# Pengembangan Layanan Pengumpulan Data Tempat Makan di Indonesia Secara Kolaboratif

Disusun Oleh:

- Eric Simahan
- Nirmala Agatha Santoso
- Mikhael Diastama Santoso
- Heryatmo Benediktus Sembiring
- Vincentius Andri Kurnianto
- Sandi Prando Saragih

Program Studi Teknik Informatika

Universitas Atma Jaya Yogyakarta

2017

# Daftar Isi

# Bab 1 Pendahuluan

Indonesia sebagai negara yang cukup luas memiliki banyak daerah-daerah unik yang belum diketahui oleh masyarakat dalam negeri maupun luar negeri, termasuk dalam hal tempat makan yang ada di seluruh Indonesia.

Bagaimana agar seluruh masyarakat mengetahui dan mengenal tempat tersebut?

Disinilah data yang bersifat kolaboratif dibutuhkan. Dengan adanya pengumpulan data secara kolaboratif maka seluruh masyarakat bisa ikut menambah data-data tersebut dari berbagai tempat yang berbeda di seluruh Indonesia.

Hal ini membantu mempercepat proses pengumpulan data dan juga sekaligus meningkatkan popularitas negara Indonesia di bidang tempat makan.

Adapun tantangan tersendiri dalam pengumpulan data yang bersifat kolaboratif ini, yaitu daya tarik para pengguna-pengguna yang akan membantu pengumpulan data.

Disini aplikasi layanan pengumpulan data tersebut akan dibentuk dengan nama Restopedia. Aplikasi ini berbasis Android sebagai front-end pengumpulan data dan Web sebagai tempat pengambilan data yang telah menyediakan API secara publik bagi pengembang-pengembang lain yang memerlukan data sejenis.

# Bab 2 Pembuatan API dengan Framework CodeIgniter

## 2.1. Instalasi CodeIgniter dan REST Library

Dalam pembuatan aplikasi web Restopedia kita akan menggunakan Framework yang bernama CodeIgniter.

CodeIgniter yang digunakan adalah CodeIgniter versi ke 3.0.6 yang dapat diunduh di website berikut: https://www.codeigniter.com/download

Setelah berhasil mengunduh, anda akan mendapatkan folder yang berisikan file-file utama dari framework tersebut.

### Instalasi CodeIgniter pada localhost:

Pindahkan folder tersebut ke htdocs dan akses dengan alamat url sebagai berikut : localhost/codeIgniter-3.0.6

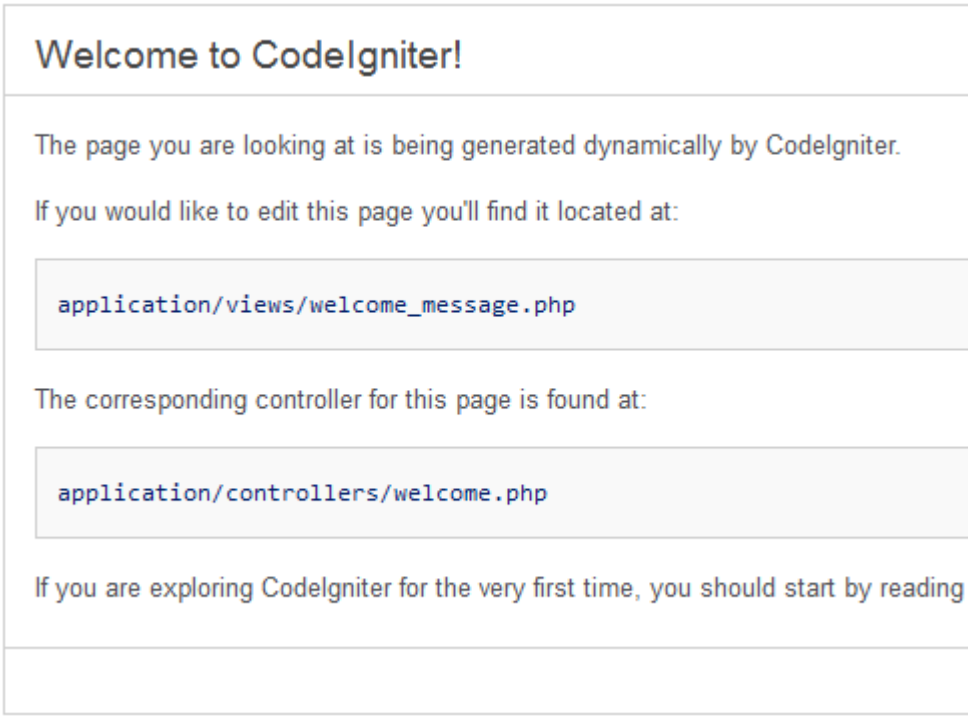
### Instalasi CodeIgniter pada server online:

Pindahkan folder tersebut ke public_html dan akses dengan alamat url sebagai berikut : [nama_domain]/codeIngiter-3.0.6

Catatan : Anda dapat merubah nama folder dan mengakses alamat ke nama folder baru tersebut.

Jika tampilan web browser anda menampilkan teks yang bertuliskan "Welcome to CodeIgniter!" maka selamat anda telah berhasil menginstalasi framework tersebut dan lanjut ke tahap selanjutnya.

Contoh tampilan awal CodeIgniter:




## Pengenalan REST

REST (Representational State Transfer) adalah jenis web service yang menerapkan konsep perpindahan antar state. Perintah HTTP yang biasa digunakan untuk aktivitas-aktivitas ini adalah GET, POST, PUT atau DELETE.

Masing-masing memiliki arti sendiri sehingga memudahkan pembacaan dari sisi client.

GET digunakan untuk pengambilan data.

POST digunakan untuk memasukkan data baru.

PUT digunakan untuk mengedit data yang sudah ada.

DELETE digunakan untuk menghapus data.

Konsep ini akan mempermudah penggunaan API (Application Programming Interface) di kedua belah pihak dan dapat memberi keuntungan lain dengan cara mempublikasi API terhadap data yang telah kita miliki.

## RESTful API Server Library (CodeIgniter)

Library ini dapat diunduh pada link berikut :

https://github.com/chriskacerguis/codeigniter-restserver

Didalam link tersebut juga tercantum README.md yang mencakup hal-hal penting seperti Spesifikasi kebutuhan, Cara Instalasi dan Pemakaian Library yang diberikan.

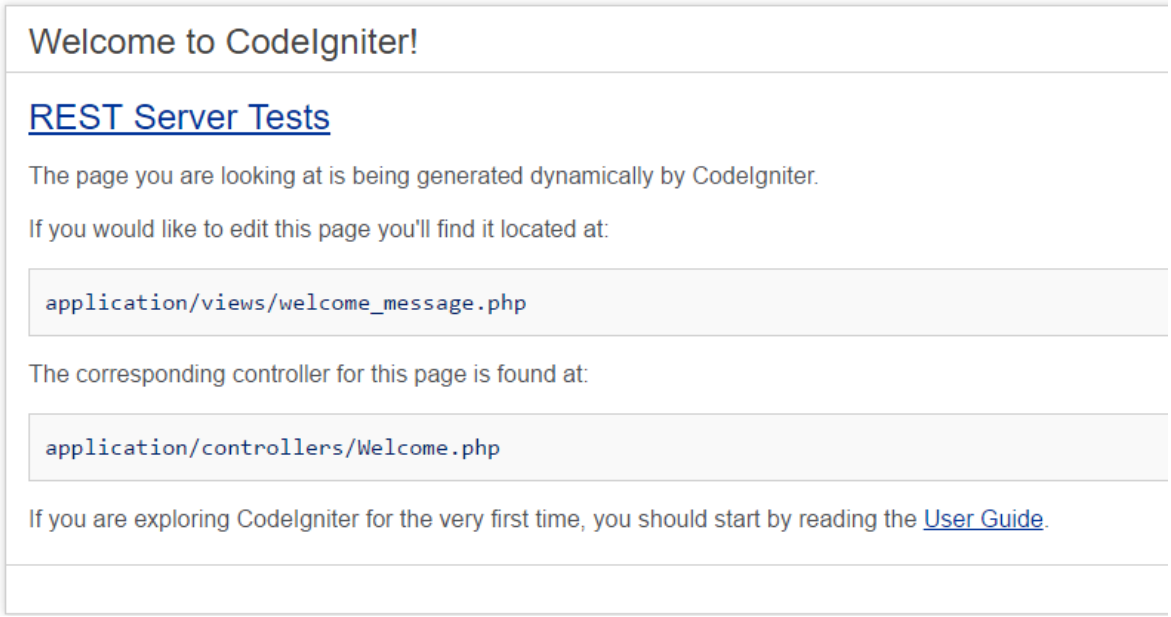
Secara singkat hal yang perlu anda lakukan saat menginstalasi library ini adalah dengan memindahkan file yang terdapat di application/libraries/Format.php dan application/libraries/REST_Controller.php ke direktori aplikasi anda.

Kemudian pindahkan juga file dengan nama **rest.php** yang ada pada **application/config** ke direktori aplikasi anda.

Setelah anda yakin instalasi berjalan sesuai dengan dokumentasi yang diberikan, maka akses index.php dari aplikasi anda.

Jika tampilan browser anda menampilkan teks bertuliskan “REST Server Tests” maka anda telah berhasil memasukkan library ini.

Contoh tampilan awal CodeIgniter dengan RESTful Server Library:



## 2.2. Pembuatan API

Untuk membuat sebuah API, sebelumnya anda harus mengetahui apa saja data-data yang anda atau konsumen butuhkan. Terutama data yang akan diakses oleh aplikasi mobile anda yang tidak dapat secara langsung mengakses ke database server dan data yang akan anda publikasikan.

Aplikasi android yang akan kita bangun yaitu Restopedia secara garis besar akan memiliki fungsi-fungsi sebagai berikut:

### Daftar Pengguna

Befungsi untuk mendaftarkan pengguna baru yang akan masuk kedalam sistem. Data yang dikirimkan berupa Email dan Password dengan tipe pengiriman POST.

### Login Pengguna

Berfungsi untuk mengotentikasi pengguna yang masuk kedalam sistem. Data yang dikirimkan berupa Email dan Password dengan tipe pengiriman POST.

### Mendaftarkan Konten Baru

Berfungsi untuk menambah konten tempat makan yang dimiliki oleh pengguna. Data-data yang dikirimkan berupa data konten yaitu email pengirim, nama konten, detail konten, judul konten, gambar, tags dan lokasi dengan tipe pengiriman POST.

### Mengambil Konten (Privat dan Publik)

Berfungsi untuk mengambil konten-konten yang telah didaftarkan oleh pengguna secara kolaboratif. Konten-konten yang diambil akan ditampilkan dengan format JSON dengan tipe pengiriman GET.

Setelah berhasil mengidentifikasi seluruh API dari fungsi-fungsi yang akan dibentuk. Maka kita dapat menyimpulkan apa saja API yang dibutuhkan dalam pembangunan layanan ini.

Persiapkan dan masukkan seluruh database anda pada database local atau server. Masing-masing table dari database anda nantinya akan memiliki sebuah file dengan nama [Nama_table]_model.php untuk memudahkan penamaan.

Dalam projek ini memiliki 2 model yaitu: **Konten_model** dan **User_model** yang terletak di folder **models**.

Perlu diketahui bahwa dalam pembuatan API, seluruh akses database akan dilakukan oleh model anda yang kemudian fungsinya baru akan dipanggil oleh rest controller.

### 2.2.1. API Daftar Pengguna

Model yang akan dimodifikasi: User_model.php
Tipe pengiriman: POST
Parameter: Username, Password
Hasil: Status, Message

Pada saat pendaftaran kita akan menambahkan isi dari database user_app dengan syntax insert yang merupakan bagian dari query builder CodeIgniter.

Maka fungsi pada model akan terlihat seperti berikut:

```
function SignUp($Username,$Password){
    $randstring = random_string('alnum', 20);
    $this->db->select_max('ID_User');
    $query = $this->db->get('user');
    foreach ($query->result() as $row)
    {
        $maxid = $row->ID_User+1;
    }
    $data = array(
            'username' => $Username,
            'password' => $Password,
            'api_key' => $randstring.$maxid,
        );

    $this->db->insert('user',$data);
}
```

Kemudian panggil fungsi dari model di rest controller anda.
Pertama-tama tentukan dulu nama pemanggilan yang diinginkan diikuti dengan underscore (_) jenis pengiriman yang dilakukan yaitu post.

Siapkan pesan yang akan dikeluarkan di hasil dan juga mengambil parameter-parameter yang dikirmkan dengan cara $this->post('nama parameter');

Maka fungsi pada controller akan terlihat seperti berikut:

```php
public function signUp_post(){
$username=$this->post('username');
$password=$this->post('password');
$apikey=$this->post('api_key');

if($this->User_model->Authenticate($apikey)){
  $message = [
        'status'=>1,
        'message'=>'Added User',
    ];

    $this->User_model->SignUp($username,$password);
}else{
  $message = [
    'status' => 101,
    'message' => 'Gagal',
  ];
}

    $this->set_response($message, REST_Controller::HTTP_CREATED);
}
```

### 2.2.2. API Login Pengguna

Model yang akan dimodifikasi: User_model.php

Tipe pengiriman: POST

Parameter: Username, Password

Hasil: Status, Message, userD

ata

Pada fungsi login kita ingin melihat apakah email dan password pengguna cocok dengan yang kita miliki di database.

Syntax query builder yang akan kita gunakan disini adalah select yang memiliki kondisi dengan menggunakan where apabila email dan password nya sama. Selanjutnya hasil query akan dihitung dengan menggunakan num_rows.

Maka fungsi pada model akan terlihat seperti berikut:

```
function Login($Username,$Password){
    $this->db->select('*');
    $this->db->from('user');
    $this->db->where('username',$Username);
    $this->db->where('password',$Password);

    $query = $this->db->get();

    if($query ->num_rows()==1)
    {
        return $query->result();
    }
    else
    {
        return false;
    }
}
```

dan fungsi pada controller akan terlihat seperti berikut:

```php
public function login_post(){
$username=$this->post('username');
$password=$this->post('password');
$apikey=$this->post('api_key');

$userData = $this->User_model->Login($username,$password);

if($this->User_model->Authenticate($apikey)){
    if($userData != NULL){
        $message = [
            'status' => 200,
            'message' => 'Login Succeed',
            'userData' => $userData,
        ];
    }else{
        $message = [
            'status' => 201,
            'message' => 'Login Not Succeed',
            'userData' => 'User not Found',
        ];
    }
}else{
  $message = [
    'status' => 101,
    'message' => 'Gagal',
    'userData' => 'Akses tidak valid',
  ];
}
```

### 2.2.3. API Mendaftarkan Konten Baru

Model yang akan dimodifikasi: Konten_model.php

Tipe pengiriman: POST

Parameter: Username, Nama_Resto, Detail_Resto, Alamat, Kota, Gambar, Nama_Gambar

Hasil: Status, Message

Mendaftarkan konten baru mirip seperti fungsi daftar baru pengguna. Perbedaanya hanya terletak di database yang akan ditambahkan yang itu konten_resto.

Oleh karena itu fungsi pada model akan terlihat seperti berikut:

```php
function AddKonten($Username, $Nama_Resto, $Detail_Resto, $Alamat, $Kota, $Gambar, $Nama_Gambar)
{
    $konten = array(
        'Username' => $Username,
        'Nama_Resto' => $Nama_Resto,
        'Detail_Resto' => $Detail_Resto,
        'Alamat' => $Alamat,
        'Kota' => $Kota,
        'Gambar' => 'http://restopedia.890m.com/restopedia-doc/'.$Nama_Gambar
    );

        $imsrc = base64_decode($Gambar);
        $fp = fopen($_SERVER['DOCUMENT_ROOT'].'/restopedia-doc/'.$Nama_Gambar.'', 'w');
        fwrite($fp, $imsrc);

    $this->db->insert('konten_resto',$konten);
}
```

Untuk pengiriman gambar akan menggunakan base64_encode yang digunakan di android dan base64_decode pada saat diterima di server. Setelah di decode file tersebut disimpan di lokasi yang telah ditentukan.

Kode untuk decode pada fungsi AddKonten dapat dilihat pada 3 baris yang ada sebelum data dimasukkan ke database.

Walaupun menggunakan base64_decode, tidak ada perubahan yang dilakukan pada controller. Hanya saja pastikan bahwa seluruh parameter telah didapatkan dan dikirim ke fungsi yang ada pada model seadanya tanpa ada perubahan.

Berikut potongan kode yang ada pada controller:

```php
public function addKonten_post(){
$apikey=$this->post('api_key');
$username=$this->post('username');
$nama=$this->post('nama');
$detail=$this->post('detail');
$alamat=$this->post('alamat');
$kota=$this->post('kota');
$gambar=$this->post('gambar');
$namagambar=$this->post('namagambar');

if($this->User_model->Authenticate($apikey)){
    $message = [
        'status'=>1,
        'message'=>'Added Konten',
    ];

    $this->Konten_model->AddKonten($username,$nama,$detail,$alamat,$kota,$gambar,$namagambar);
}else{
  $message = [
    'status' => 101,
    'message' => 'Gagal',
  ];
}

    $this->set_response($message, REST_Controller::HTTP_CREATED);
}
```

### 2.2.4. API Mengambil Konten (Privat dan Publik)

Model yang akan dimodifikasi: Konten_model.php

Tipe pengiriman: GET

Parameter: Username

Hasil: Status, Message, Kontendata

Akan ada 2 fungsi yang digunakan untuk mengambil konten yaitu untuk API privat dan API publik. Fungsinya akan terjadi sedikit perbedaan pada saat pengambilan data.

Fungsi pengambilan data konten API Privat pada model:

```php
function GetMyKonten($LoggedUsername)
{
    $this->db->select();
    $this->db->from('konten_resto')->where('Username', $LoggedUsername)->order_by('Created_at', 'DESC');

    $query = $this->db->get();

    $result =array();

    foreach($query->result() as $row) {

        $result[] = array(
        'ID_Resto' => $row->ID_Resto,
        'Username' => $row->Username,
        'Nama_Resto' => $row->Nama_Resto,
        'Detail_Resto' => $row->Detail_Resto,
        'Alamat' => $row->Alamat,
        'Kota' => $row->Kota,
        'Gambar' => $row->Gambar,
        'Created_at' => $row->Created_at
        );
    }
    return $result;
}
```

Fungsi pengambilan data konten API Publik pada model:

```php
function GetKonten()
{
    $this->db->select();
    $this->db->from('konten_resto')->order_by('Created_at', 'DESC');

    $query = $this->db->get();

    $result =array();

    foreach($query->result() as $row) {

        $result[] = array(
        'ID_Resto' => $row->ID_Resto,
        'Username' => $row->Username,
        'Nama_Resto' => $row->Nama_Resto,
        'Detail_Resto' => $row->Detail_Resto,
        'Alamat' => $row->Alamat,
        'Kota' => $row->Kota,
        'Gambar' => $row->Gambar,
        'Created_at' => $row->Created_at
        );
    }
    return $result;
}
```

Dapat dilihat bahwa data yang diberikan di API privat jauh lebih banyak dibandingkan data yang diberikan melalui API publik.

Hal ini dikarenakan ada beberapa data yang sifatnya sensitif dan tidak diberikan kepada publik contohnya seperti vote masing-masing

pengguna terhadap suatu konten yang takutnya bisa di manipulasi oleh pihak-pihak yang tidak bertanggung jawab.

Sedangkan untuk fungsi di controller tidak memiliki perbedaan yang terlalu mencolok, hanya saja pada API publik akan memiliki fungsi otentikasi API KEY yang digunakan oleh pengguna sebelum memberikan data hasil JSON.

Fungsi pengambilan data konten privat pada controller:

```
public function getMyKonten_get(){
    $username=$this->get('username');
$apikey=$this->get('api_key');

    $kontenData = $this->Konten_model->GetMyKonten($username);

if($this->User_model->Authenticate($apikey)){
    if($kontenData != NULL){
        $message = [
            'status' => 200,
            'message' => 'All Konten Data Received',
      'kontenData' => $kontenData,
        ];
    }else{
        $message = [
            'status' => 101,
            'message' => 'Konten Data Retrieve Failed',
            'kontenData' => 'Konten not Found',
        ];
    }
}else{
  $message = [
    'status' => 101,
    'message' => 'Gagal',
    'kontenData' => 'Akses tidak valid',
  ];
}

    $this->set_response($message, REST_Controller::HTTP_CREATED);
}
```

Jangan lupa bila jenis pengiriman yang digunakan adalah GET, oleh karena itu setelah menentukan nama fungsi pemanggilan tambahkan underscore (_) get. Pengambilan parameter tidak ada perbedaan yang mencolok hanya saja mengganti post menjadi get.

Pada fungsi pengambilan data konten privat yang ada pada controller akan ada fungsi yang dipanggil yaitu Authenticate. Fungsi ini akan dijelaskan pada tahap berikutnya yaitu Otentikasi API.

Fungsi pengambilan data konten publik pada controller:

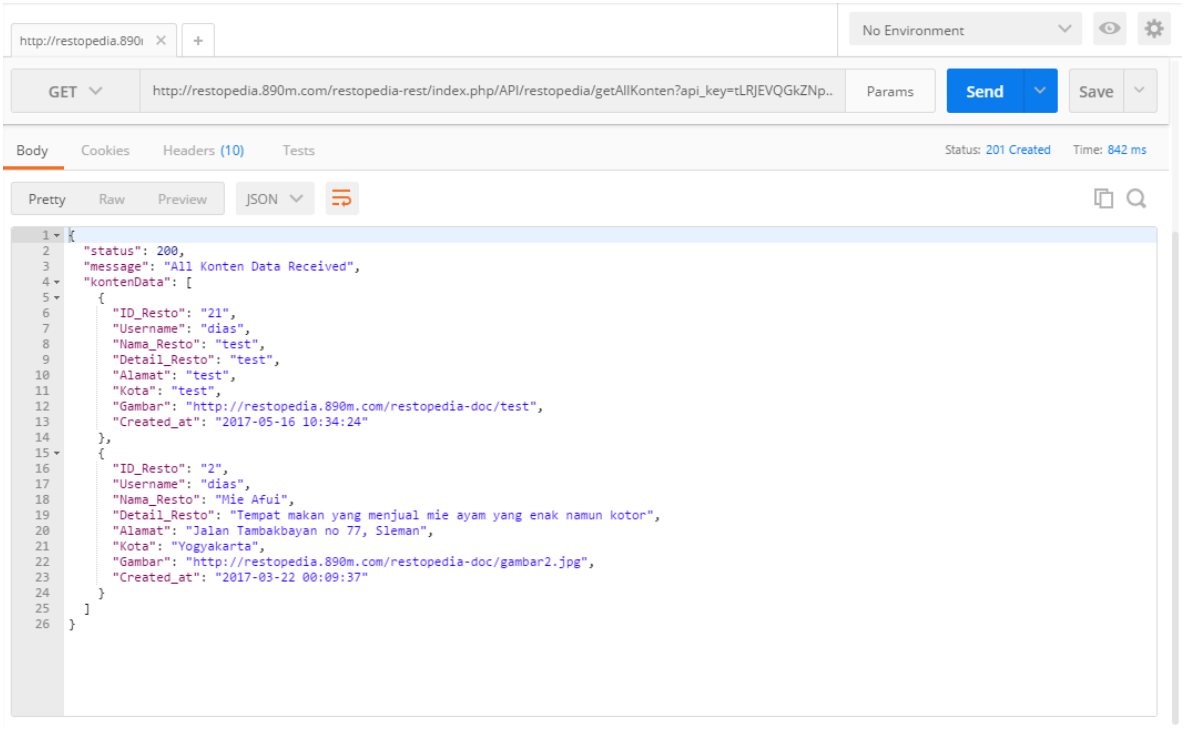
```
public function getAllKonten_get(){
$apikey=$this->get('api_key');

    $kontenData = $this->Konten_model->GetKonten();

if($this->User_model->Authenticate($apikey)){
    if($kontenData != NULL){
        $message = [
            'status' => 200,
            'message' => 'All Konten Data Received',
      'kontenData' => $kontenData,
        ];
    }else{
        $message = [
            'status' => 101,
            'message' => 'Konten Data Retrieve Failed',
      'kontenData' => 'Konten not Found',
        ];
    }
}else{
        $message = [
            'status' => 101,
            'message' => 'Gagal',
    'kontenData' => 'Akses tidak valid',
        ];
    }

    $this->set_response($message, REST_Controller::HTTP_CREATED);
}
```

Jika anda menggunakan Google Chrome maka anda dapat mengunduh aplikasi yang bernama Postman.

URL:
https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en

Aplikasi ini sangat membantu dalam proses testing API menjadi lebih cepat dan akurat.

Contoh keluaran yang dihasilkan oleh Postman:




Catatan : Pastikan seluruh API telah berjalan dengan sempurna sebelum melanjutkan ke tahap berikutnya.

## 2.3. Otentikasi API

Otentikasi API aplikasi ini dilakukan untuk API yang bersifat publik yaitu dalam pengambilan konten.

Fungsi pemanggil controller otentikasi ini telah dilakukan pada pembuatan API pengambilan data konten private yang dijelaskan pada butir sebelumnya.

Fungsi model yang dipanggil dengan nama Authenticate memiliki isi seperti berikut:

```
function Authenticate($apikey){
    $this->db->select('*');
    $this->db->from('user');
    $this->db->where('api_key',$apikey);

    $query = $this->db->get();

    if($query ->num_rows()==1)
    {
        $this->db->set('total_request', 'total_request+1', FALSE);
        $this->db->where('api_key', $apikey);
        $this->db->update('user');
        return true;
    }
    else
    {
        return false;
    }
}
```

Fungsinya adalah memastikan apakah API KEY pemanggil telah terdaftar sebelumnya pada database user_api. Jika ada maka tambahkan jumlah requestnya sebagai bantuan ketika kita ingin melihat jumlah pemanggilan yang dilakukan oleh masing-masing pengguna.

# Bab 3 Pengaksesan API dengan Android

## 3.1. Instalasi Library

Selanjutnya kita akan mengakses API yang telah dibuat dengan menggunakan aplikasi Android Restopedia.

Android memiliki berbagai macam library salah satunya ada library yang membantu pembuat aplikasi untuk melakukan akses ke API yang bersifat REST lebih cepat dan mudah dibandingkan dengan menggunakan Asynctask atau kelas JSONParser yang ada dari bawaan android.

Library yang akan digunakan disini memiliki nama Retrofit. Retrofit kini telah sampai di versi 2.0 yang dapat diakses dokumentasi dan cara instalasinya di https://inthecheesefactory.com/blog/retrofit-2.0/en

Bila dikompilasi cara penginstalan secara sederhananya adalah sebagai berikut:

1. Tambahkan Library berikut di build.gradle (module:app) anda.

```
compile 'com.squareup.retrofit2:retrofit:2.+'
compile 'com.squareup.retrofit2:converter-gson:2.+'
```

2. Buat kelas baru dengan nama ApiClient

3. Isi kelas ApiClient seperti berikut dengan mengganti baseUrl dengan baseUrl anda.

```
public class ApiClient {
    public static final String BASE_URL = "http://restopedia.890m.com/restopedia-rest/index.php/API/restopedia/";
    private static Retrofit retrofit = null;

    public static Retrofit getClient() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
        }
        return retrofit;
    }
}
```

Setelah 3 langkah tersebut telah dikerjakan, hal selanjutnya yang perlu anda lakukan adalah membuat model masing-masing hasil kembalian API anda dan memanggilnya di kelas yang anda inginkan. Lebih lanjut akan dipelajari pada pokok bahasan berikutnya.

### 3.1.1. Pembuatan User Interface

Demi kenyamanan pengguna dalam menggunakan aplikasi Restopedia, tentu tampilan dari aplikasi ini juga harus menarik sehingga meningkatkan tingkat ketertarikan pengguna.

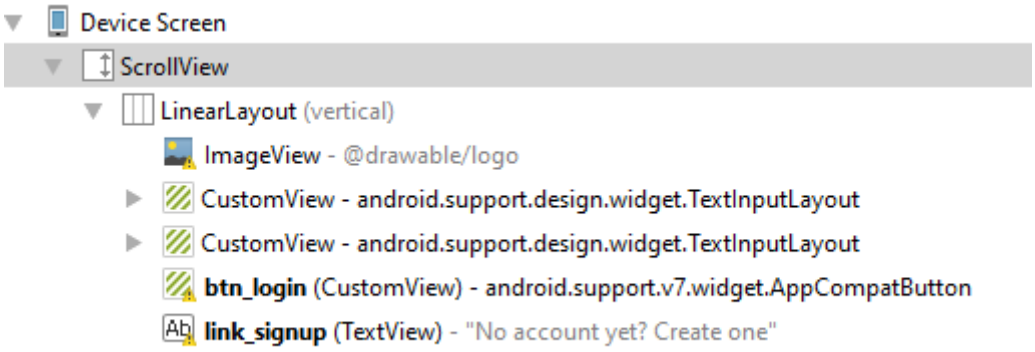
### 3.1.2. Tampilan Login dan Daftar

Pembuatan tampilan Login dan Daftar tidak memiliki struktur yang begitu rumit. Hal ini dikarenakan tampilan ini berbentuk lurus saja kebawah tanpa ada objek di kiri atau kanan dari objek lain. Oleh karena itu disini kita akan menggunakan LinearLayout(vertical).

Sebelumnya kita perlu menginstal library desain yang disediakan oleh android dengan menambahkan kode berikut pada build.gradle(Module:app) anda.

```
compile 'com.android.support:design:23.4.0'
```

Berikut adalah component tree dari Tampilan Login.




CustomView adalah layout yang merupakan bagian library yang sebelumnya dimasukkan ke gradle. CustomView ini memberikan tampilan Edit Text yang lebih menarik dari Edit Text default android.

Contoh kode dengan menggunakan CustomView Android Support Text Input Layout:

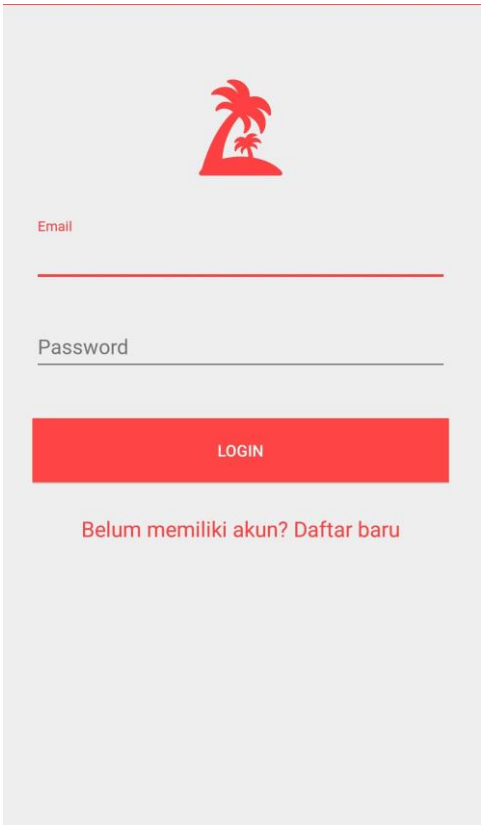
```
<!--  Username Label -->
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp">
    <EditText android:id="@+id/input_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Username" />
</android.support.design.widget.TextInputLayout>
```

ImageView diisi dengan gambar dari drawable yang dinamakan logo. logo.png adalah gambar dari Logo yang digunakan oleh Restopedia.

```
<ImageView android:src="@drawable/logo"
    android:layout_width="400dp"
    android:layout_height="72dp"
    android:layout_marginBottom="24dp"
    android:layout_gravity="center_horizontal"
    android:layout_weight="0.07"/>
```

Selanjutnya adalah tinggal mengatur margin atau padding sehingga tampilan terlihat lebih rapi dan menarik.

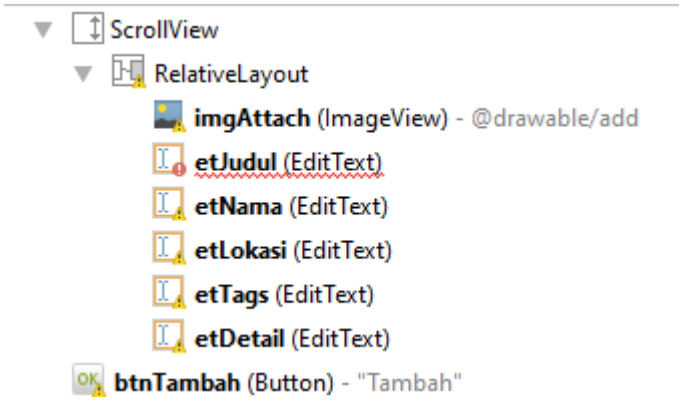
Hasil Akhir Tampilan Login dan Daftar:



### 3.1.3. Tampilan Daftar Konten

Pada tampilan ini tidak memakai library untuk mempercantik layout tambahan. Kita akan mencoba untuk membuat tampilan tetap seperti yang ada pada dokumen desain android dengan memanfaatkan objek objek yang disediakan android secara default.

Berikut adalah component tree dari tampilan daftar konten:




Pada component tree tersebut terlihat ada ScrollView sebelum diletakkan RelativeLayout. ScrollView berfungsi agar tampilan anda dapat di scroll ke atas dan kebawah jika layout melebih batas layar.

Hal yang dapat menyebabkan layout ini melebihi batas layar adalah karena ImageView dengan nama imgAttach memiliki height yang di set “Wrap Content”. Artinya tinggi dari layout ini tergantung dari gambar yang akan dimasukkan kedalamnya.

```
<ImageView
    android:id="@+id/imgAttach"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/rectangle"
    android:scaleType="center"
    android:visibility="visible"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_below="@+id/etDetail"
    android:layout_marginTop="20dp"
    android:src="@drawable/add"
    android:layout_marginBottom="50dp" />
```

Untuk scrollview sendiri diletakkan seperti berikut:

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
```

Catatan: Hati-hati dalam penggunaan ScrollView karena lokasi dari dimana diletakannya benar-benar akan berpengaruh pada layout tersebut.

Kemudian kita akan membuat EditText default dari Android terlihat seperti memiliki border berwarna merah.


Untuk melakukan hal ini anda perlu membuat XML baru di folder drawable. XML tersebut akan berisikan kode untuk membuat bentuk kotak yang digarisi garis berwarna merah untuk setiap objek yang diberikan nilai drawable tersebut pada backgroundnya.

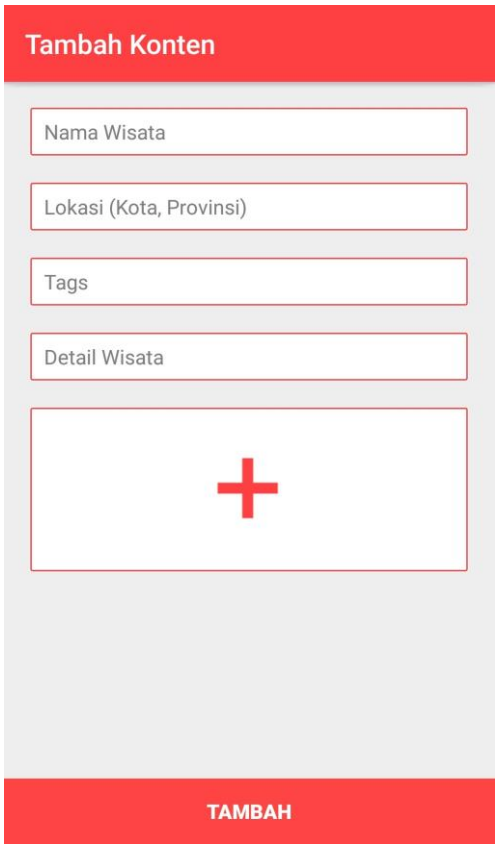
```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <!-- view background color -->
    <solid
        android:color="@color/white" >
    </solid>

    <!-- view border color and width -->
    <stroke
        android:width="1dp"
        android:color="@color/colorPrimary" >
    </stroke>

    <!-- The radius makes the corners rounded -->
    <corners
        android:radius="1dp"    >
    </corners>

</shape>
```

Anda bisa merubah radius dari corners untuk membuat ujung-ujung dari kotak lebih melengkung atua membulat dan tidak terlalu tajam dengan meningkatkan dp nya.

Setelah itu anda hanya perlu mengisi nilai background dari EditText yang ada ke drawable yang baru anda buat dengan menggunakan @drawable/namafile.XML

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/etNama"
    android:background="@drawable/rectangle"
    android:layout_below="@+id/etJudul"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="20dp"
    android:layout_marginTop="20dp"
    android:layout_marginRight="20dp"
    android:hint="Nama Wisata"
    android:paddingLeft="10dp"
    android:minHeight="35dp"
    android:textSize="15dp"
    android:singleLine="true" />
```

Hasil Akhir Tampilan Daftar Konten:




### 3.1.4. Tampilan Lihat Konten

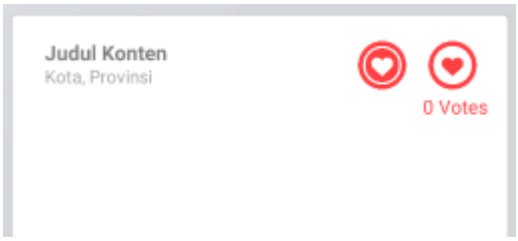
Untuk layout ini kita akan memerlukan drawable baru dan menambah beberapa warna pada color.xml.

```
<color name="white">#ffffff</color>
<color name="feed_bg">#d3d6db</color>
<color name="feed_item_bg">#ffffff</color>
<color name="feed_item_border">#c2c3c8</color>
<color name="link">#0a80d1</color>
<color name="timestamp">#a0a3a7</color>
```

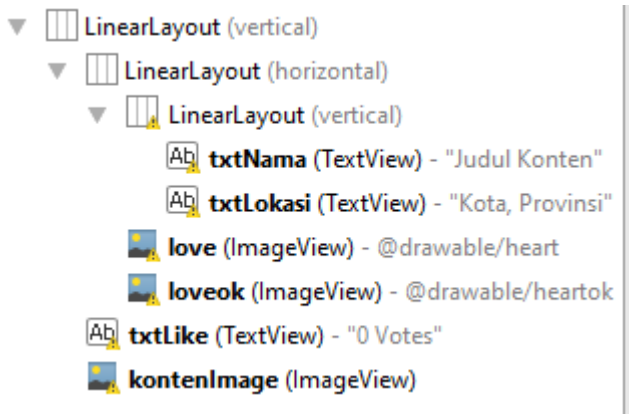
Kemudian buatlah file dengan ekstensi XML baru dan beri nama bg_parent_rounded_corner.xml yang memiliki isi sebagai berikut:

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <!-- view background color -->
    <solid android:color="@color/feed_item_bg" >
    </solid>

    <!-- view border color and width -->
    <stroke
        android:width="1dp"
        android:color="@color/feed_item_border" >
    </stroke>

    <!-- Here is the corner radius -->
    <corners android:radius="3dp" >
    </corners>

</shape>
```

Dilanjutkan dengan desain dan component tree dari layout konten item ini.




Untuk component tree sendiri memiliki kerumitan yang cukup sehingga dibutuhkan banyak LinearLayout didalamnya.




Dapat dilihat bahwa terdapat 2 macam ImageView yang bersebelahan yaitu dengan id love dan loveok. Pada saat muncul di tampilan konten maka salah satu dari gambar tersebut akan di sembunyikan tergantung dari pengguna tersebut apakah sudah melakukan vote atau belum pada konten yang bersangkutan.

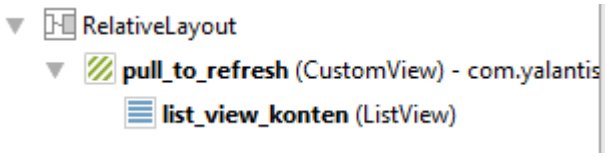
Untuk layout utama tidak memiliki desain hanya saja memanfaatkan library yang sudah ada sehingga mempercantik pull to refresh selain bawaan dari android sendiri.

Oleh karena itu pada build.gradle(Module:app) tambahkan kode berikut:

```
compile 'com.yalantis:phoenix:1.2.3'
```

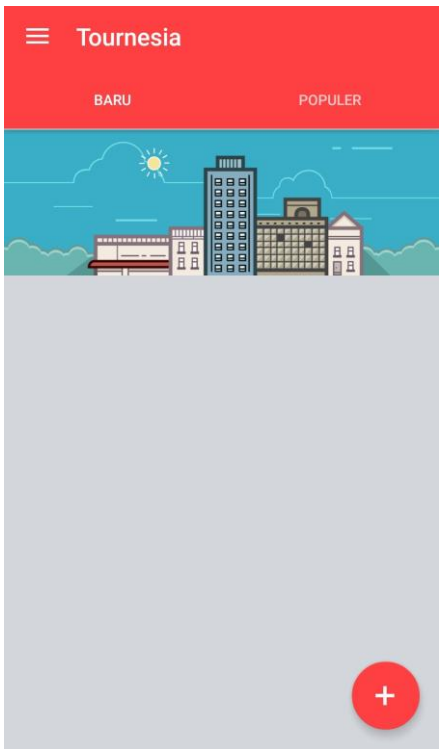
Dengan menambahkan library tersebut maka anda dapat mengakses CustomView baru dengan nama yalantis.phoenix.PullToRefreshView.

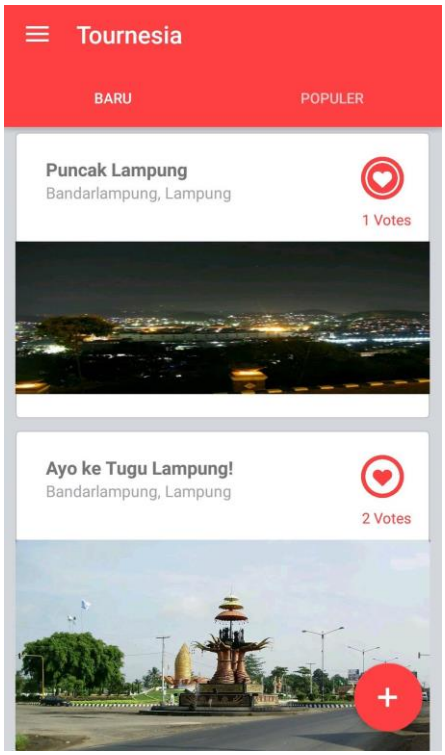
Gunakan library tersebut seperti yang terlihat pada component tree dan kode XML berikut:




```
<com.yalantis.phoenix.PullToRefreshView
    android:id="@+id/pull_to_refresh"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/list_view_konten"
        android:divider="@null"
        android:dividerHeight="0dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</com.yalantis.phoenix.PullToRefreshView>
```

Dengan demikian maka anda akan memperoleh hasil sebagai berikut:



## 3.2. Pengaksesan API

Pertama-tama anda memerlukan interface yang dapat menampung seluruh url API yang anda miliki. Buat kelas interface dan disimpan dengan nama ApiInterface.java untuk memudahkan.

```
public interface ApiInterface {

}
```

Perlu diketahui bahwa seluruh API akan memerlukan minimal satu buah model yang merupakan hasil dari balikan JSON nya.

Sebagai contoh seluruh API yang kita miliki memiliki hasil berupa status dan message yang bertipe data string. Maka model yang terbentuk adalah sebagai berikut:

```
public class StatusMessage {

    @SerializedName("status")
    @Expose
    private int status;
    @SerializedName("message")
    @Expose
    private String message;

    public int getStatus() { return status; }

    public void setStatus(int status) { this.status = status; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

}
```

Catatan : Nama variable pada model harus memiliki nama yang sama dengan parameter yang ada pada JSO

### 3.2.1. Akses API Daftar Pengguna

Karena API Daftar Pengguna hanya memiliki hasil berupa status dan message maka kita dapat menggunakan model StatusMessage yang telah dibuat sebelumnya.

Oleh karena itu pembuatan model tidak perlu dilakukan, langsung untuk mengisi interface dari API Daftar Pengguna itu sendiri yaitu dengan cara:

```
@FormUrlEncoded
@POST("signUp")
Call<StatusMessage> signUp(@Field("username") String username, @Field("password") String password, @Field("api_key") String api_key);
```

@POST menandakan tipe pengiriman yang dilakukan dari android dan @FormUrlEncoded berguna untuk mengirimkan parameter dengan tipe pengiriman POST.

“signUp” adalah link url pengiriman API yang dipakai.

Call<StatusMessage> adalah tempat untuk meletakan model hasil pemanggilan API yang disebutkan diatas. Retrofit akan secara otomatis melakukan parsing dari JSON ke model tersebut.

Selanjutnya parameter diletakan dengan menggunakan @Field diikuti dengan tipe data dari parameter tersebut.
Pindah ke kelas dimana anda ingin memanggil API tersebut dan gunakan kode sebagai berikut:

```
void userRegister(String username, String password) {
    ApiInterface apiService = ApiClient.getClient().create(ApiInterface.class);
    Call<StatusMessage> call = apiService.signUp(username, password, API_KEY);
    call.enqueue(new Callback<StatusMessage>() {
        @Override
        public void onResponse(Call<StatusMessage> call, Response<StatusMessage> response) {
            statusMessage = response.body();
            Log.i("Success", "Sign Up Success");
            onSignupSuccess();
        }

        @Override
        public void onFailure(Call<StatusMessage> call, Throwable t) {
            Log.i("Failed", "Sign Up Failed");
            onSignupFailed();
        }
    });
}
```

Bentuk variable service yang bertipe data ApiInterface dan callS dengan tipe data Call<StatusMessage> dan diberikan nilai beserta parameter username dan password seperti potongan code diatas.

Retrofit telah menyediakan 2 kondisi dimana ketika pemanggilan berhasil dan gagal. Berhasil juga terbagi atas 2 kondisi ketika kode response 200 atau 201 yang artinya sukses atau 400 dan seterusnya yang berarti ada yang salah.

### 3.2.2. Akses API Login Pengguna

Login pengguna memiliki hasil lain yaitu Userdatum. Userdatum berfungsi untuk mengetahui pengguna mana yang sedang login pada aplikasi android.

Maka kita perlu membuat model UserDatum tersebut.

```
public class UserDatum {
    @SerializedName("ID_User")
    @Expose
    private String iDUser;
    @SerializedName("username")
    @Expose
    private String username;
    @SerializedName("password")
    @Expose
    private String password;
    @SerializedName("api_key")
    @Expose
    private String apiKey;
    @SerializedName("total_request")
    @Expose
    private String totalRequest;
    public String getIDUser() { return iDUser; }
    public void setIDUser(String iDUser) { this.iDUser = iDUser; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public String getApiKey() { return apiKey; }
    public void setApiKey(String apiKey) { this.apiKey = apiKey; }
    public String getTotalRequest() { return totalRequest; }
    public void setTotalRequest(String totalRequest) { this.totalRequest = totalRequest; }
}
```

Gunakan UserLogin yang akan dipanggil dalam API Login.

```
public class UserLogin {

    @SerializedName("status")
    @Expose
    private int status;
    @SerializedName("message")
    @Expose
    private String message;
    @SerializedName("userData")
    @Expose
    private List<UserDatum> userData = null;

    public int getStatus() { return status; }

    public void setStatus(int status) { this.status = status; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

    public List<UserDatum> getUserData() { return userData; }

    public void setUserData(List<UserDatum> userData) { this.userData = userData; }

}
```

Setelah model berhasil terbentuk yang perlu anda lakukan selanjutnya adalah mendaftarkan API ke interface.

```
@FormUrlEncoded
@POST("login")
Call<UserLogin> login(@Field("username") String username, @Field("password") String password, @Field("api_key") String api_key);
```

Kemudian API tersebut tinggal dipanggil dengan mengunakan call dan service seperti butir sebelumnya.

```
void userLogin(String username, String password)
{
    ApiInterface apiService = ApiClient.getClient().create(ApiInterface.class);
    Call<UserLogin> call = apiService.login(username, password, API_KEY);
    call.enqueue(new Callback<UserLogin>() {
        @Override
        public void onResponse(Call<UserLogin> call, Response<UserLogin> response) {
            userLogin = response.body(); //data login-----------------------------------------
            Log.i("Success","Login Success");
            onLoginSuccess(userLogin.getUserData().get(0).getUsername());
        }

        @Override
        public void onFailure(Call<UserLogin> call, Throwable t) {
            Log.i("Failed","Login Failed");
            onLoginFailed();
        }
    });
```

### 3.2.3. Akses API Mendaftarkan Konten Baru

Mendaftarkan konten tidak memerlukan Model baru karena hasil dari API hanya berupa status dan message yang telah dibentuk modelnya yaitu APIBaseResponse.

Daftarkan API menambahkan konten anda di API Interface.

```
@FormUrlEncoded
@POST("addKonten")
Call<StatusMessage> addKonten(@Field("api_key") String api_key, @Field("username") String username,
                              @Field("nama") String nama, @Field("detail") String detail,
                              @Field("alamat") String alamat, @Field("kota") String kota,
                              @Field("gambar") String gambar, @Field("namagambar") String namagambar);
```

Berikan seluruh parameter yang dibutuhkan untuk menambahkan data konten sesuai dengan apa yang telah dibuat di REST server sebelumnya.

Karena gambar yang dikirimkan akan di kompresi dengan format base_64 oleh karena itu kode ini dibutuhkan untuk melakukan konversi gambar yang ada pada ImageView ke base_64 string.

```
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
((BitmapDrawable) img.getDrawable()).getBitmap().compress(Bitmap.CompressFormat.JPEG, 50, byteArrayOutputStream);
String gambar = Base64.encodeToString(byteArrayOutputStream.toByteArray(), Base64.DEFAULT);
```

img adalah nama variable dari ImageView dimana tempat user menyimpan gambar yang akan di unggah ke konten tempat makan yang akan dikirimkan.

```
ApiInterface apiService = ApiClient.getClient().create(ApiInterface.class);
Call<StatusMessage> call = apiService.addKonten(API_KEY, USERNAME, nama, detail, alamat, kota, gambar, namagambar);
call.enqueue(new Callback<StatusMessage>() {
    @Override
    public void onResponse(Call<StatusMessage> call, Response<StatusMessage> response) {
        statusMessage = response.body();
        Log.i("Success", "Upload Success");
        onUploadSuccess();
    }

    @Override
    public void onFailure(Call<StatusMessage> call, Throwable t) {
        Log.i("Failed", "Upload Failed");
        onUploadFailed();
    }
});
```

Letakan hasil konversi gambar yang telah di encode di parameter ketika akan memanggil API tersebut.

### 3.2.4. Akses API Mengambil Konten (Privat dan Publik)

Kita memerlukan model baru dengan nama KontenDatum yang berisikan data-data konten hasil kembalian dari API.

```
@SerializedName("ID_Resto")
@Expose
private String iDResto;
@SerializedName("Username")
@Expose
private String username;
@SerializedName("Nama_Resto")
@Expose
private String namaResto;
@SerializedName("Detail_Resto")
@Expose
private String detailResto;
@SerializedName("Alamat")
@Expose
private String alamat;
@SerializedName("Kota")
@Expose
private String kota;
@SerializedName("Gambar")
@Expose
private String gambar;
@SerializedName("Jumlah_Likes")
@Expose
private int jumlahLikes;
@SerializedName("Created_at")
@Expose
private String createdAt;
@SerializedName("Flag")
@Expose
private List<Flag> flag = null;

public String getIDResto() { return iDResto; }
public void setIDResto(String iDResto) { this.iDResto = iDResto; }
public String getUsername() { return username; }
public void setUsername(String username) { this.username = username; }
public String getNamaResto() { return namaResto; }
public void setNamaResto(String namaResto) { this.namaResto = namaResto; }
public String getDetailResto() { return detailResto; }
public void setDetailResto(String detailResto) { this.detailResto = detailResto; }
public String getAlamat() { return alamat; }
public void setAlamat(String alamat) { this.alamat = alamat; }
public String getKota() { return kota; }
public void setKota(String kota) { this. kota = kota; }
public String getGambar() { return gambar; }
public void setGambar(String gambar) { this.gambar = gambar; }
public int getJumlahLikes() { return jumlahLikes; }
public void setJumlahLikes(int jumlahLikes) { this.jumlahLikes = jumlahLikes; }
public String getCreatedAt() { return createdAt; }
public void setCreatedAt(String createdAt) { this.createdAt = createdAt; }
public List<Flag> getFlag() { return flag; }
public void setFlag(List<Flag> flag) { this.flag = flag; }
```

Daftarkan API untuk mendapatkan konten pada interface.

```
@GET("getAllKonten")
Call<UserKonten> getAllKonten(@Query("username") String username, @Query("api_key") String api_key);
```

Tahap terakhir gunakan adapter untuk melakukan populasi recyclerview pada setiap item list yang anda dapatkan di hasil JSON.

```
@Override
public void onResponse(Call<UserKonten> call, Response<UserKonten> response) {
    UserKonten userKonten = response.body(); //data Konten----------------------------------------
    List<KontenDatum> restoList = userKonten.getKontenData();
    //textView.setText(userKonten.getKontenData().get(0).getDetailResto());


    layoutManager = new LinearLayoutManager(TabActivity1.this);

    RecyclerView recyclerView = (RecyclerView)
            findViewById(R.id.recycler);
    recyclerView.setLayoutManager(layoutManager);

    RecyclerViewAdapter recyclerViewAdapter =
            new RecyclerViewAdapter(restoList);

    recyclerView.setAdapter(recyclerViewAdapter);

}

@Override
public void onFailure(Call<UserKonten> call, Throwable t) {
    for (int i = 0; i < 10; i++) {
        System.out.println("Load Track Failure Response");
    }
```

# Bab 4 Penutup

Demikianlah aplikasi layanan pengumpulan data tempat makan di Indonesia secara kolaboratif yang telah dibentuk. Semoga aplikasi ini dapat memberikan tujuan yang diinginkan yaitu memberikan nama baik lebih untuk Indonesia dari segi tempat makan.

Tentunya kami juga mengucapkan banyak terima kasih kepada berbagai pihak yang telah membantu terwujudnya aplikasi ini. Kiranya tutorial ini dapat membantu pembaca dan pengguna aplikasi Restopedia ini di kemudian hari.