IMPLEMENTASI LOCALSTORAGE PADA PEMROGRAMAN CLIENT BERBASIS JSON

Joko Triyono
Jurusan Teknik Informatika, Fakultas Teknologi Industri, IST AKPRIND Yogyakarta
Jl. Kalisahak 28, Komplek Balapan Yogyakarta 55222 Telp. 0274-563029
Email: jack@akprind.ac.id

Abstrak

Kemajuan dibidang komputasi semakin pesat di era 4.0, dimana sistem informasi yang tersebar harus bisa terkoneksi satu dengan yang lainnya, keamanan transaksi dan kecepatan transaksi menjadi hal yang penting untuk dikedepankan. Perkembangan teknologi HTML5 dengan localStorage yang bisa menampung sampai 5MB tanpa mempengaruhi browser menjadikan metodologi pemrograman bergeser, JSON sebagai bagian dari sistem ini memberikan kemudahan dalam format data dan komunikasi antar sistem informasi. Sehingga pemrograman berbasis client side menjadi pilihan yang menarik, selain kemudahan dalam develop, maka kecepatan transaksi menjadikan model pemrograman berbasis client ini makin dominan. JavaScript sebagai motor penggerak utama memiliki peran yang sangat besar, kemampuan dalam request ke server lain menggunakan ajax dan JSON secara background. Dalam penelitian ini telah diujikan dan dihasilkan bahwa proses pengembangan aplikasi berbasis client site bisa diimplementasikan, dengan pemanfaatan localStorage sebagai penampung data dalam format JSON dan sessionStorage sebagai penampung data tunggal serta kombinasi penggunaan ajax untuk komunikasi ke service berbasis server side secara tepat akan menghasilkan aplikasi client side tidak kalah dengan aplikasi server side. Juga dalam penggunaan data atau record bersama, dengan design rdbms yang tepat maka proses penguncian record bisa dilakukan.

Kata kunci: ajax, javaScript, JSON, localStorage, sessionStorage

Pendahuluan

Kemajuan komputasi menjadi sangat diperlukan saaat memasuki era 4.0, dimana sistem informasi yang tersebar harus bisa saling berkoneksi satu sama lain, efektifitas penggunaan media komunikasi menjadi salah satu point untuk meningkatkan kemampuan komunikasi antar sistem informasi maupun user dengan sistem informasi. Sesuai dengan perkembangan dan tuntutan jaman, teknologi sistem informasi baik itu *software* dan *hardware* mengalami perkembangan bahkan bisa dikatakan mengalami lompatan kemajuan yang sangat pesat. Dengan perkembangan yang seperti itu maka tentunya metodologi dalam pengembangan sistem informasi juga berubah. HTML5 dengan fitur-fitur barunya yang terkait dengan komputasi programming telah mengembangkan kemampuan mulai dari *Cookies*, *sessionStorage* dan *localStorage*. Ketiganya memiliki fungsi yang hampir sama dengan perbedaan bahwa *localStorage* bisa menampung data lebih dari 5MB tanpa membebani *performa browser*, sehingga memudahkan pengembang web dalam membuat web yang lebih *responsive* pada sisi klien. Sedangkan perbedaan antara *sessionStorage* dan *localStorage* adalah bahwa *sessionStorage* data yang tersiman akan hilang ketika *browser* ditutup sedangkan *localStorage* tidak memiliki waktu *expired*. (Deswitansyah, 2018). JSON sebagai teknologi baru yang mengiringi munculnya HTML5 memiliki kemampuan transfer data yang sangat luar biasa, **JSON** (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. (Crockford), teknologi API telah menjadi sangat popular saat ini. API menjadi bagian penting dalam menghubungkan antara *backend* dengan *frontend* dan *mobile*. Kualitas API bisa dilihat dari *response body, status code, error message* yang dihasilkan. (Rohman, 2017). Dengan perkembangan teknologi diatas, maka metodologi perkembangan programming menjadi sangat berbeda dengan era sebelumnya, peluang untuk melakukan komunikasi data antar *server* menjadi sangat mudah dan cepat, yang pada akhirnya penelitian ini berusaha untuk menggali dan menghasilkan permodelan dalam menerapkan teknologi yang ada sehingga diperoleh model

programming yang relevan dengan kebutuhan serta dengan menambahkan beberapa teknik *locking* data dikarenakan data bisa berada pada sistem informasi yang berbeda dan besar kemungkinan akan atau diakses oleh orang yang berbeda disaat yang bersamaan, sehingga dibutuhkan informasi bahwa sebuah data atau *record* saat ini sedang di akses secara eksklusive oleh seseorang, sehingga orang lain tidak boleh mengakses untuk proses perubahan data dan seterusnya.

Dalam sebuah penelitian tentang membangun aplikasi *front end* berbasis *web* APPML oleh (Triyono *and* Haryani 2017) disimpulkan bahwa dengan menggunakan teknik APPML yang dikombinasikan dengan JSON akan bisa mendukung komunikasi antar sistem informasi, ditinjau dari sisi keamanan maupun *networking* lebih bagus karena data bisa langsung di peroleh secara langsung antara *service client* dan *service server* secara *realtime* selama ada koneksi internet sehingga penyebaran dan atau penyajian informasi tidak hanya dilakukan oleh sebuah sistem informasi saja, melainkan bisa di lakukan oleh sistem-sistem informasi yang lain. Dalam penelitian tentang Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada *Database PostgreSQL* oleh (Rosyid, 2016) disimpulkan bahwa JSON ternyata juga dapat digunakan untuk menyelesaikan kasus yang memerlukan tabel dengan kolom dinamis atau tabel tersebut berpotensi bertambah sesuai kebutuhan aplikasi yang menggunakan tabel tersebut. JSON dapat mereduksi dengan hanya menggunakan sebuah kolom yang berisi string format JSON. Untuk merubah *array* menjadi format JSON menggunakan fungsi *json_encode*, sedangkan untuk mengembalikan ke bentuk semula dapat menggunakan fungsi *json_decode*. Juga dalam penelitian tentang *JSON and its use in Semantic Web oleh (Pandey & Pandey, 2017)* disimpulkan bahwa fakta tentang informasi menjadi sangat penting untuk bisa berbagi diseluruh domain dan berbagai jenis pengguna, JSON sebagai media yang bisa dan mampu melakukannya. Dalam penelitian ini juga dengan jelas menyoroti betapa pentingnya ekspresi agen, entitas dan kegiatan yang dibuat dengan JSON. Makalah ini dengan jelas menyoroti relevansi dan kekuatan JSON dalam menciptakan asersi Provenansi sehubungan dengan klien yang berbeda. Dalam penelitian yang berbeda (Triyono, 2015) di peroleh hasil tentang pengembangan sebuah *prototype* sistem informasi yang di kombinasikan dengan jejaring sosial *twitter*, dimana jejaring *twitter* digunakan petani untuk melaporkan semua kegiatannya ke sistem informasi, dengan menggunakan fasilitas APIs (*Aplication Programming Language*) maka informasi yang masuk akan di kirimkan ke sistem informasi dengan menggunakan account dari twitter pengirim. Dengan metode ini secara teknologi dan biaya petani tidak mengalami kesulitan dalam melaporkan kegiatannya, sedangkan dari sisi investor akan bisa melihat perkembangan investasinya.

Dalam sebuah buku dengan judul *JSON at Work* (Marrs, 2017) *JavaScript Object Notation* (JSON) telah menjadi standar de facto untuk antarmuka *RESTful*, tetapi kelompok yang tidak banyak diketahui yaitu standar, alat, dan teknologi yang tersedia dapat mulai digunakan oleh perancang dan pengembang saat ini untuk membangun aplikasi yang dirancang dengan baik. JSON lebih dari sekadar pengganti sederhana untuk XML ketika anda melakukan panggilan AJAX. Ini menjadi tulang punggung dari pertukaran data yang serius melalui internet. Standar yang utama dan praktik terbaik dapat digunakan untuk memanfaatkan kemampuan dan keinginan menggunakan JSON untuk membangun aplikasi yang benar-benar elegan, berguna, dan efisien. Dalam sebuah buku dengan judul *Beginning JSON Learn and Preferred Data Format of The Web* (Smith, 2015) tentang Struktur Komposit JSON di katakan bahwa karena asal-usul JSON berasal dari standardisasi *ECMAScript*, implementasi dari dua struktur diwakili dalam bentuk objek dan larik. *Crockford* menguraikan dua representasi struktural JSON melalui serangkaian diagram sintaks. Karena saya yakin Anda akan setuju, diagram ini menyerupai rel kereta api dari pandangan mata burung dan dengan demikian juga disebut sebagai diagram rel kereta api. JSON terbuat dari dua struktur: (1) Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*. (2) Daftar nilai terurutkan (*an ordered list of values*).

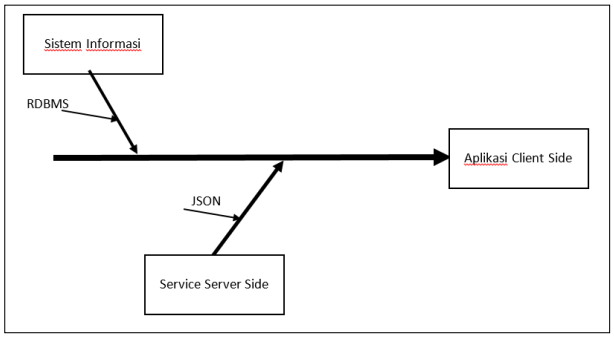
Metodologi Penelitian

Metode penelitian dilakukan dalam skala laboratorium, data atau model yang digunakan dalam penelitian ini adalah data tidak sesungguhnya. Fokus dari penelitian ini adalah sebuah sistem informasi yang menggunakan database *mysql* akan diakses melalui *service* oleh sebuah sistem informasi lain yang berbasis *html*. Gambar 1 menunjukkan *fishbone* diagram metode penelitian.

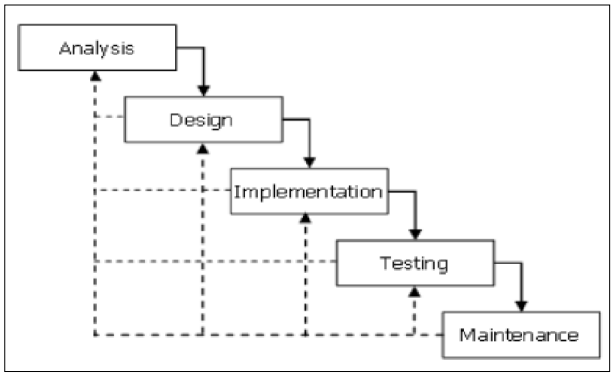
Sistem informasi dalam hal ini adalah sebuah sistem informasi semu yang memiliki database berbasis *MySQL*, ***Service*** adalah sisi *server side* yang akan melayani *request* dari luar untuk mengakses sistem informasi dan atau database, adapun akses yang dilakukan adalah membaca, menambah, mengedit, menghapus, mengunci dan melepas kunci. **Client side** adalah sebuah aplikasi atau sistem informasi berbasis HTML yang akan melakukan transaksional ke sistem informasi melalui *service*, dalam *client side* ini akan menggunakan *localStorage, sessionStorage, JavaScript* dan *HTML5*.

Pengembangan aplikasi menggunakan metode pengembangan *waterfall*. Metode *waterfall* ini dibangun oleh *Winston W. Royce* pada tahun 1970 untuk menggambarkan praktik produk perangkat lunak. Model *waterfall* terdiri

dari lima fase yaitu *analysis*, *design*, *implementation*, *testing* dan *maintenance*. Setiap fase pada model waterfall dapat dilihat pada Gambar 2.



Gambar 1. Fishbone Diagram



Gambar 2. Fase pada model waterfall

Hasil dan Pembahasan

Sistem informasi

Sebuah sistem informasi yang telah berjalan dengan metode lama, terdiri dari RDBMS MySQL dan aplikasi web berbasis PHP, dengan perubahan pada struktur tabel untuk control locking record. Gambar 3 menunjukkan diagram relasi dari tabel yang digunakan untuk ujicoba. Dari gambar 3 tersebut ada dua buah tabel yang berelasi, yaitu tabel barang dan tabel kunci, dimana tabel barang akan memuat data barang meliputi (kode, nama, harga dan keterangan) sedangkan tabel kunci akan memuat kode yang mana jika kode tersebut sedang dilakukan locking data oleh pengguna, jika proses locking data telah selesai maka data kode tersebut juga akan dihapus. Gambar 4 menunjukkan model data dari tabel barang, sedangkan gambar 5 menunjukkan model kunci, terlihat ada satu record data barang yang sedang dilock oleh user, yaitu kode=3. Artinya saat ini barang dengan kode tersebut sedang dilock eksklusif oleh user dan user lain tidak bisa melakukan event yang mengakibatkan perubahan pada data dengan kode=3 tersebut.

Gambar 3. Diagram relasi tabel

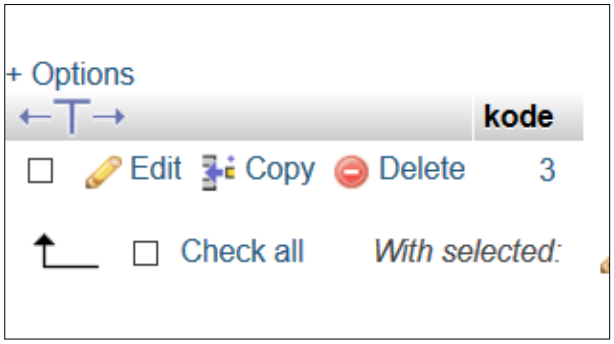
Service

Service diletakkan pada sisi server side yang akan melayani request dari luar untuk mengakses sistem informasi dan atau database, adapun akses yang dilakukan adalah membaca, menambah, mengedit, menghapus, mengunci dan melepas kunci.

Membaca Data, adalah service yang digunakan untuk membaca data dan menghasilkan atau mengembalikan data dengan format JSON. Script 1 adalah script untuk membaca data.

kode	nama	harga	keterangan
3	Mangga Manalagi	5000	Aseli palembang
4	Mangga Madu	4000	Manis harum aseli Sragen
5	Mangga Krikil	3000	Sikecil yang masih kecut segar
7	Apel Kuning Malang	6000	Apel kuning dari malang
8	Apel Hijau Malang	5000	Apel hijau kecut segar

Gambar 4 Contoh Model data dari Tabel Barang



Gambar 5 Model Locking data barang.

Script 1 Membaca Data

```
<?php
session_start();
$sql = "SELECT * FROM barang order by kode";
include("buka.php");
$result = $conn->query($sql);
if ($result->num_rows > 0) {
$arr = array();
while($r = $result->fetch_assoc()) {
$tmp=array('kode' => $r['kode'], 'nama' => $r['nama'], 'harga' => $r['harga'], 'keterangan' => $r['keterangan']);
array_push($arr, $tmp);   } }
$data2 =json_encode($arr);
echo $data2;
$conn->close();
?>
```

Hasil dari script baca tersebut adalah sebuah data JSON sebagai berikut:

```
[{"kode":"3","nama":"Mangga Manalagi","harga":"5000","keterangan":"Aseli palembang"},
{"kode":"4","nama":"Mangga Madu","harga":"4000","keterangan":"Manis harum aseli Sragen"},
{"kode":"5","nama":"Mangga Krikil","harga":"3000","keterangan":"Sikecil yang masih kecut segar"},
{"kode":"7","nama":"Apel Kuning Malang","harga":"6000","keterangan":"Apel kuning dari malang"},
{"kode":"8","nama":"Apel Hijau Malang","harga":"5000","keterangan":"Apel hijau kecut segar"}]
```

Menambah Data, adalah proses penambahan data yang dilakukan service dan akan mengembalikan data tunggal berupa data kode (last_id) hasil perintah insert, dengan script php pada script 2

Script 2 Menambah Data

```
<?php
session_start();
include("buka.php");
$sql = "insert into barang (nama,harga,keterangan) values('".$_GET['nama']."', '".$_GET['harga']."', '".$_GET['keterangan']."')";
$result = $conn->query($sql);
$last_id = $conn->insert_id;
echo $last_id;
$conn->close();
?>
```

Mengedit, adalah proses update data yang dilakukan oleh service dan secara otomatis juga melepas locking data dari tabel kunci, script 3 untuk mengedit data

Script 3 Mengedit Data

```
<?php
session_start();
```

```
$sql = "delete from kunci where kode=".$_GET['kode'];
include("buka.php");
$result = $conn->query($sql);
$sql = "update barang set nama='".$_GET['nama']."', harga='".$_GET['harga']."', keterangan='".$_GET['keterangan']."' where
kode=".$_GET['kode'];
$result = $conn->query($sql);
$conn->close();
?>
```

Menghapus, adalah proses menghapus data yang dilakukan oleh service, pada proses ini harus dilakukan check apakah data tersebut sedang dikunci atau tidak, jika tidak sedang dikunci maka proses penghapusan akan dilanjutkan sedangkan jika sedang dikunci maka akan ditampilkan berita tidak bisa melakukan penguncian data, hasil dari proses ini akan mengembalikan sebuah nilai tunggal beruka SUKSES atau GAGAL. Script 4 proses menghapus

Script 4 Proses Menghapus

```
<?php
session_start();
$sql = "select * from kunci where kode=".$_GET['kode'];
include("buka.php");
$result = $conn->query($sql);
if ($result->num_rows > 0)
{   $data2="gagal"; } else { $data2="sukses";
if($_GET['hapus']=='Y') { $sql = "delete from barang where kode=".$_GET['kode'];  $result = $conn->query($sql);
} } $conn->close();
echo $data2;
?>
```

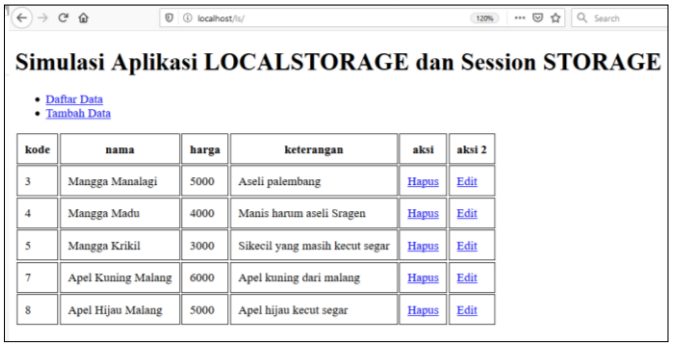
Mengunci, adalah proses mengunci record untuk memastikan bahwa record tersebut tidak sedang digunakan oleh user lain, hasil dari proses ini akan mengembalikan nilai tunggal. Script 5 dari proses mengunci.

Script 5 Proses Mengunci

```
<?php
session_start();
$sql = "insert into kunci values(".$_GET['kode'].")";
include("buka.php");
$result = $conn->query($sql);
if($result)   $data2="sukses";
else $data2="gagal";
$conn->close();
echo $data2;
?>
```

Client Side

Sebuah aplikasi atau sistem informasi berbasis HTML yang akan melakukan transaksional ke sistem informasi melalui *service*, dalam *client side* ini akan menggunakan *localStorage*, *sessionStorage*, *JavaScript* dan HTML5. Gambar 6 menampilkan sisi tampilan *client site*.



Gambar 6. Tampilan Client Site

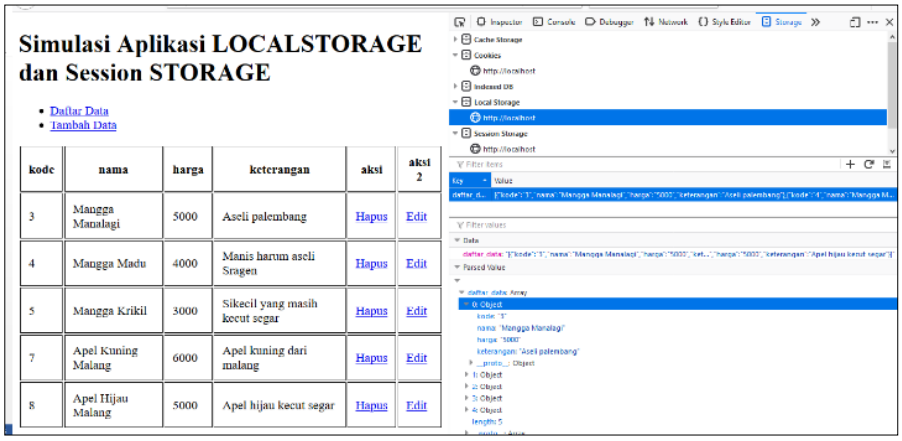
Membaca Data, membaca data dari posisi klien ada dua cara, yaitu membaca melalui *service* ini dilakukan jika data belum ada di *localStorage*, dan membaca data dari *localStorage* setelah data sudah masuk di *localStorage*, Script 6 proses membaca data dari *service* dan *localStorage*

Script 6 Membaca Data

```
function muatDaftarData(){
        if (localStorage.daftar_data){
           daftar_data = JSON.parse(localStorage.getItem('daftar_data'));
```

```
            var data_app = "";
            if (daftar_data.length > 0){
               data_app = '<table class="table1">'+'<thead>'+'<th>kode</th>'+'<th>nama</th>'+'<th>harga</th>'+'<th>keterangan</th>'+
                           '<th>aksi</th>'+'<th>aksi 2</th>'+'</thead><tbody>';
               for (i in daftar_data){
                 data_app += '<tr><td>'+ daftar_data[i].kode + ' </td>'+'<td>'+ daftar_data[i].nama + ' </td>'+'<td>'+ daftar_data[i].harga + ' </td>'+
'<td>'+ daftar_data[i].keterangan + ' </td>'+'<td><a class="btn btn-danger btn-small" href="javascript:void(0)"
onclick="hapusData(\''+daftar_data[i].kode+'\')">Hapus</a></td>'+'<td><a class="btn btn-warning btn-small" href="javascript:void(0)"
onclick="editData(\''+daftar_data[i].kode+'\')">Edit</a></td>';
                  data_app += '</tr>';
               } data_app += '</tbody></table>';  }
            else { data_app = "Tidak ada data..."; }
           $('#list-data').html(data_app); $('#list-data').hide(); $('#list-data').fadeIn(100);
        }    else {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {  localStorage.setItem('daftar_data', this.responseText);
            } }; xhttp.open("GET", "service/baca.php", true); xhttp.send();
            } }
```

Gambar 7 menampilkan kondisi *Storage* dari *browser* menggunakan *inspect element*



Gambar 7 Kondisi localStorage Klien

Dari gambar 7 terlihat bahwa *localStorage* sudah terisi data JSON dengan nama daftar_data dengan 5 data, proses pengambilan data melalui service menggunakan ajax *xhttp.open("GET", "service/baca.php", true);*, dari pemanggilan ini *service* akan mengembalikan data dalam bentuk JSON dan dimasukkan ke *daftar_data* sebagai berikut : *localStorage.setItem('daftar_data', this.responseText);* , Sehingga pada *browser* dengan menggunakan *inspect element* terlihat *localStorage* seperti pada gambar 7. Sedangkan untuk menampilkan data yang sudah ada pada *localStorage* maka digunakan perintah *daftar_data = JSON.parse(localStorage.getItem('daftar_data'));* untuk mengkonversi menjadi *array* yang akan dibaca oleh *javascript*.

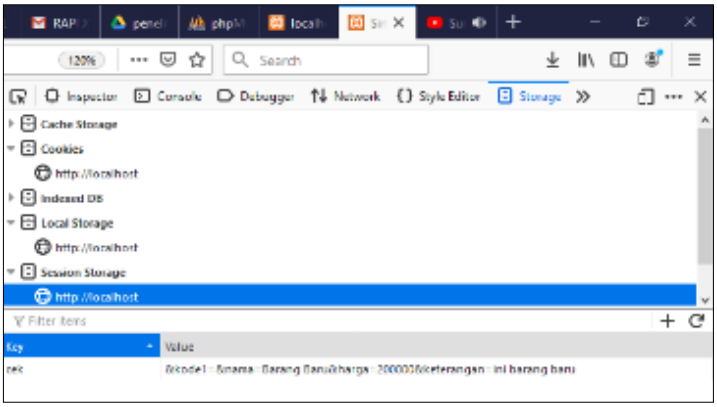
Menambah Data, Proses menambah data dengan mengisikan data di form HTML dan dikirimkan ke *javaScript* untuk proses penambahan data mula-mula data dikirimkan ke *service* menggunakan *ajax* baru jika server sudah memberikan response sukses maka data baru akan ditambahkan ke JSON di *localStorage*. Script 7 memperlihatkan proses menambah data.

Script 7 Proses Menambah Data

```
function simpanData(){
        kode = $('#kode').val(); nama = $('#nama').val(); harga = $('#harga').val();  keterangan = $('#keterangan').val();
        if (localStorage.daftar_data){ daftar_data = JSON.parse(localStorage.getItem('daftar_data'));
        } else {  daftar_data = []; kode = ''; }
            daftar_data_edit='&kode1='+kode+'&nama='+nama+'&harga='+harga+'&keterangan='+keterangan;
            sessionStorage.setItem("cek", daftar_data_edit);
            var xhttp = new XMLHttpRequest(); xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) { document.getElementById("demo").innerHTML =" ";
            kodebaru=this.responseText; daftar_data.push({'kode':kodebaru, 'nama':nama, 'harga':harga, 'keterangan':keterangan});
            localStorage.setItem('daftar_data', JSON.stringify(daftar_data)); document.getElementById('form-data').reset();
            gantiMenu('list-data'); }};
            xhttp.open("GET", "service/add.php?kode="+kode+"&edite="+daftar_data_edit, true);     xhttp.send();
```

```
            gantiMenu('list-data');
        return false;       }
```

Proses menambah data ini menggunakan *sessionStorage* seperti terlihat pada gambar 8, saat melakukan *submit* maka data akan dimasukkan ke *sessionStorage* dulu untuk kemudian dikirimkan menggunakan ajax *xhttp.open("GET", "service/add.php?kode="+kode+"&edite="+daftar_data_edit, true);* Dengan pengiriman parameter tersebut, dan jika respon sukses maka data akan ditambahkan ke *localStorage*.



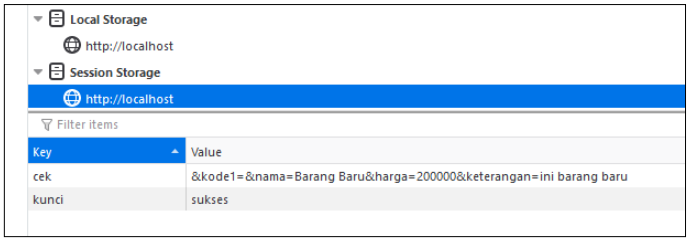
Gambar 8 *Session Storage* saat penambahan data

Mengedit Data, proses mengedit data dilakukan dengan melakukan *locking record* terhadap data yang akan diedit, *locking* harus sukses dulu di sisi *service* yang ditandai dengan respon dari *service* dan disimpan di *sessionStorage*, baru kemudian data bisa diupdate di *localStorage*. Gambar 9 menunjukkan kondisi *sessionStorage* saat *locking record* status *kunci* berisi data *sukses*. Dan jika proses edit sudah selesai maka proses *locking record* akan dilepaskan dan melakukan *update* di rdbms dengan mengirimkan ke *service*. Script 8 menunjukkan proses *editing.*

Script 8 Proses Editing Data

```
function editData(id){
        if (localStorage.daftar_data){
           daftar_data = JSON.parse(localStorage.getItem('daftar_data'));  idx_data = 0;
           for (i in daftar_data){
              if (daftar_data[i].kode == id){
                        $("#ekode").val(daftar_data[i].kode); $("#enama").val(daftar_data[i].nama);
                        $("#eharga").val(daftar_data[i].harga); $("#eketerangan").val(daftar_data[i].keterangan);
                        daftar_data.splice(idx_data,1);
                        }  idx_data ++; }
            //check lock
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                        sessionStorage.setItem("kunci", this.responseText);
                        if(sessionStorage.getItem("kunci")=='sukses')
                                    gantiMenu('edit-data');
                        else
                        document.getElementById("demo").innerHTML ="Data sedang diedit orang lain";
            } };
            xhttp.open("GET", "service/lock.php?kode="+id, true);  xhttp.send();      }}
function simpanEditData(){
        kode = $('#ekode').val();nama = $('#enama').val();harga = $('#eharga').val();
        keterangan = $('#eketerangan').val();
        daftar_data_edit='&kode1='+kode+'&nama='+nama+'&harga='+harga+'&keterangan='+keterangan;
        sessionStorage.setItem("cek", daftar_data_edit);
        daftar_data.push({'kode':kode, 'nama':nama, 'harga':harga, 'keterangan':keterangan});
        localStorage.setItem('daftar_data', JSON.stringify(daftar_data));
        document.getElementById('eform-data').reset();
            sessionStorage.removeItem("kunci");
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {  document.getElementById("demo").innerHTML =" "; gantiMenu('list-data'); }};
            xhttp.open("GET", "service/unlock.php?kode="+kode+"&edite="+daftar_data_edit, true);  xhttp.send();
```

```
            gantiMenu('list-data');
        return false;       }
```



Gambar 9 *sessionStorage* saat *locking record*

Menghapus Data, proses menghapus data dilakukan dengan terlebih dahulu melakukan *locking* data ke *service*, jika berhasil maka data di RDBMS dihapus terlebih dahulu baru kemudian dilakukan penghapusan pada *localStorage*. Script 9 menunjukkan proses penghapusan data.

Script 9 Proses Hapus Data

```
function hapusData(id){
 if (localStorage.daftar_data){
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
            if(this.responseText=='sukses') { daftar_data = JSON.parse(localStorage.getItem('daftar_data'));
            idx_data = 0;
            for (i in daftar_data){
            if (daftar_data[i].kode == id){ daftar_data.splice(idx_data, 1);  } idx_data ++; }
            localStorage.setItem('daftar_data', JSON.stringify(daftar_data));
            muatDaftarData();
            } else document.getElementById("demo").innerHTML ="Data sedang diedit orang lain"; }};
            xhttp.open("GET", "service/getlock.php?kode="+id+"&hapus=Y", true);  xhttp.send();}}
```

Kesimpulan

Setelah dilakukan penelitian dan diujikan maka dapat disimpulkan dari hasil penelitian yang telah dilakukan adalah sebagai berikut

a. bahwa proses pengembangan aplikasi berbasis client site bisa diimplementasikan, dengan pemanfaatan *localStorage* sebagai penampung data dalam format JSON dan *sessionStorage* sebagai penampung data tunggal.
b. Kombinasi penggunaan *ajax* untuk komunikasi ke *service* berbasis *server side* secara tepat akan menghasilkan aplikasi *client side* tidak kalah dengan aplikasi *server side*.
c. Dalam penggunaan data atau *record* bersama, maka dengan design rdbms yang tepat maka proses penguncian record bisa dilakukan.

Dalam penelitian ini yang belum dilakukan sehingga bisa dilakukan penelitian lanjutan dimasa yang akan datang meliputi dua hal sebagai berikut.

a. Penggunaan kunci akses ke *service* agar hanya client yang memiliki akses saja yang bisa menggunakan *service*.
b. Penerapan *update* data pada *localStorage* secara otomatis, karena data dipakai oleh banyak user, sehingga data yang ada di *localStorage* selalu terjamin keterbaruannya.

Daftar Pustaka

Crockford, Douglas. n.d. *JSON.* JSON. Accessed 11 16, 2019. http://www.json.org/json-id.html.

Deswitansyah, Irwan. 2018. *Cara Menggunakan LocalStorage dan SessionStorage.* DUMET school. Nov 2. Accessed 11 16, 2019. https://kursuswebprogramming.com/cara-menggunakan-local-storage-dan-session-storage.

Marrs, T. (2017). JSON at Work. Sebastopol: O'Reilly Media inc.

Pandey, M., & Pandey, R. (2017). JSON and its use in Semantic Web. International Journal of Computer Applications , 10-16.

Rohman, Ardani. 2017. *Dokumentation & Testing API dengan Postman part 2.* skyshi. 03 21. Accessed 11 16, 2019. https://medium.com/skyshidigital/documentation-testing-api-dengan-postman-part-2-d10ff0becf31.

Rosyid, M A. 2016. "Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada Database PostgreSQL." *Journal Of Informatics, Network, And Computer Science* 33-42.

Smith, B. (2015). Beginning JSON. New York: Apress.

Triyono, J. (2015). Sistem Informasi Agroteknologi Berbasis Web Dan Jejaring Sosial Twitter. Seminar Nasional IENACO , 205-212.

Triyono, Joko, And Prita Haryani. 2017. "Membangun Aplikasi Front End Web Dosen Berbasis Web Appml (Application Modeling Language)." Surakarta: Simposium Nasional RAPI XVI – 2017 FT UMS