inception* (Permulaan)

Proses bisnis yang akan diterapkan pada Sistem Informasi Manajemen Arsip hanya pada bagian Petugas Perpus dan Admin setelah *login*. Sistem yang diimplementasikan berupa pemberitahuan menggunakan *PushJS*, *JQuery* dan *AJAX* dengan bentuk pertukaran data menggunakan *JSON*. Pertukaran data *JSON* dapat diamankan menggunakan *JSON Web Token (JWT)* (Rahmatulloh, et al., 2018).

Gambar 5 merupakan rancangan usulan *push notification* menggunakan *PushJS* berbasis Javascript. Setelah melakukan *login*, *web browser*akan menjalankan *script* dengan mengecek status tab window atau *window web browser* primer atau sekunder, dilihat dari waktu pembuatan *tab window* atau *window* dalam satuan *milisecond* atau biasa disebut *timestamp* disimpan dalam *localStorage* menggunakan *Web Storage API HTML5*.



Gambar 5. Usulan rancangan *push notification*

Apabila status *tab window* atau *window* primer maka akan menjalankan *script* pengambilan data berbentuk *JSON* menggunakan *AJAX* jika ada maka akan ditampilkan sedangkan jika tidak ada atau sudah semua data maka akan kembali lagi ke proses pengecekan status *tab window* atau *window web browser* yang digunakan.

Aktor-aktor yang berhubungan dengan sistem informasi manajemen arsip, diantaranya: Petugas SKPD (Satuan Kerja Pemerintah Daerah), Petugas DISPERPUSKA dan Pengunjung. Kebutuhan fungsional sistem informasi manajemen arsip adalah Manajemen Arsip, Manajemen Instansi, Manajemen Pengguna, Manajemen Ruangan, Manajemen Rak, Manajemen Box, Manajemen Surat, ScannerBox dan Pencarian Arsip. Kebutuhan non fungsional sistem informasi manajemen arsip adalah sistem notifikasi pemberitahuan yang bisa memberikan informasi atau pengingat ketika tanpa harus memperhatikan aplikasi atau sedang mengerjakan aktifitas lain di komputer.

4.2. Tahap *Elaboration* (Perencanaan)

Pada tahap kedua ini proses perencanaan dimulai dari menganalisa, identifikasi dan evaluasi proses bisnis yang sedang berjalan. Sehingga akan didapatkan solusi alternatif yang mengarah kepada perbaikan dan pengembangan yang lebih baik serta sesuai dengan kebutuhan.

Sistem yang sedang berjalanharus diketahui dan dipelajari terlebih dahulu, diperlukan penggambaran aliran informasi dari berbagai bagian yang terkait

sistem. Aliran informasi yang sedang berjalan pada sistem manajemen arsip adalah sebagai berikut:

1. SKPD/Kecamatan menyerahkan Arsip untuk di dititipkan di depo arsip.
2. DIPERPUSKA menerima Arsip dari SKPD/Kecamatan yang akan ditipkan di depo arsip.
3. DIPERPUSKA menyeleksi Arsip yang akan dititipkan
4. DIPERPUSKA membuat Jadwal Resensi Arsip (JRA).

4.3. Tahap *Construction* (Kontruksi)

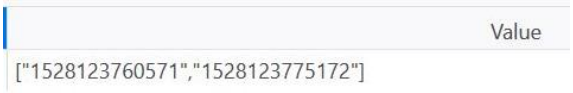
Implementasi sistem menggunakan *PushJS* ke dalam Sistem Informasi Manajemen Arsip digabungkan dengan *Javascript*, *JQuery*, *AJAX*, *WebStorage API HTML5*. Konsep pemberitahuan yang digunakan seperti *runningtext* pada *channel* televisi yang memberikan informasi terus menerus sampai informasi tersebut di ubah oleh operator. Konsep ini pula menjadi inspirasi untuk diimplementasikan pada Sistem Informasi Manajemen Arsip yang dapat memberikan pemberitahuan terus menerus sampai ditindak lanjutioleh pengguna. Pemberitahuan yang dimunculkan pada *Web Push Notification* seperti Arsip yang belum diatur JRA dan Arsip yang sudah melewati masa Inaktif harus ditindak sesuai jenis Arsip.

Konsep teknis yang digunakan *Javascript* mengecek ketersediaan *web* dibuka di *tab window* yang masa saja, kemudian script *Web Push Notification* akan berjalan pada *tab window* yang terakhir dibuka berdasarkan waktu pembuatan *tab window* menggunakan *localStorage* dan *sessionStorage* dari *Web Storage API HTML5*. Setelah itu *script Web Push Notification* yang menggunakan *PushJS* akan mengambil data berbentuk *JSON* dari *database* dengan menggunakan *AJAX* secara *realtime* dan dihitung jumlah data yang akan di munculkan pada pemberitahuan dengan nilai *timeout* atau waktu tampil yang telah ditentukan. Apabila ada perubahan dalam pemberitahuan akan di eksekusi pada saat element data yang terakhir telah dimunculkan Gambar 6 merupakan tampilan dari antarmuka *Web Push Notification*.

Gambar 6 Tampilan Antarmuka *Web Push Notification*

4.4. Tahap *Transition* (Transisi)

Sebelum proses transisi, pengujian merupakan hal terpenting yang harus dilakukan untuk mengetahui kekurangan atau kesalahan yang ada. Pengujian dilakukan dengan pengecekan secara fungsional (*black* box). Pada proses pengecekan menggunakan *localStorage* menghasilkan *timestamp* dalam format *milisecond* atau milidetik kemudian dibandingkan waktu yang paling terbaru membuat *window* atau *tab window* pada *web browser* dengan menghasilkan data berbentuk JSON seperti pada Gambar 7 bernilai [“1528123760571”,”1528123775172”] yang berarti 06-04-2018 14:49:20 dan 06-04-2018 14:49:35 pada zona *Universal TimeCoordinated (UTC)* sedangkan Asia/Jakarta ditambah 7 jam lebih cepat. Jadi, *tab window* atau *window* yang akan menjalankan *web push notification* yang dibuat pada 1528123775172 atau 06-04-2018 14:49:35 pada zona *UTC*.



Gambar 7. Isi data *localStorage* menggunakan *Inspect Element* pada *Web Browser*

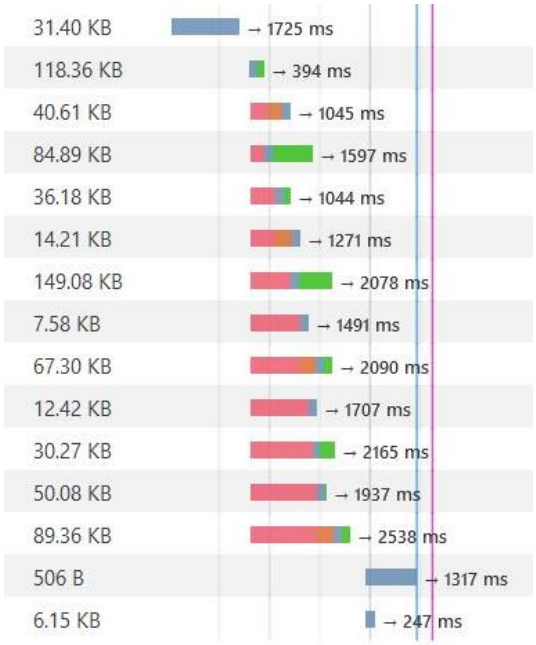
Pengujian selanjutnya uji fungsional sistem menggunakan skenario uji yang hasilnya dapat dilihat pada Tabel 1.

Tabel 1 Bagian Uji Fungsional Sistem

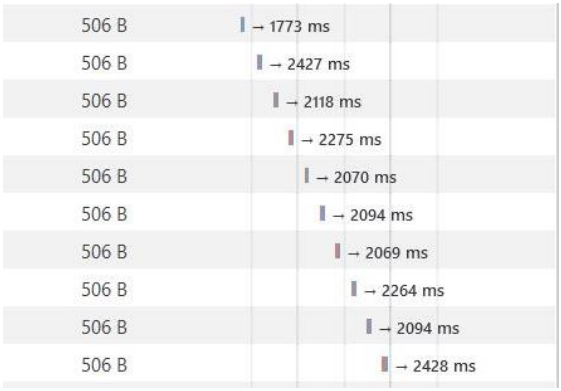
Nama Bagian	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Web Push Notification	Belum melakukan Login	Tidak menampilkan pemberitahuan	[√] Berhasil [ ] Tidak Berhasil
	Setelah Pengguna Login (Petugas Perpus)	Menampilkan pemberitahuan	[√] Berhasil [ ] Tidak Berhasil
	Setelah Pengguna Login (Petugas SKPD)	Tidak Menampilkan pemberitahuan	[√] Berhasil [ ] Tidak Berhasil
	Membuka *web* di *tab window* yang lain	Tidak terjadi bentrok *script* yang berjalan	[√] Berhasil [ ] Tidak Berhasil

Dalam memproses *script Web Push Notification*, *web browser* membutuhkan beberapa detik untuk membuka halaman setelah selesai dengan pengecekan validasi *window* selama 1.000 *ms* (*milisecond*) atau 1 detik dan menampilkan notifikasi selama 5.000 ms (*milisecond*) atau 5 detik. Jadi, jika ada 10 notifikasi maka dibutuhkan waktu 5x10 = 50 detik untuk menampilkan semua pemberitahuan. Proses tersebut belum termasuk lama proses membuka halaman *web* dan data *JSON*.

Gambar 8 menunjukan hasil analisa menggunakan *Inspect Element* pada *web browser* Mozilla Firefoxdiperoleh 1.725 *milisecond* (*ms*) atau 1,7 *second*.



Gambar 8. Tampilan Visual Lama Waktu Proses



Gambar 9. Proses Pengulangan pengambilan *JSON*

Berdasarkan Gambar 9 diperoleh 10 sampel kemudian dimasukan kedalam Tabel 2 dengan dihitung rata-rata lama waktu pemrosesan dengan karakteristik 9.500 data arsip pada *database* dengan hasil *compressing* menjadi 506 *Bytes*. Hasil rata-rata diperoleh 2.161,2 milidetik yang dapat dilihat pada Tabel 2.

Tabel 2. Uji Waktu Pemrosesan Data JSON

No	*milisecond* (*ms*)
1	1.773
2	2.427
3	2.118
4	2.275
5	2.070
6	2.094
7	2.069
8	2.264
9	2.094
10	2.428
Jumlah	21.612
Rata-rata	2.161,2

Gambar 10 *Unit Testing Web Push Notification*menggunakan QUnit

Gambar 10 merupakan hasil *unit testing Web Push Notification* menggunakan QUnit menunjukan fungsi *webpush()* pada HTML berjalan sesuai dengan harapan.

Tabel 3. Bagian Pernyataan Pengujian Beta

No.	Pernyataan
1	Kesesuaian warna tulisan dengan warna latar belakang (*background*) *pop-up* pemberitahuan
2	Tulisan pada *pop-up* pemberitahuan terbaca
3	Tampilan kotak *pop-up* pemberitahuan terlihat jelas
4	Penggunaan *pop-up* pemberitahuan mudah digunakan saat membuka aplikasi lain
5	Ketepatan lama waktu menampilkan *pop-up* pemberitahuan

Tabel 3 merupakan pernyataan yang digunakan pada pengujian *User Acceptance Testing (UAT)* pada 5 *reponden* menghasilkan persentase nilai kepuasan dalam penggunaan aplikasi yang telah diterapkan teknologi *Web Push Notification* dengan 5 pilihan seperti pada Tabel 4.

Tabel 4. Skor Jawaban Kuesioner

Skala Jawaban	Keterangan	Skor
SS	Sangat Setuju	5
S	Setuju	4
N	Netral	3
TS	Tidak Setuju	2
STS	Sangat Tidak Setuju	1

Pengujian UAT pada aplikasi setelah diterapkan teknologi Web Push Notification menghasilkan beberapa tabel berdasarkan pernyataaan, diantaranya:

1. Data Pernyataan "Kesesuaian warna tulisan dengan warna latar belakang (*background*) *pop-up* pemberitahuan" pada Tabel 5.

Tabel 5. Data Pernyataan 1

Skala Jawaban	Skor	Responden	Jumlah
Sangat Setuju	5	3	15
Setuju	4	2	8
Netral	3	0	0
Tidak Setuju	2	0	0

Skala Jawaban	Skor	Responden	Jumlah
Sangat Tidak Setuju	1	0	0
Jumlah		5	23

$$Y = \frac{23}{25} \times 100 = 92 \quad (1)$$

2. Tulisan pada *pop-up* pemberitahuan terbaca

Tabel 6. Data Pernyataan 2

Skala Jawaban	Skor	Responden	Jumlah
Sangat Setuju	5	1	5
Setuju	4	4	16
Netral	3	0	0
Tidak Setuju	2	0	0
Sangat Tidak Setuju	1	0	0
Jumlah		5	21

$$Y = \frac{21}{25} \times 100 = 84 \quad (2)$$

3. Tampilan kotak *pop-up* pemberitahuan terlihat jelas.

Tabel 7. Data Pernyataan 3

Skala Jawaban	Skor	Responden	Jumlah
Sangat Setuju	5	4	5
Setuju	4	1	16
Netral	3	0	0
Tidak Setuju	2	0	0
Sangat Tidak Setuju	1	0	0
Jumlah		5	5

$$Y = \frac{24}{25} \times 100 = 96 \quad (3)$$

4. Penggunaan pop-up pemberitahuan mudah digunakan saat membuka aplikasi lain.

Tabel 8. Data Pernyataan 4

Skala Jawaban	Skor	Responden	Jumlah
Sangat Setuju	5	3	15
Setuju	4	2	8
Netral	3	0	0
Tidak Setuju	2	0	0
Sangat Tidak Setuju	1	0	0
Jumlah		5	23

$$Y = \frac{23}{25} \times 100 = 92 \quad (4)$$

5. Ketepatan lama waktu menampilkan pop-up pemberitahuan.

Tabel 9. Data Pernyataan 5

Skala Jawaban	Skor	Responden	Jumlah
Sangat Setuju	5	0	0
Setuju	4	5	20
Netral	3	0	0
Tidak Setuju	2	0	0
Sangat Tidak Setuju	1	0	0
Jumlah		5	5

$$Y = \frac{20}{25} \times 100 = 80 \quad (5)$$

Hasil pengujian UAT berdasarkan rata-rata dari persentase persamaan (1), (2), (3), (4) dan (5) menghasilkan :

$$Y = \frac{92+84+96+92+80}{500} \times 100 = 89{,}6 \quad (6)$$

Tabel 10. Skor Jawaban Kuesioner

Skala Jawaban	Keterangan	Nilai
SS	Sangat Setuju	100%
S	Setuju	80%
N	Netral	60%
TS	Tidak Setuju	40%
STS	Sangat Tidak Setuju	20%

Persamaan (6) mengasilkan nilai 89,6 % mendekati skala Sangat Setuju sesuai dengan parameter pada Tabel 10, sehingga penerapan Web Push Notification berdasarkan pengujian UAT dengan 5 responden menghasilkan nilai Sangat Setuju dengan persentase 89,6%.

5. KESIMPULAN

Penerapan *Web Push Notification* dalam pemberitahuan pada Sistem Informasi Manajemen Arsip dapat membantu dalam proses penyampaian informasi sehingga lebih interaktif dan *realtime*. Sementara penggunaan *Web Storage API HTML5* berfungsi untuk mengecek *status tab window* pada *web browser* yang sama, sehingga *Javascript* tidak bentrok saat dijalankan pada *multi tab window*.

Sebagai bahan penelitian selanjutnya teknologi *Web Push Notificatio*n dapat dikembangkan lebih lanjut dengan fitur *input* atau *multiple choice* sehingga pengguna dapat memasukan data langsung atau memilih opsi yang ada pada notifikasi tanpa harus membuka halaman *web*.PushJS dapat dikembangkan menggunakan *serviceWorker* dengan teknologi *Push API HTML5* dengan memanfaatkan *Firebase Cloud Messaging*.

DAFTAR PUSTAKA

ABDUL, K., 2008. *Dasar Pemrograman Web Dinamis Menggunakan PHP*. 1 penyunt. Yogyakarta: Penerbit Andi.

ASRI, N. F., HAMZAH, A. & SHOLEH, M., 2014. Nagios untuk Monitoring Server dengan Pengiriman Notifikasi Gangguan Server menggunakan Email dan SMS Gateway (Studi Kasus: PT. Gamatechno Indonesia - Yogyakarta). *Jurnal JARKOM*, 1(2), pp. 151-161.

ATER, T., 2017. *Building Progressive Web Apps.* 1 penyunt. Sebastopol: O'Reilly Media.

BAHTIAR, M. R. & MULWINDA, A., 2016. Pengembangan Fitur Notifikasi pada Website Application Comic Strip rupi.co Menggunakan Metode Agile. *Jurnal Teknik Elektro*, 8(1), pp. 25-30.

FIADE, A., MULANA, A. A. & SUSENO, H. B., 2013. Aplikasi Monitoring Jaringan Berbasis Mobile Web dengan Sistem Notifikasi Berbasis SMS Gateway. *Jurnal Teknik Informatika*, 6(2).

HUSEN, H., RAHMATULLOH, A. & SULASTRI, H., 2018. Implementasi Komunikasi Full

Duplex Menggunakan Sistem Informasi Pengelolaan Anggaran Universitas ABC. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 1 4, 9(1), pp. 597-606.

NICKERSON, T., 2017. *Installing | Push v1.0*. [Online] Available at: https://pushjs.org/docs/introduction [Diakses 15 07 2018].

RAHMATULLOH, A., SULASTRI, H. & NUGROHO, R., 2018. Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*.

RAMADHAN, T. & UTOMO, V. G., 2014. Rancang Bangun Aplikasi Mobile Untuk Notifikasi Jadwal Kuliah Berbasis Android (Studi Kasus: Stmik Provisi Semarang). *Jurnal Teknologi Informasi dan Komunikasi*, 5(2), pp. 47-55.

ROSIDIN, 2017. *Manfaat Menggunakan Push Notification Untuk Website - JURNAL ROSID*. [Online] Available at: https://jurnal.rosid.net/manfaat-menggunakan-push-notification-untuk-website/ [Diakses 15 07 2018].

SATRIO, dkk., 2018. Rancang Bangun Sistem E2OV (Electronic - Election Observation and Voting) Menggunakan SMS. *Jurnal Teknologi Informasi dan Ilmu Komputer*, 5(1), pp. 1-6.

SETIAWAN, J., KRISTIANTO, E. & F., 2015. Implementasi Push Notification pada Informasi Perkuliahan dan Kegiatan Mahasiswa Berbasis Android. *Jurnal Teknik dan Ilmu Komputer*, 4(14), pp. 211-219.

SUKAMTO, R. A. & SHALAHUDDIN, M., 2013. *Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Objek)*. Bandung: BI-Obses.

Urbanairship, 2018. *Web Push Notifications Explained | Urban Airship*. [Online] Available at: https://www.urbanairship.com/web-push-notifications-explained [Diakses 15 07 2018].

W3C, 2015. *Notification API : W3C Recommendation 22 October 2015*. [Online] Available at: https://www.w3.org/TR/notifications/ [Diakses 15 07 2018].

W3C, 2017. *Push API : W3C Working Draft*. [Online] Available at: https://www.w3.org/TR/push-api/ [Diakses 15 07 2018].