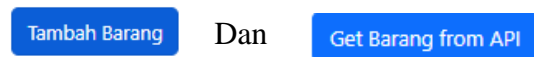
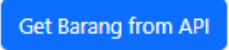




Gambar 1.

Gambar 1 merupakan halaman utama dari program get API Produk. Terdapat button



Jika tombol  di klik, program akan mengirimkan request ajax ke controller sebagai berikut.

```
$("#getBarang").click(function(e) {
    $.ajax({
        type: "POST",
        url: '<?= base_url('Index/getApi') ?>',
        success: function(response) {
            Swal.fire({
                icon: 'success',
                title: 'Berhasil',
                text: `${response.data.created} ditambahkan, ${response.data.alreadythere} sudah ada`,
            }).then((result) => {
                location.reload();
            });
        },
        error: function(jqXHR, textstatus, errorThrown) {
            Swal.fire({
                icon: 'error',
                title: 'Oops ... ',
                text: textstatus,
            });
        }
    });
});
```

Gambar 2.

Request dikirimkan ke controller Index dengan fungsi/method “getAPI” dengan kode PHP sebagai berikut.

```

public function getApi()
{
    date_default_timezone_set('Asia/Jakarta');
    $now = date('dmy');
    $hour = date('H');
    $username = 'tesprogrammer' . $now . 'C' . $hour;
    $password = md5('bisacoding-' . date('d-m-y'));
    $data_post = array(
        'username' => $username,
        'password' => $password
    );

    $payload = http_build_query($data_post);

    $ch = curl_init('https://recruitment.fastprint.co.id/tes/api_tes_programmer');
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HEADER_OUT, true);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
    $response = curl_exec($ch);
    curl_close($ch);
    $data = json_decode($response);
    $create = $this->barang->create($data->data);
    if ($create['status'] === true) {
        $this->responseJSON(201, [
            'status' => 'Success',
            'message' => 'Berhasil get API dan menambahkan data',
            'data' => [
                'created' => $create['created'],
                'alreadythere' => $create['alreadythere'],
            ]
        ]);
    } else {
        $this->responseJSON(400, [
            'status' => 'Failed',
            'message' => 'Gagal menambahkan data',
            'data' => $create['error']
        ]);
    }
}

```

Gambar 3.

Fungsi *getAPI* dimulai dari deklarasi timezone Asia/Jakarta atau WIB. Variabel *now* dan *hour* dideklarasikan dengan nilai datetime dari php. Kedua variabel ini digunakan sebagai username dan password untuk request ke API. Untuk nilai password perlu diubah ke dalam string MD5.

Variabel *data_post* merupakan array asosiatif yang kemudian diubah menjadi form request payload dengan fungsi *http_builder_query* dari PHP. Lalu kemudian dilanjutkan dengan inisiasi CURL menuju link API, beserta dengan option method, header, dan postfield dari variabel *data_post*.

String response JSON kemudian di decode agar menjadi format JSON, yang kemudian value array dari key *data* dilakukan proses creating data di dalam fungsi model *barang* sebagai berikut.

```
public function create($data)
{
    $created = 0;
    $alreadythere = 0;
    $databarang = null;
    foreach ($data as $d) {
        $isThere = $this->db->get_where('barang', ['id_produk' => $d->id_produk])->row();
        if ($isThere == null) {
            $databarang[] = [
                'id_produk' => $d->id_produk,
                'nama_produk' => $d->nama_produk,
                'kategori' => $d->kategori,
                'harga' => $d->harga,
                'status' => $d->status,
            ];
            $created++;
        } else $alreadythere++;
    }
    $error = null;
    $this->db->trans_start();

    if ($databarang != null) {
        $this->db->insert_batch('barang', $databarang);
    }
    $error[] = [
        'error' => $this->db->error(),
        'query' => $this->db->last_query()
    ];

    $this->db->trans_complete();
    if ($this->db->trans_status() == TRUE) {
        $this->db->trans_commit();
        return [
            'status' => true,
            'created' => $created,
            'alreadythere' => $alreadythere
        ];
    }
    $this->db->trans_rollback();

    return [
        'status' => false,
    ];
}
```

Gambar 4.

Dalam fungsi create terdapat pengecekan value ke dalam database. Jika nama barang sudah ada di dalam database, maka data dengan nama barang tersebut tidak akan dimasukkan ke dalam database. Semua barang yang bersifat unik (masih belum ada di database) akan disimpan dalam variabel array *databarang* yang kemudian akan di masukkan ke dalam tabel *barang* dengan fungsi *insert_batch* dari codeigniter. Proses memasukkan data ke dalam tabel barang

menggunakan transaksi untuk meminimalisir error. Dengan menggunakan transaksi, jika terdapat suatu *query* insert data barang yang error, maka proses pemasukkan data tidak akan di commit atau di lakukan.

```
public function index()
{
    $data['barang'] = $this->barang->get('*', ['status' => 'bisa dijual']);
    $data['status'] = [
        'bisa dijual',
        'tidak bisa dijual'
    ];
    $this->renderTo('index', $data);
}
```

Gambar 5.

```
/**
 * get
 *
 * @param array $select digunakan untuk field apa yang diambil
 * @param array $where digunakan untuk query where
 * @param int $limit limit digunakan untuk membatasi jumlah kolom
 * @return void
 */
public function get($select = null, $where = null, $limit = null)
{
    if ($select != null)
        $this->db->select($select);
    else
        $this->db->select('*');

    if ($where != null)
        $this->db->where($where);

    if ($limit != null)
        $this->db->limit($limit);

    $this->db->order_by('kategori', 'asc');
    $result = $this->db->get('barang');
    if ($limit == 1) {
        return $result->row();
    }
    return $result->result();
}
```

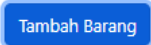
Gambar 6.

Fungsi index pada controller di gambar 5 digunakan untuk menampilkan view pada yang ada pada gambar 1. Data barang diambil dari fungsi *get* dalam model *barang*. Fungsi *get* ini

digunakan untuk mengambil data yang ada pada tabel barang, dimana fungsi get memiliki 3 parameter, yakni *select* (untuk kolom yang dipilih), *where* (untuk kondisi where), dan *limit* (untuk melimit atau membatasi jumlah baris). Data barang ditampilkan ke dalam view dengan menggunakan *foreach*.

```
<tbody>
  <?php
    $i = 0;
    foreach ($barang as $b) :
      $i++; ?>
      <tr>
        <td><?= $i ?></td>
        <td><?= $b->id_produk ?></td>
        <td><?= $b->nama_produk ?></td>
        <td><?= $b->harga ?></td>
        <td><?= $b->kategori ?></td>
        <td><?= $b->status ?></td>
        <td>
          <button type="button" onclick="edit(<?= $b->id_produk ?>)" class="btn btn-primary btn-sm">Edit</button>
          <button type="button" onclick="deletes(<?= $b->id_produk ?>)" class="btn btn-danger btn-sm">Hapus</button>
        </td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
```

Gambar 7.

Tombol  jika di klik akan menampilkan modal form isian tambah barang dimana form ini akan di submit dengan menggunakan ajax jquery.



The image shows a modal window titled "Tambah Barang" with a close button (X) in the top right corner. Inside the modal, there are four input fields: "Nama Barang" (text input), "Harga (Rp)" (text input), "Kategori" (text input), and "Status" (dropdown menu with "bisa dijual" selected). At the bottom right of the modal, there is a blue button labeled "Save Data".

Gambar 8.

```

$("#tambah").submit(function(e) {
  e.preventDefault();
  let formdata = new FormData(this);
  $.ajax({
    type: "POST",
    data: formdata,
    processData: false,
    contentType: false,
    url: '<?= base_url() ?>index/add',
    success: function(response) {
      Swal.fire({
        icon: 'success',
        title: 'Berhasil',
        text: response.message,
      }).then((result) => {
        if (result.isConfirmed) location.reload();
      });
    },
    error: function(jqXHR, textStatus, errorThrown) {
      let response = jqXHR.responseJSON;
      response.data.forEach(({
        field,
        message
      }) => {
        $('.invalid-feedback[for="${field}"]').html(message);
        $('#${field}').addClass('is-invalid');
      })
    }
  });
});

```

Gambar 9.

Untuk form validation dihandle oleh controller Index dengan fungsi *validation_form* sebagai berikut.

```

private function validation_form($prefix = '', $onupdate = false)
{
    $errors = false;
    $this->form_validation->set_rules([
        'nama_produk' . $prefix,
        'Nama',
        'required|min_length[5]|max_length[255]' . ($onupdate ? '' : '|is_unique[barang.nama_produk]')
    ]);
    $this->form_validation->set_rules('harga' . $prefix, 'Harga', 'required|integer');
    $this->form_validation->set_rules('kategori' . $prefix, 'Kategori', 'required|min_length[5]|max_length[50]');
    $this->form_validation->set_rules('status' . $prefix, 'Status', 'required');

    if ($this->form_validation->run() == false) {
        $errors = [];
        foreach ($this->input->post() as $field => $value) {
            if (form_error($field)) {
                $errors[] = [
                    'field' => $field,
                    'message' => trim(form_error($field, ' ', ' ')),
                ];
            }
        }
    }

    return $errors;
}

```

Gambar 10.


Fungsi ini terdiri dari dua parameter yaitu prefix dan onupdate, dimana kedua parameter ini digunakan untuk pengecekan apakah fungsi ini digunakan untuk validasi edit atau tambah. Fungsi ini merupakan fungsi private yang diakses oleh fungsi *add* ketika melakukan tambah data.

```

public function add()
{
    $isError = $this->validation_form();
    if ($isError) {
        return $this->responseJSON(400, [
            'status' => 'Gagal',
            'message' => 'Gagal menambahkan data!',
            'data' => $isError
        ]);
    } else {
        $data = $this->input->post();
        if ($data == null) {
            return $this->responseJSON(400, [
                'status' => 'Failed',
                'message' => 'Akses hanya dengan method POST!',
            ]);
        }
        $this->db->insert('barang', $data);
        return $this->responseJSON(201, [
            'status' => 'Berhasil',
            'message' => 'Berhasil menambahkan data',
        ]);
    }
}

```

Gambar 11.

Ketika tombol  di klik, maka program akan melakukan request menggunakan ajax ke fungsi untuk melihat detail suatu produk. Semua detail dari produk yang didapatkan akan ditampilkan ke dalam form di modal untuk edit data.

```
function edit(id) {  
  $.ajax({  
    type: "GET",  
    data: {  
      id: id  
    },  
    url: '<?= base_url() ?>index/produk_get',  
    success: function(response) {  
      $("#modal_edit").modal('show');  
  
      const produk = response.data;  
      $("#id_produk").val(produk.id_produk);  
      $("#nama_produk").val(produk.nama_produk);  
      $("#kategori").val(produk.kategori);  
      $("#harga").val(produk.harga);  
      $("#status").val(produk.status).trigger('change');  
    },  
    error: function(jqXHR, textStatus, errorThrown) {  
      Swal.fire({  
        icon: 'error',  
        title: 'Gagal',  
        text: jqXHR.responseJSON.message,  
      });  
    }  
  });  
}
```

Gambar 12.



Edit Barang

Nama Barang
BOTOL 1000ML BLUE KHUSUS UNTUK EPSON R1800/R800 - 4180 IM (T054920)

Harga (Rp)
10000

Kategori
CI MTH TINTA LAIN (IM)

Status
bisa dijual

Save Data

Gambar 13.


```

public function edit()
{
    $isError = $this->validation_form('_', true);
    if ($isError) {
        return $this->responseJSON(400, [
            'status' => 'Gagal',
            'message' => 'Gagal mengubah data!',
            'data' => $isError
        ]);
    } else {
        $data = $this->input->post();
        if ($data == null) {
            return $this->responseJSON(400, [
                'status' => 'Failed',
                'message' => 'Akses hanya dengan method POST!',
            ]);
        }

        $this->db->update(
            'barang',
            [
                'nama_produk' => $data['nama_produk__'],
                'harga' => $data['harga__'],
                'kategori' => $data['kategori__'],
                'status' => $data['status__'],
            ],
            ['id_produk' => $data['id_produk']]
        );
        return $this->responseJSON(201, [
            'status' => 'Berhasil',
            'message' => 'Berhasil mengubah data',
        ]);
    }
}

```

Gambar 14.

```

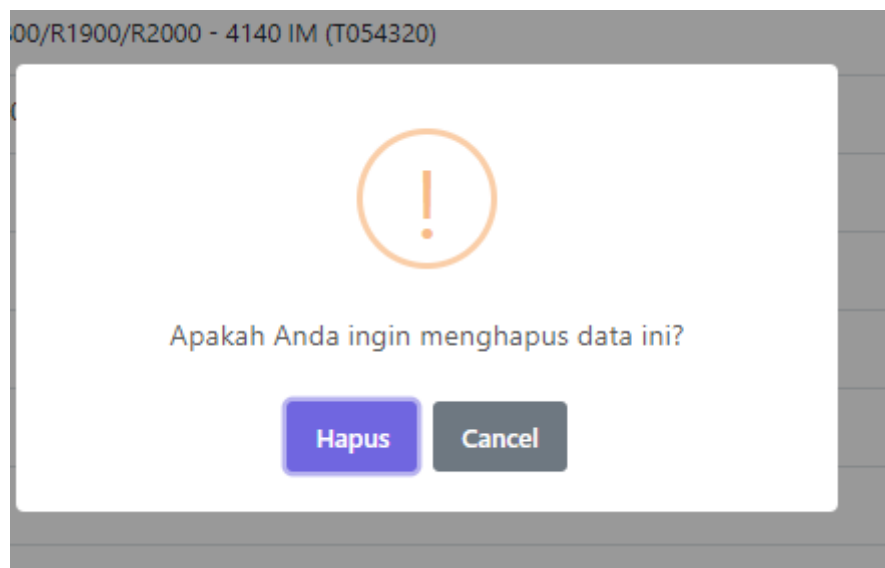
$("#edit").submit(function(e) {
    e.preventDefault();
    let formdata = new FormData(this);
    $.ajax({
        type: "POST",
        data: formdata,
        processData: false,
        contentType: false,
        url: '<?= base_url() ?>index/edit',
        success: function(response) {
            Swal.fire({
                icon: 'success',
                title: 'Berhasil',
                text: response.message,
            }).then((result) => {
                if (result.isConfirmed) location.reload();
            });
        },
        error: function(jqXHR, textStatus, errorThrown) {
            let response = jqXHR.responseJSON;
            response.data.forEach(({
                field,
                message
            }) => {
                $('<div>.invalid-feedback[for="<div>${field}</div>"</div>').html(message);
                $('<div>#<div>${field}</div>').addClass('is-invalid');
            })
        }
    });
});

```

Gambar 15.

Fungsi *edit* digunakan untuk handling event form edit yang dikirimkan melalui request ajax ketika form edit disubmit. Fungsi ini menggunakan *validation_form* yang sudah di dekarasikan sebelumnya.

Ketika tombol **Hapus** diklik pada suatu kolom tabel data. Maka program akan menampilkan alert konfirmasi dari sweetalert sebagai berikut.



Gambar 16.

Jika pengguna klik tombol **Hapus** maka program akan mengirimkan request ke controller untuk menghapus data tersebut.

```

function deletes(id) {
    Swal.fire({
        icon: 'warning',
        text: 'Apakah Anda ingin menghapus data ini?',
        showCancelButton: true,
        confirmButtonText: 'Hapus',
        denyButtonText: `Batal`,
    }).then((result) => {
        if (result.isConfirmed) {
            $.ajax({
                type: "POST",
                data: {
                    id: id
                },
                processData: true,
                dataType: 'JSON',
                url: '<?= base_url() ?>index/delete',
                success: function(response) {
                    Swal.fire({
                        icon: 'success',
                        title: 'Berhasil',
                        text: response.message,
                    }).then((result) => {
                        if (result.isConfirmed) location.reload();
                    });
                },
                error: function(jqXHR, textStatus, errorThrown) {
                    Swal.fire({
                        icon: 'error',
                        title: textStatus,
                        text: jqXHR.responseText.message.body,
                    });
                }
            });
        }
    });
}

```

Gambar 17.

```

public function delete()
{
    $id = $this->input->post('id');
    if ($id == null) {
        return $this->responseJSON(400, [
            'status' => 'Failed',
            'message' => 'Akses hanya dengan method POST!',
        ]);
    }
    $this->db->delete('barang', ['id_produk' => $id]);
    return $this->responseJSON(200, [
        'status' => 'Success',
        'message' => 'Berhasil menghapus data',
    ]);
}

```

Gambar 18.

Proses hapus data dilakukan pada fungsi yang terlampir pada gambar 18, dimana terdapat pengecekan/validasi untuk id_produk. Jika id_produk dikirimkan melalui body request, maka baris dengan id_produk yang dikirimkan akan dihapus