**BAB 4**

**IMPLEMENTASI DAN UJI COBA**

**4.1 Implementasi**

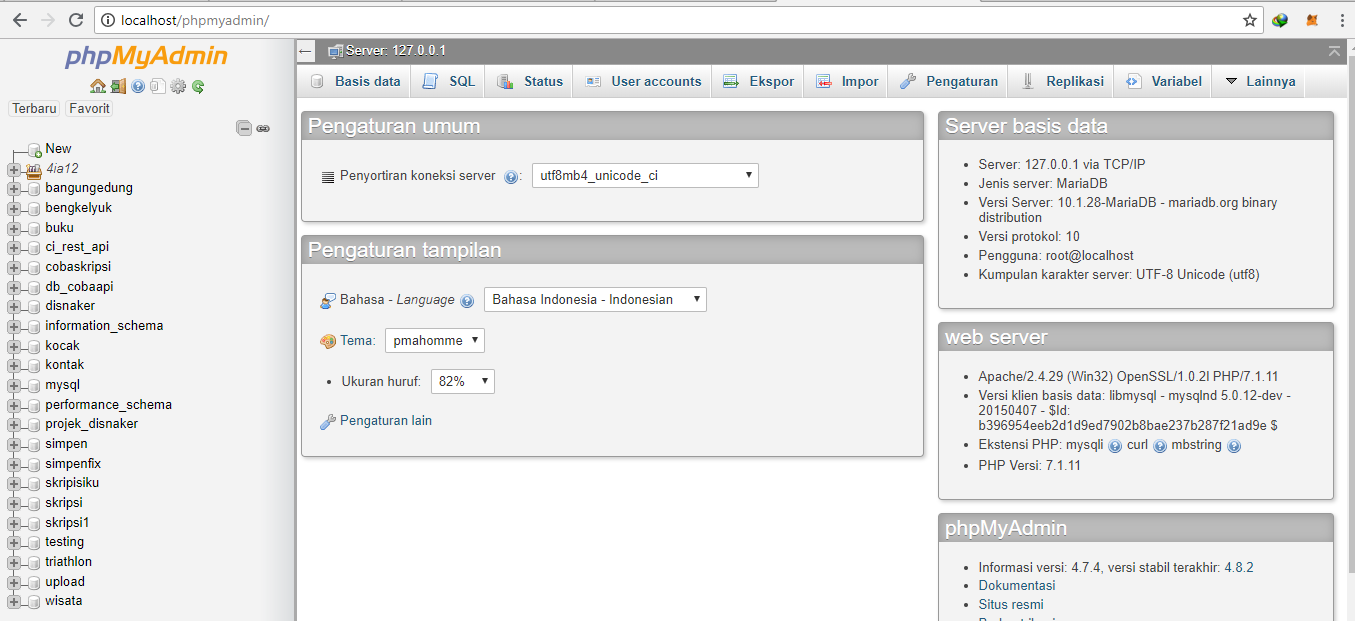
Implementasi adalah sebuah tahapan pembuatan android dengan menggunakan Bahasa pemrograman setelah melakukan perancangan aplikasi dengan UML dan struktur navigasi. Dalam pembuatan aplikasi ini, menggunakan Bahasa pemrograman Java untuk fungsi utama, XML untuk desain pada aplikasi, Codeigniter sebagai framework untuk website dan SQL sebagai database dangan Bahasa PHP. Implementasi android menggunakan Android Studio. Setelah tahapan implementasi, tahap pengujian dilkakuan untuk menilai kesesuaian kinerja aplikasi dengan percangan..

**4.2 Pembuatan Basis Data (*Database)***

1. Database akan menyimpan semua data yang berhubungan dengan identitas bangun gedung dan yang akan ditampilkan di aplikasi survei bangun gedung. Baik berupa teks ataupun posisi pada *maps*. Berikut merupkan langkah-langkah dalam pembuatan *database* aplikasi :

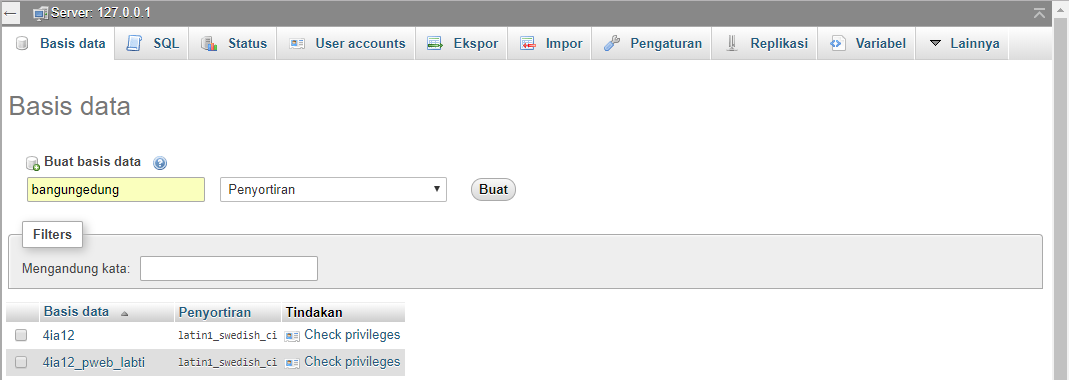
a. pada aplikasi ini, penulis menggunakan localhost pada komputer penulis sebagai server aplikasi.

b. penulis mengakses alamat localhost dan masuk kedalam pembuatan database seperti gambar 4.1.



**Gambar 4.1** Tampilan Home *Database* Localhost phpMyAdmin

c. Langkah selanjutnya penulis memilih *new* yang terdapat di menu sebelah kiri untuk pembuatan *database* baru. Pembuatan *database* baru dapat dilihat pada gambar 4.2.

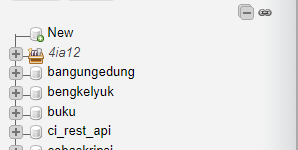


**Gambar 4.2** Tampilan Pembuatan *Database* Baru

2. Setelah pembuatan *database,* penulis membuat table baru ke dalam *database* yang telah dibuat. Berikut langkah-langkah dalam pembuatan tabel baru :

a. Penulis masuk ke halaman utama Localhost dan memilih *menu* phpMyAdmin pada gambar 4.1.

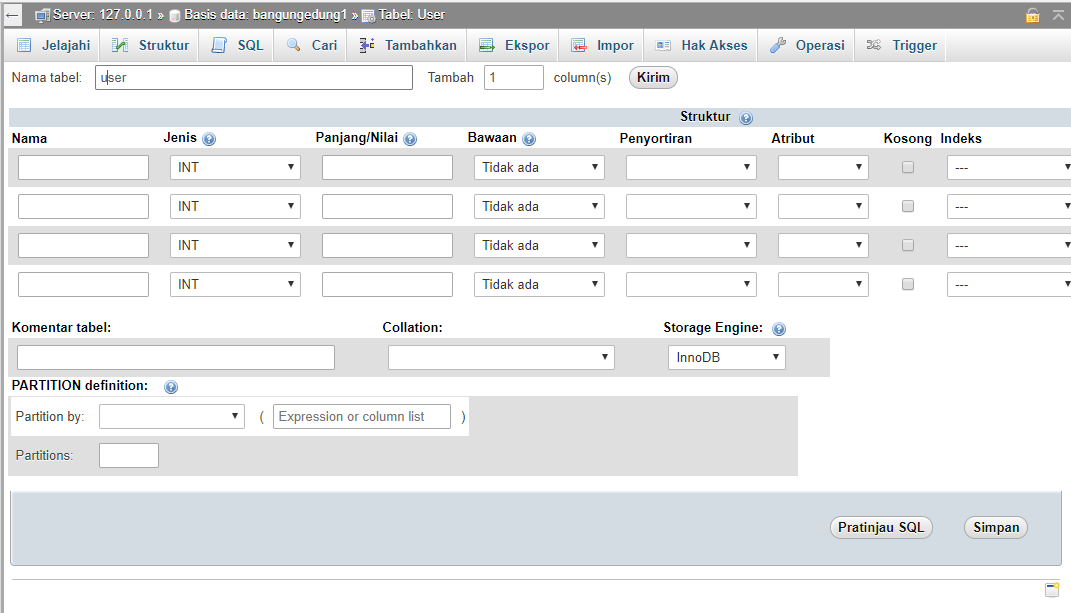
b. Pada halaman phpMyAdmin berisi daftar *database* yang telah tersimpan di dalam sistem. Halaman phpMyAdmin dapat dilihat pada gambar 4.3 (a), kemudian penulis memilih *new* pada gambar 4.4 (b) untuk membuat tabel baru.

1. (b)

**Gambar 4.3** (a) Daftar *Database,* (b) *Menu* Menambahkan Tabel

c. Kemudian penulis membuat tabel baru dan mendefinisikan isi tabel. *Nama* Tabelmerupakan *field* yang berisi nama tabel yang akan ditambahkan dan Tambahakan berisi jumlah kolom yang terdapat di dalam tabel. Tampilan pendefinisian struktur tabel dapat dilihat pada gambar 4.4.

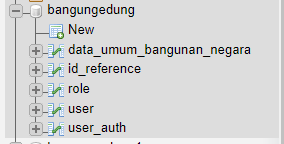


**Gambar 4.4** Pembuatan Tabel Baru

d. Setelah itu penulis mengulang langkah 2a sampai dengan 2c untuk pembuatan tabel lainnya jika dibutuhkan. Pembuatan serta pendefinisian isi tabel sesuai dengan rancangan tabel pada sub bab 3.3.4.

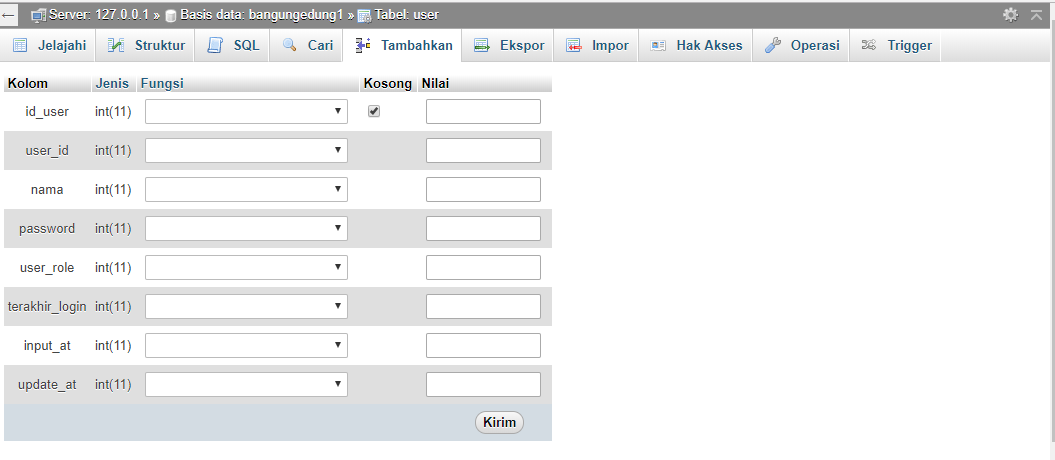
3. Setelah membuat tabel baru, penulis akan mengisi semua *record* sesuai dengan struktur tabel yang telah didefinisikan. Berikut ini langkah- langkah yang dilakukan :

a. Penulis memilih tabel yang telah dibuat, daftar tabel yang ada di dalam *database* yang penulis sudah buat yaitu tabel user, user\_auth, data\_bangun\_gedung\_negara, id\_reference, dan role dapat dilihat pada gambar 4.5

.

**Gambar 4.5** Tampilan Daftar Tabel

b. Setelah itu penulis akan dibawa pada tampilan gambar 4.6 dan memilih *tab* Tambahkan*.* Pada halaman tersebut akan berisi semua *field* sesuai dengan tabel yang telah dibuat pada langkah 2. *Value* yang kosong akan diisikan oleh semua *record* terkait. Tombol kirimdipilih untuk menyimpan perubahan yang telah dilakukan.



**Gambar 4.6** Tampilan Memasukan *Record*

**4.3 Pembuatan Aplikasi**

Setelah rancangan tampilan dibuat, struktur navigasi, diagram uml dan *pseudocode* program tahap ini merupakan tahap pembuatan aplikasi berdasarkan perancangan yang telah dibuat.

**4.3.1 Pembuatan Web Admin**

1. Langkah pertama pada pembuatan admin dalam web ini adalah menggunakan bahasa pemrograman web, dengan menggunakan html dan codeigniter sebagai framework dan menggunakan konsep MVC( Model View Controller).

Berikut listing program untuk Controller pada website :

**Admin.php**

|  |
| --- |
| <?php  defined('BASEPATH') OR exit('No direct script access allowed');  class Admin extends CI\_Controller {  public function \_\_construct(){  parent::\_\_construct();  $this->load->library('session');  $this->load->model('Model\_Web');  }  public function index()  {  }  public function dashboard(){  $this->load->library('session');  if($this->session->userdata('logged\_in')==true){  $this->load->view('Templates/HeaderDash');  $this->load->view('Main/Dashboard');  $this->load->view('Templates/FooterDash');  }else{  redirect('LoginWeb');  } |

Pada kode fungsi diatas yaitu *function dashboard* digunakan untuk memunculkan halaman *dashboard* dengan kondisi apabila *array* userdata bernilai *true* maka akan memunculkan *file* Headerdash, Dashboard, dan footerdash yang terdapat pada *folder view* di dalam codeigniter. Apabila *userdata* bernilai *false* maka *controller* akan mengarahkan kembali ke tampilan LoginWeb.

|  |
| --- |
| public function halaman\_register(){  $this->load->library('session');  if($this->session->userdata('logged\_in')==true){  $data['option']='add';  $this->load->view('Templates/HeaderDash');  $this->load->view('Main/Register',$data);  $this->load->view('Templates/FooterDash');  }else{  redirect('LoginWeb');  }  } |

Berdasarkan fungsi kode diatas yaitu *function* halaman\_register digunakan digunakan untuk membuka halaman pendaftaran akun baru. Salah satu baris kode diatas $data['option']='add' berfungsi sebagai parameter yang menandakan bahwa halaman ini digunakan untuk menambah akun baru.

|  |
| --- |
| public function register(){  $id=$this->input->post('id');  $role=$this->input->post('role');  $nama\_depan=$this->input->post('nama\_depan');  $nama\_belakang=$this->input->post('nama\_belakang');  $user\_id=$this->input->post('user\_id');  $name= "{$nama\_depan} {$nama\_belakang}";  $password\_input=$this->input->post('password');  $password=crypt($password\_input,'');  $data=array(  'user\_id'=>$user\_id,  'nama'=>$name,  'password'=>$password,  'user\_role'=>$role,  'input\_at'=>date('Y-m-d H:i:s'),  );  $this->Model\_Web->register\_user($data);  redirect('Admin/tabel\_user');  } |

Berdasarkan fungsi kode *register,* digunakan sebagai fungsi untuk memasukan data dengan method *POST* pada *view* halaman\_register. Data yang dimasukan pada halaman tersebut diantaranya id, *role*, nama depan, nama belakang dan *password.* Fungsi pada baris kode diatas digunakan untuk enskripsi kata atau kalimat yang telah di masukan. Fungsi *crypt* tersebut telah disediakan oleh codeigniter .

|  |
| --- |
| public function edit(){  $id=$this->input->post('id');  $role=$this->input->post('role');  $user\_id=$this->input->post('user\_id');  $name= $this->input->post('nama');  $password\_input=$this->input->post('password');  $password=crypt($password\_input,'');  $data=array(  'user\_id'=>$user\_id,  'nama'=>$name,  'password'=>$password,  'user\_role'=>$role,  'update\_at'=>date('Y-m-d H:i:s')  );  $this->Model\_Web->update($id,$data); redirect('Admin/tabel\_user');  } |

Berikut kode diatas adalah untuk mengeditakun yang telah dibuat sebelumnya pada fungsi *register* dan pada halaman\_register. Baris kode tersebut memiliki fungsi yang sama yaitu dengan menggunakan method *POST* untuk memasukan data pada database. Pada baris kode $this->Model\_Web->update($id,$data); digunakan untuk memanggil model untuk *update* data akun yang telah terdaftar.

|  |
| --- |
| public function edit\_user($id){  if ($this->session->userdata('logged\_in')==true){  $data['option']='edit';  $data['sql'] = $this->Model\_Web->edit($id);  $this->load->view('Templates/HeaderDash');  $this->load->view('Main/Edit',$data);  $this->load->view('Templates/FooterDash');  }else {  redirect('LoginWeb'); |

Berdasarkan fungsi kode diatas yaitu *function edit* digunakan digunakan untuk membuka halaman edit akun. Salah satu baris kode diatas $data['option']='edit' berfungsi sebagai parameter yang menandakan bahwa halaman ini digunakan untuk mengubah akun yang telah terdaftar.

|  |
| --- |
| public function delete\_user($id){  $data['sql'] = $this->Model\_Web->delete($id);  redirect('Admin/Tabel\_User');  } |

Pada kode fungsi *delete\_user* digunakan untuk menghapus data akun yang telah terdaftar pada database, pada baris kode Model\_Web->delete($id); digunakan untuk memanggil *model* yang terdapat pada Model\_Web dan menggunakan method *DELETE* untuk menghapus data akun tersebut.

|  |
| --- |
| public function tabel\_user(){  $this->load->library('session'){  if ($this->session->userdata('logged\_in')==true){  $data['sql'] = $this->Model\_Web->get\_user();  $this->load->view('Templates/HeaderDash');  $this->load->view('Main/Table\_User',$data);  $this->load->view('Templates/FooterDash');  }else{  redirect('LoginWeb');  }  } |

Dari kode diatas yaitu fungsi tabel\_user digunakan untuk menampilkan tabel data user yang telah dibuat pada halaman\_register pada *website*. Pada kode baris kode $data['sql'] = $this->Model\_Web->get\_user(); tersebut berfungsi untuk memanggil model get\_user yang terdapat pada Model\_Web dan menggunakan method *GET* untuk mengambil data akun yang terdapat pada basis data.

|  |
| --- |
| public function tabel\_gedung(){  $this->load->library('session');  if ($this->session->userdata('logged\_in')==true){  $data['sql'] = $this->Model\_Web->get\_gedung();  $this->load->view('Templates/HeaderDash');  $this->load->view('Main/Table\_Gedung',$data);  $this->load->view('Templates/FooterDash');  }else{ redirect('LoginWeb');  }  }  } |

Dari kode diatas yaitu fungsi tabel\_gedung digunakan untuk menampilkan tabel data user yang telah dibuat pada halaman\_register pada *website*. Pada kode baris kode $data['sql'] = $this->Model\_Web->get\_gedung(); tersebut berfungsi untuk memanggil model get\_gedung yang terdapat pada Model\_Web dan menggunakan method *GET* untuk mengambil gedung yang terdapat pada basis data.

**LoginWeb.php**

|  |
| --- |
| <?php  defined('BASEPATH') OR exit('No direct script access allowed');  class LoginWeb extends CI\_Controller {  public function \_\_construct(){  parent::\_\_construct();  $this->load->model('Model\_Web');  } |

Pada kode diatas berfungsi untuk memanganggil model pada Model\_web sebagai konsturktor yang berarti akan terus dijalan setiap program atau web dijalankan.

|  |
| --- |
| public function index()  {  if ($this->session->userdata('logged\_in'))  {  redirect('Admin/dashboard','refresh');  }else{  $this->load->helper(array('form'));  $this->load->view('Templates/Header');  $this->load->view('Main/Login');  $this->load->view('Templates/Footer');  }  } |

Berdasarkan kode *function* index diatas, kode tersebut digunakan untuk pemanggilan halaman utama pada saat website dijalankan yaitu memanggil Main/Login sebagai index dari website.

|  |
| --- |
| public function login\_check(){  $user\_id = $this->input->post('user\_id',true);  $password = $this->input->post('password',true);  $this->Model\_Web->login\_check($user\_id,$password);  } |

Pada kode diatas fungsi dari login\_check digunakan untuk apakah *id* dan *password* yang dimasukan kedalam *form login* benar atau salah.

|  |
| --- |
| public function logout(){  $this->session->unset\_userdata('logged\_in');  redirect('LoginWeb','refresh');  }  } |

Berdasarkan kode diatas fungsi *logout* digunakan untuk keluar dari sebuah akun pada website.

**Auth.php**

*Controller* digunakan sebagai penghubung antara *view* dengan model pada *website*, setelah *controller* semua sudah dibuat, berikut model untuk menampilkan basis data yang sudah dibuat dan yang ditampilkan oleh *view* pada *website.* Model ini menggunakan bahasa yang digunakan pada basis data yang digunakan

Berikut listing program untuk model untuk website :

**Model\_Web.php**

Model\_Web.php ini model yang hanya digunakan untuk akses pada *website*. Berikut listing program pada Model\_Web.php

|  |
| --- |
| <?php  defined('BASEPATH') OR exit('No direct script access allowed');  class Model\_Web extends CI\_Model{  function login\_check($user\_id,$password)  {  $q=$this->db->select('password,id\_user')->from('User')->where('user\_id',$user\_id)->get()->row();  $hashed\_password = $q->password;  if(hash\_equals($hashed\_password,crypt($password,$hashed\_password)) && $user\_id=='50414797') {  print\_r(hash\_equals($hashed\_password,crypt($password,$hashed\_password)));  $sess\_array = array();  $sess\_array = array('username' => $q->user\_id, 'logged\_in' => true);  $this->session->set\_userdata($sess\_array);  $status = array('status' =>true);  }  redirect('Admin/dashboard');  } |

pada kode model login\_check digunakan untuk mengecek id dan *password* yang telah di enskripsi apakah benar atau salah dengan data pada basis data. Apabila *login* berhasil dan berhasil maka akan diarahkan kedalam halaman admin/dashboard.

|  |
| --- |
| public function register\_user($data){  return $this->db->insert('user',$data);  } |

Pada kode model diatas yaitu *function* register\_user digunakan untuk memasukan data akun kedalam basis data.

|  |
| --- |
| public function edit($id){  $this->db->where('id\_user',$id);  return $this->db->get('user');  } |

Berdasarkan kode model diatas yaitu function *edit* digunakan untuk mengambil data akun yang terdapat pada basisdata.

|  |
| --- |
| public function update($id,$data){  $this->db->where('id\_user',$id);  $this->db->update('user', $data);  return $this->db->affected\_rows();  } |

Kode function *update* diatas digunakan untuk merubah data akun yang telah ada pada basis data.

|  |
| --- |
| public function delete($id){  $this->db->where('id\_user',$id);  return $this->db->delete('user');  } |

Pada kode function *delete* diatas digunakan untuk menghapus akun user yang telah terdaftar pada basis data dengan menggunakan id sebagai parametenya.

|  |
| --- |
| public function get\_user(){  $sql = $this->db->query("select \* from user");  return $sql;  } |

Berdasarkan kode dari *function* get\_user diatas digunakan untuk mengambil seluruh data akun yang ada pada basis data.

|  |
| --- |
| public function get\_gedung(){  $sql=$this->db->query("select\*from data\_umum\_bangunan\_negara");  return $sql;  } |

Berdasarkan kode dari *function* get\_gedung diatas digunakan untuk mengambil seluruh data gedung yang ada pada basis data.

**MyModel.php**

MyModel.php ini model yang digunakan untuk android dalam akses basis data yang telah dibuat pada *website*. Berikut listing kodenya :

|  |
| --- |
| <?php  defined('BASEPATH') OR exit('No direct script access allowed');  class MyModel extends CI\_Model {  var $client\_service = "frontend-client";  var $auth\_key = "simplerestapi";  public function check\_auth\_client(){  $client\_service = $this->input->get\_request\_header('Client-Service', TRUE);  $auth\_key = $this->input->get\_request\_header('Auth-Key', TRUE);  if($client\_service == $this->client\_service && $auth\_key == $this->auth\_key){  return true;  } else {  return json\_output(401,array('status' => 401,'message' => 'Unauthorized.'));  }  } |

Kode diatas adalah memberikan header pada saat android ingin mengakses database survei bangun gedung. Custom header tersebut digunakan untuk memberikan hak akses secara unik kepada setiap akun. Kode diatas adalah pengecekan apakah header client\_service dan auth\_key terdapat pada android pada saat mengakses data. Jika terdapat pada android maka client\_service dan auth\_key bernilai true. Jika tidak maka akan memberikan informasi bahwa akun tersebut *Unathorized.*

|  |
| --- |
| public function login($username,$password)  {  $q = $this->db->select('password,id\_user,user\_id,nama,user\_role')->from('User')->where('user\_id',$username)->get()->row();  if($q == ""){  return array('status' => 204,'message' => 'Username not found.');  } else {  $nama = $q->nama;  $role = $q->user\_role;  $hashed\_password = $q->password;  $id = $q->id\_user;  $user\_id =$q->user\_id; |

Pada kode diatas digunakan untuk *login* pada android dengan menggunakan parameter username dan password. Kode diatas membandingkan antara nama, *role,* password yang telah di hash, id dan user id pada tabel user. Dan apabila pada saat *login* salah satunya kosong maka akan memberikan pesan username tidak ditemukan.

|  |
| --- |
| if (hash\_equals($hashed\_password, crypt($password, $hashed\_password))) {  $last\_login = date('Y-m-d H:i:s');  $token = crypt(substr( md5(rand()), 0, 7),'');  $expired\_at = date("Y-m-d H:i:s", strtotime('+12 hours'));  $this->db->trans\_start();  $this->db->where('id\_user',$id)->update('User',array('terakhir\_login' => $last\_login));  $this->db->insert('user\_auth',array('id\_user' => $id,'token' => |

Lanjutan dari kode *function login* apabila kondisi *user* sudah terdapat *login* pada aplikasi, maka pada saat *splashscreen* akan di cek baris data user tersebut. mulai dari last\_login, token, Expired\_at, expired\_at digunakan untuk memberikan *session* selama 12 jam kepada setiap *user* android yang telah *login* pada aplikasi.

|  |
| --- |
| $token,'expired\_at' => $expired\_at));  if ($this->db->trans\_status() === FALSE){  $this->db->trans\_rollback();  return array('status' => 500,'message' => 'Internal server error.');  } else {  $this->db->trans\_commit();  return array('status' => 200,'message' => 'Successfully login.','id' => $id, 'token' => $token, 'nama'=> $nama, 'role'=> $role, 'user\_id'=> $user\_id);  }  } else {  echo "Wrong password";  exit();  return array('status' => 204,'message' => 'Wrong password.');  } |

Berdasarkan kode diatas apabila token sudah expired maka akan kembali rollback ke model *login,* dan apabila token masih kurang dari 12 jam dari terakhir login maka akan memberikan pesan *successfully login* dan memberikan data id, token, nama, tole dan user\_id pada saat login di android.

|  |
| --- |
| public function logout()  {  $users\_id = $this->input->get\_request\_header('User-ID', TRUE);  $token = $this->input->get\_request\_header('Authorization', TRUE);  $this->db->where('ID\_USER',$users\_id)->where('token',$token)->delete('user\_auth');  return array('status' => 200,'message' => 'Successfully logout.');  } |

Pada kode daitas model *function logout* digunakan untuk *logout* pada android, pada kode diatas *user* akan meminta custom header yang telah dibuat, User-ID yaitu *id* pengguna serta meminta header *Authorization* sebagai token pada tabel *user.* Apabila keduanya bernilai benar maka akan menghapus 1 baris pada tabel user\_auth. Setelah terhapus akan memberikan pesan berupa array berisi “*Succesfully logout”.*

|  |
| --- |
| *public function input\_data\_bangunan($data){*  *$this->db->insert('data\_umum\_bangunan\_negara',$data);*  *return array('status' => 200,'message' => 'Data has been created.');*  *}* |

Kode diatas adalah model data bangunan yang digunakan untuk memasukan hasil survei kedalam data dengan menggunakan *method insert,* dan apabila data telah berhasil masuk kedalam basis maka akan memberikan sebuah pesan “*data has been created”*

|  |
| --- |
| *public function hapus\_data\_umum\_bangunan\_negara($id\_bangunan)*  *{*  *$this->db->where('id\_bangunan',$id\_bangunan)->delete('data\_umum\_bangunan\_negara');*  *return array('status' => 200,'message' => 'Data has been deleted.');*  *}* |

Fungsi kode diatas digunakan untuk menghapus data pada basis data, dengan menggunakan method *delete*, dan apabila data telah berhasil terhapus di android maka akan memberikan pesan “data has been deleted” pada *website*.

|  |
| --- |
| public function list\_bangunan\_user(){  $this->db->select('u.nama,p.\*');  $this->db->from('data\_umum\_bangunan\_negara as p');  $this->db->join('user as u','u.user\_id = p.user\_id','inner');  $this->db->order\_by('nama\_bangunan');  return $this->db->get()->result();  }  } |

Berdasarkan kode diatas adalah berfungsi sebagai model untuk menampilkan seluruh data yang terdapat pada data\_bangunan\_umum\_negara dengan menggunakan *method get*. Model tersebut menggunakan *inner join* untuk menyatukan nama yang terdapat pada tabel *user*. Nama tersebut diambil agar dapat ditampilkan nama petugas survei yang memasukan data bangunan gedung dengan menggunakan aplikasi survei bangun gedung.

|  |
| --- |
| public function list\_bangunan\_petugas survei($user\_id){  $this->db->select('u.nama,p.\*');  $this->db->from('data\_umum\_bangunan\_negara as p');  $this->db->join('user as u','u.user\_id = p.user\_id','inner');  $this->db->where('p.user\_id', $user\_id);  $this->db->order\_by('nama\_bangunan');  return $this->db->get()->result();  } |

Berikutnya adalah kode dari *function* list\_bangunan\_survey, fungsi tersebut berfungsi untuk menampilkan hanya yang dimasukan oleh petugas survei yang bersangkutan dengan menggunakan *method get,*, dengan menggunakan parameter user\_id yang tentunya unik sehingga dapat dipisahkan dari seluruh data bangunan. Model tersebut menggunakan *inner join* dengan nama yang terdapat pada tabel *user*.

|  |
| --- |
| public function cari\_bangunan($kata){  $this->db->like('nama\_bangunan',$kata);  $query = $this->db->get('data\_umum\_bangunan\_negara');  return $query->result();} |

Yang terakhir pada kode MyModel.php adalah fungsi cari\_bangunan.. fungsi tersebut digunakan untuk mencari dan mendapatkan data meggunakan *methode get,* data bangun gedung pada android yang berupa *list* daftar dari nma departemen dan nama bangunan. Pencarian tersebut dapat dilakukan dengan mencari nama bangunan, fungsi bangunan dan nama departemen.

2. Langkah kedua pada pembuatan admin dalam web ini penulis membuat *REST API* yang berfungsi sebagai penghubung atau jembatan antara aplikasi android dan *website*, sehingga *website*, aplikasi android, basisdata saling terhubung atau saling terintegrasi. Pembuatan *REST API* ini masih menggunakan Codeigniter sebagai frameworknya.

Berikut listing program untuk REST API :

**DataBangunan.php**

|  |
| --- |
| <?php  defined('BASEPATH') OR exit('No direct script access allowed');  class DataBangunan extends CI\_Controller {  public function \_\_construct()  {  parent::\_\_construct();  $this->output->enable\_profiler(TRUE);  }  public function index()  {  $method = $\_SERVER['REQUEST\_METHOD'];  if($method != 'GET'){  json\_output(400,array('status' => 400,'message' => 'Bad request.'));  } else {  $check\_auth\_client = $this->MyModel->check\_auth\_client();  if($check\_auth\_client == true){  $response = $this->MyModel->auth();  if($response['status'] == 200){  $resp = $this->MyModel->semua\_data\_bangunan();  json\_output($response['status'],$resp);  } |

Berdasarkan kode diatas yaitu *function* databangunan digunakan sebagai pemanggilan data dari basis data kedalam android. Pada baris kode if($response['status'] == 200){ $resp = $this->MyModel->semua\_data\_bangunan(); json\_output($response['status'],$resp);

digunakan sebagai pemanggilan data dengan method *GET* dari basis data. Sebelum mengambil data pada basis data website akan memeriksa terlebih dahulu kedalam MyModel->check\_auth\_client(); dan apabila pemeriksaan tersebut benar maka android dapat mengambil data dari basis data berbentuk json.

|  |
| --- |
| public function masukandata(){  $method=$\_SERVER['REQUEST\_METHOD'];  if($method != 'POST'){  json\_output(400,array('status' => 400,'message' => 'Bad request.'));  } else {  $check\_auth\_client = $this->MyModel->check\_auth\_client();  if($check\_auth\_client == true){  $response = $this->MyModel->auth();  $respStatus = $response['status'];  if($response['status'] == 200){  $params=json\_decode(file\_get\_contents('php://input'), TRUE);  print\_r($params);  $resp=$this->MyModel->input\_data\_umum\_bangunan\_negara($params);  json\_output($respStatus,$resp);  } |

Berdasarkan kode diatas *function* masukan digunakan untuk memasukan data pada android. Pada baris ini if($response['status'] == 200) {$params=json\_decode(file\_get\_contents('php://input'), TRUE); berfungsi sebagai kode input data berupa json yang *decode.* data di *decode* dikarenakan pada saat input atau memasukan data pada android data sudah di *encode* terlebih dahulu.

|  |
| --- |
| public function hapusdata($id\_bangunan)  {  $method = $\_SERVER['REQUEST\_METHOD'];  if($method != 'DELETE' || $this->uri->segment(3) == '' || is\_numeric($this->uri->segment(3)) == FALSE){  json\_output(400,array('status' => 400,'message' => 'Bad request.'));  } else {  $check\_auth\_client = $this->MyModel->check\_auth\_client();  if($check\_auth\_client == true){  $response = $this->MyModel->auth();  if($response['status'] == 200){  $resp=$this->MyModel->hapus\_data\_umum\_bangunan\_negara($id\_bangunan);  json\_output($response['status'],$resp);  }  } |

Berdasarkan kode diatas digunakan untuk menghapus data pada basis data dengan menggunakan android. Sebelum menghapus data pada basis data *website* akan memeriksa terlebih dahulu kedalam MyModel->check\_auth\_client(); dan apabila pemeriksaan tersebut benar maka android dapat dapat menghapus data.

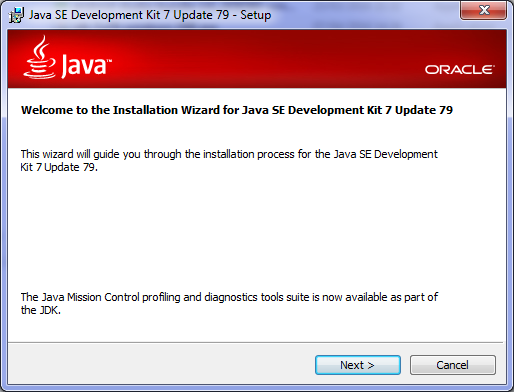
|  |
| --- |
| public function updatedata($id\_bangunan)  {  $method = $\_SERVER['REQUEST\_METHOD'];  if($method != 'PUT' || $this->uri->segment(3) == '' || is\_numeric($this->uri->segment(3)) == FALSE){  json\_output(400,array('status' => 400,'message' => 'Bad request.'));  } else {  $check\_auth\_client = $this->MyModel->check\_auth\_client();  if($check\_auth\_client == true){  $response = $this->MyModel->auth();  $respStatus = $response['status'];  if($response['status'] == 200){  $params= json\_decode(file\_get\_contents('php://input'), TRUE);  $params['date\_input'] = date('Y-m-d');  if($params['fungsi\_bangunan']==""||$params['jenis\_bangunan'] == "") {  $respStatus = 400;  $resp = array('status' => 400,'message' => `Data tidak boleh kosong`  } else {  $resp=$this->MyModel->update\_data\_umum\_bangunan\_negara($id\_bangunan,$params);  }  json\_output($respStatus,$resp); |

Pada kode diatas digunakan untuk memperbarui data bangunan yang telah dimasukan pada basis data. Pada kode diatas menggunakan method *PUT* untuk mengambil data yang ada pada basis data dan memasukan kembali data yang telah diperbaharui.

**4.3.2 Pembuatan Aplikasi Android**

1. Pada langkah pertama penulis melakukan instalasi *software* JDK (*Java Development Kit).* JDK yang digunakan yaitu versi jdk1.8.0\_77. Berikut adalah langkah-langkah instalasi *software* JDK:

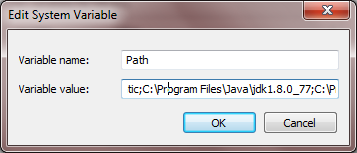
1. Penulis mengunduh *software* JDK di situs resmi oracle [8]
2. Pada langkah-langkah instalasi JDK, penulis hanya menekan tombol *next,* seperti gambar 3.9.
3. Jika proses instalasi telah selesai dan berhasil, maka akan terdapat folder JDK dan JRE pada C:\Program Files\Java.



(a) (b)

Gambar 4.7 (a) Instalasi Awal JDK, (d) Proses Instalasi JDK Selesai

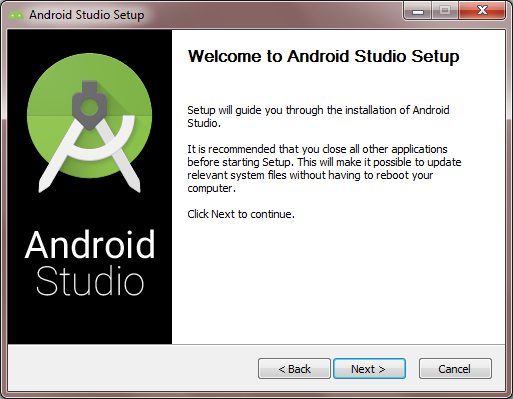
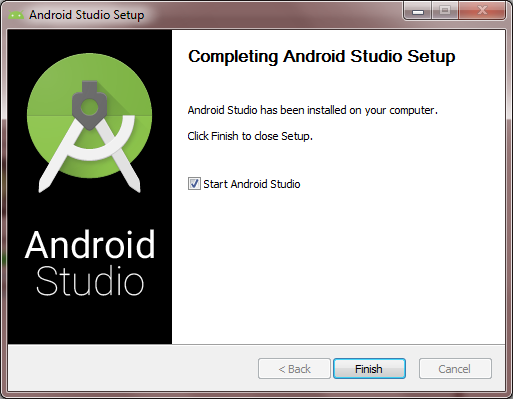
1. Selanjutnya penulis melakukan pengaturan *PATH* Java*,* dengan cara memilih *Control Panel* kemudian penulis memilih *System,* penulis melanjutkan dengan memilih *Advanced* danmengklik *Environment Variables,* kemudian penulis mengisi *variabel*  *name : PATH* dan *variable value* : C:\Program Files\Java\ jdk1.8.0\_77\bin; (menunjuk pada folder \bin JDK yang di*install*). Gambar 3.9 adalah proses yang dilakukan penulis dalam pengaturan *PATH* Java.



Gambar 4.8 Tampilan *System* *Variable*

**2.** Setelah instalasi JDK selesai dilakukan pada langkah pertama, selanjutnya penulis menginstalasi Android Studio. Berikut adalah langkah-langkah instalasi Android Studio:

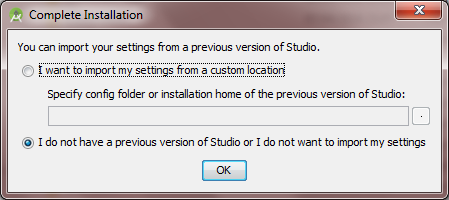
1. Penulis mengunduh *software* Android Studio di situs resmi Android Studio[2]
2. Setelah selesai mengunduh, penulis mencari *file* Android Studio instalasi *executable* (bernama **android-studio-bundle-<version>.exe**) di jendela *Windows Explorer* dan penulis mengklik dua kali untuk memulai proses instalasi, setelah itu langkah-langkah instalasi Android Studio berikutnya penulis hanya menekan tombol *next* seperti pada gambar 3.6.



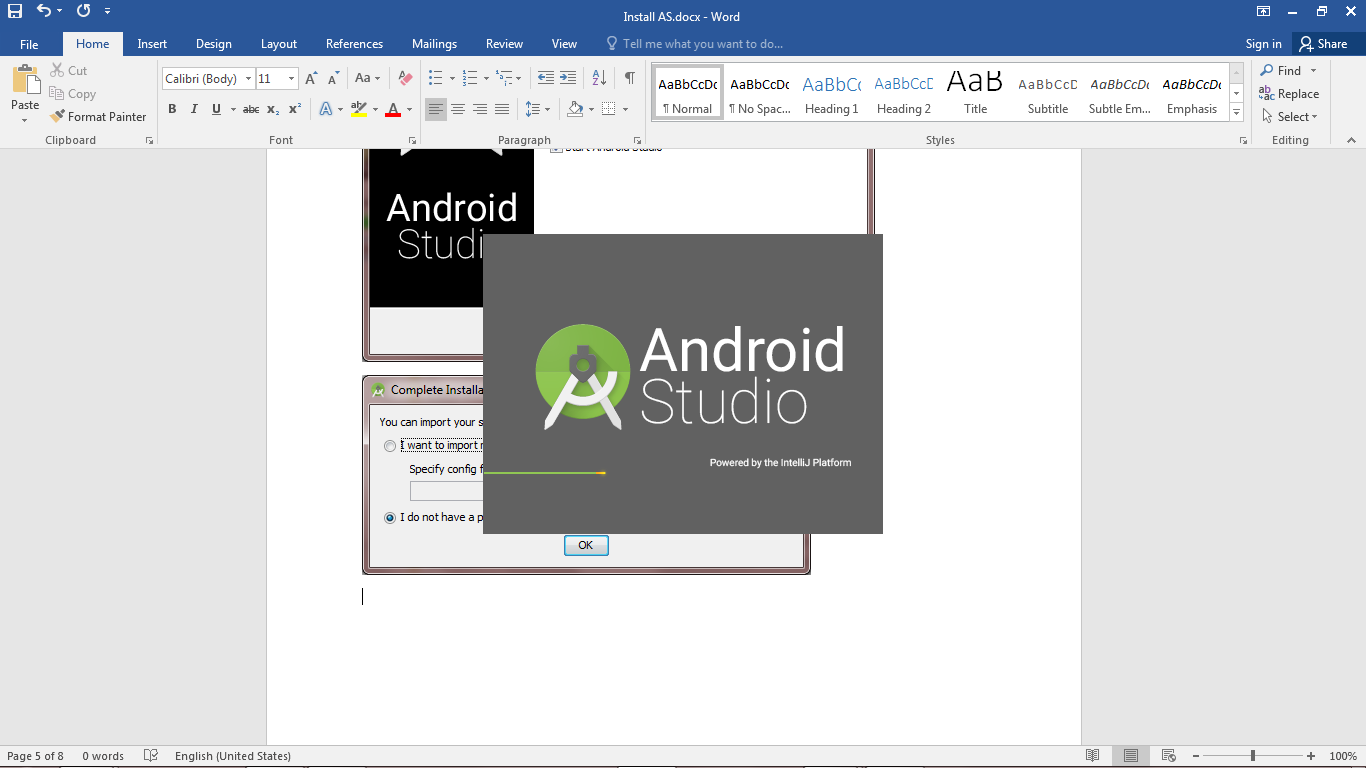
1. (b)

Gambar 4.8 (a) Proses Instalisasi Awal Android Studio, (b) Instalasi Android Studio selesai

1. Setelah langkah instalasi selesai seperti pada gambar 3.11 (b), kemudian akan muncul jendela yang menyediakan opsi untuk mengimpor pengaturan dari versi Android Studio sebelumnya seperti pada gambar 3.12. Jika memiliki pengaturan dari versi sebelumnya dan ingin mengimpor mereka ke dalam instalasi terbaru, maka opsi yang sesuai dengan lokasi dipilih. Karena penulis baru pertama kali meng*install* Android Studio maka pernyataan “*I do not have a previous version of Android Studio or I do not want to import my settings”* dipilihdan penulis menekan tombol *OK* untuk melanjutkan. Setelah penulis memilih pernyataan tersebut, kemudian jendela untuk memulai Android Studio akan tampil seperti pada gambar 3.13.

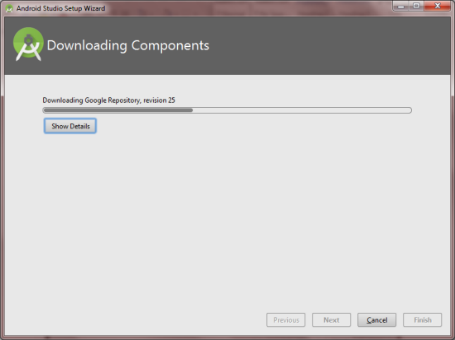


Gambar 4.9 Jendela Kelengkapan Instalasi.



Gambar 4.10 Jendela Memulai Android Studio

1. Setelah Android Studio berhasil dimuat, selanjutnya akan tampil jendela *Android Studio* *Setup Wizard* seperti pada gambar 3.14. Pada jendela ini penulis mengunduh dan menginstalasi komponen Android SDK Tools. Sebelumnya penulis memastikan komputer terhubung dengan internet



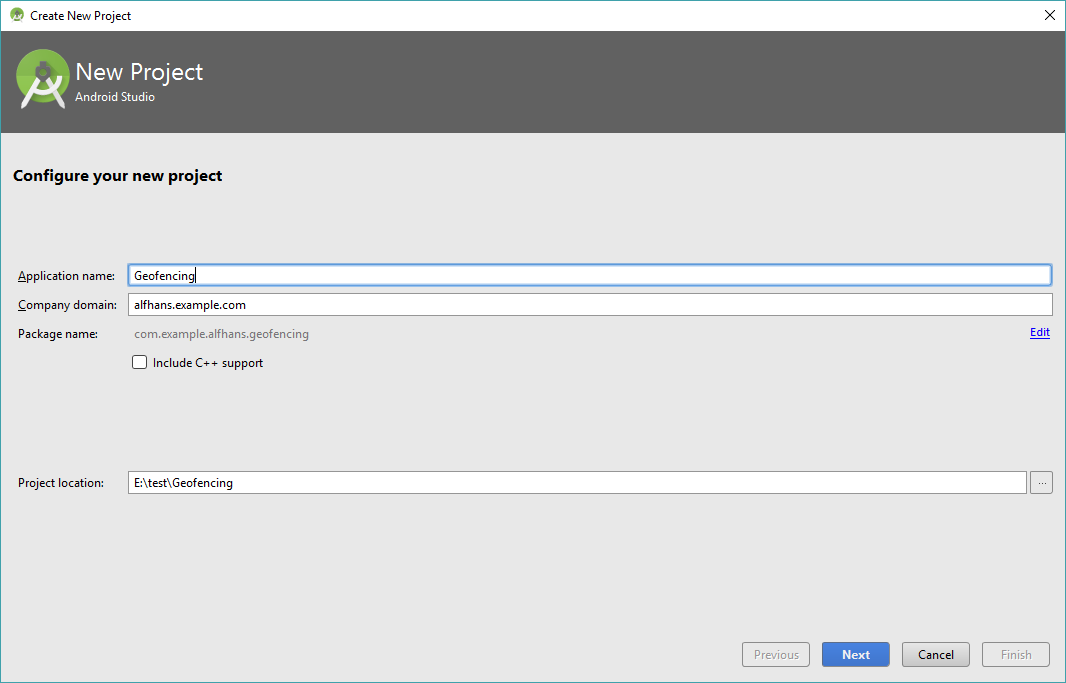
Gambar 4.11 Proses Unduhdan Instalasi Android SDK tools

3. Setelah melakukan instalasi Android Studio pada langkah ke tiga, selanjutnya penulis melakukan instalasi ADB Drivers pada komputer/PC, agar nantinya aplikasi dapat di uji coba dengan menggunakan *smartphone*. Berikut adalah langkah-langkah dalam melakukan instalasi ADB Drivers:

1. Penulis menghubungkan *Smartphone* Android dengan komputer menggunakan kabel USB.
2. Pada komputer yang terdapat di *desktop* atau *Windows Explorer*, penulis menekan tombol klik pada bagian kanan *mouse* kemudian penulis memilih *Manage****.***
3. Selanjutnya penulis memilih *Devices*yang terdapat di panel kiri.
4. Penulis mencari dan memperluas perangkat lain di panel kanan.
5. *Update Driver Software* dipilih setelah penulis menekan tombol klik dibagian kanan *mouse* pada nama perangkat. Langkah ini akan meluncurkan *Hardware Update Wizard.*
6. Penulis memilih *Browse my computer for driver software*kemudian menekan tombol *next*.
7. Kemudian penulis mencari folder driver USB. (The Google USB Driver terletak di <sdk>\extras\google\usb\_driver\.) dengan menekan *Browse*.
8. Penulis menekan tombol *Next* untuk menginstalasi *driver*.

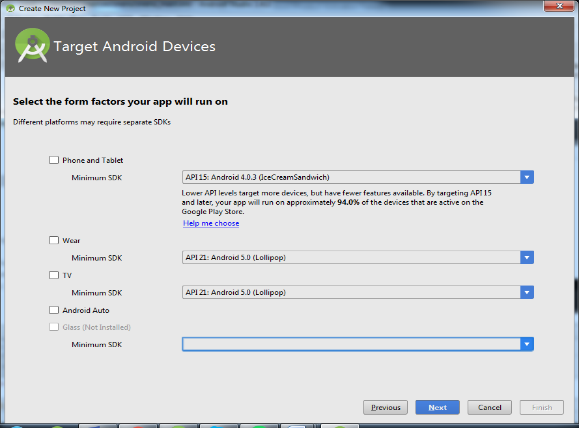
4**.** Setelah selesai menginstall android studiomaka tahap selanjutnya adalah pembuatan aplikasi dengan memilih *Start an new Android Studio Project.*

1. Langkah pertama adalah penulis mengisi nama projek “*Survei Bangun Gedung*”, *Company Domain* (alfhan.example.com)*,* kemudian penulis menentukan direktori kerja atau *workspace* yang digunakan untuk menyimpan projek Android.Lokasi direktori kerja yang ditetapkan penulis yaitu “D:\Survei Bangun Gedung”. Lalu penulis menekan tombol *Next* seperti pada gambar 4.7.



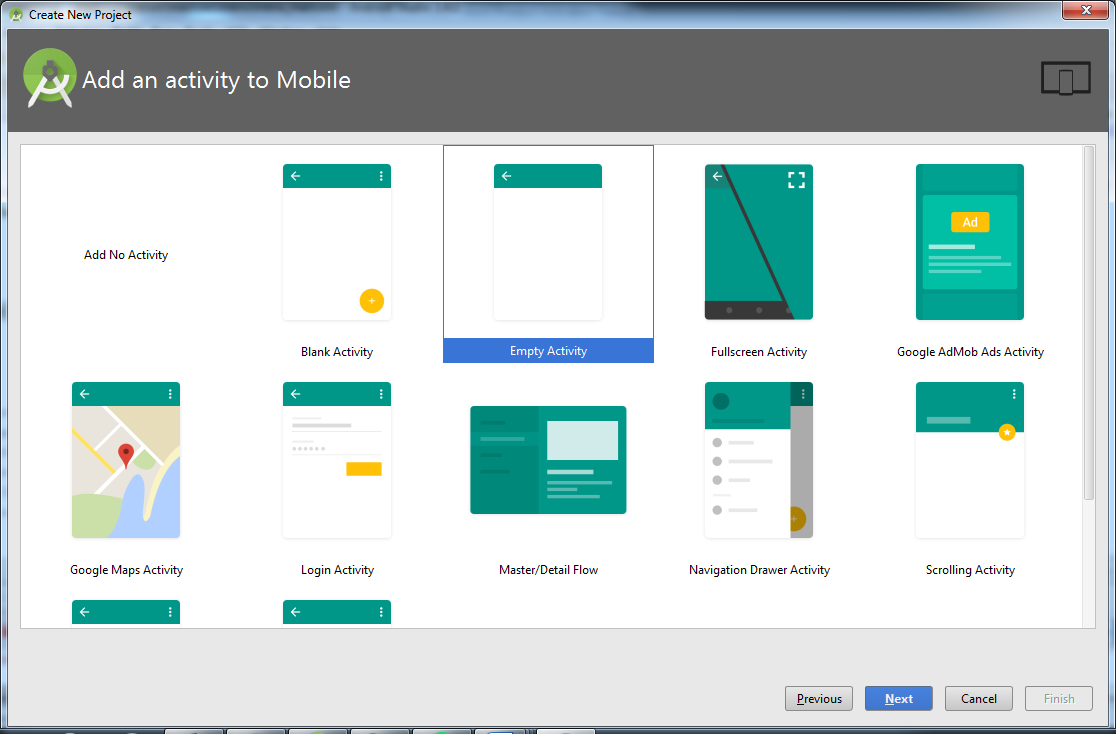
Gambar 4.12 Konfigurasi Projek Baru

1. Setelah membuat projek pada gambar 4.12, penulis melanjutkan dengan memillih target *run* aplikasi yang ada pada gambar 4.13, dalam hal ini penulis memilih *Phone and Tablet.* Dibagian minimum SDK, penulis memilih API 15: Android 4.0.3 (IceCreamSandwich). Kemudian penulis menekan tombol *Next*.



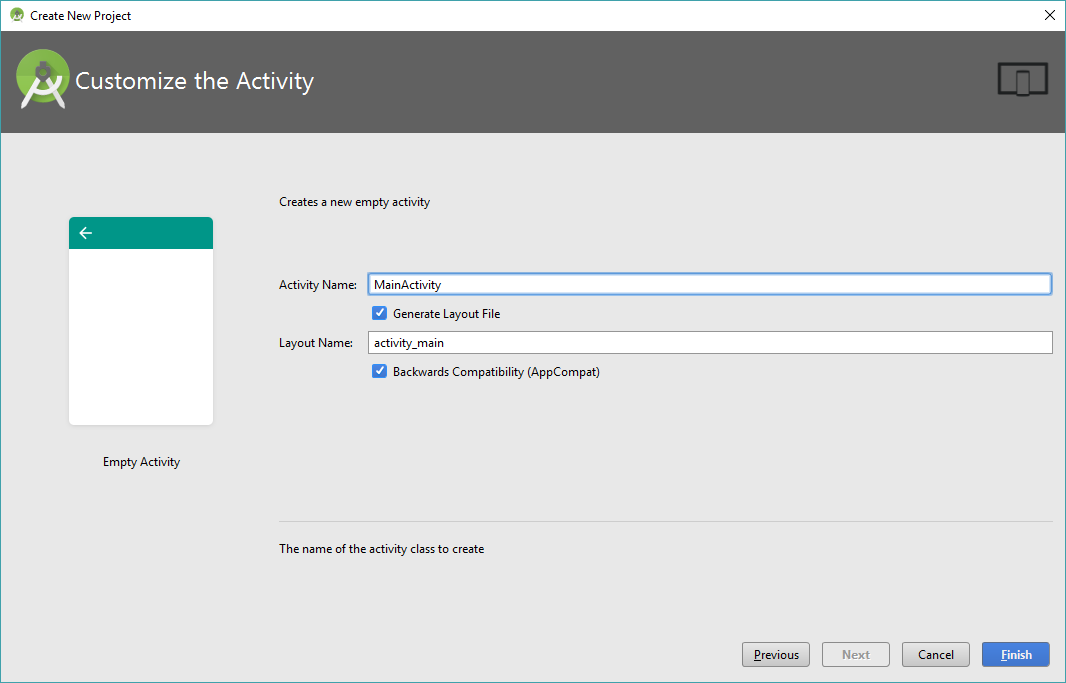
Gambar 4.13 Menentukan Target Run Aplikasi

1. Dari gambar 4.13 kemudian akan muncul tampilan untuk pilihan *Layout* (*Activity*) aplikasi. Disini pengguna Android Studio bebas memilih *activity*  yang disediakan tanpa harus membuat *layout* nya lagi dari awal. Untuk menyesuaikan dengan desain yang telah dibuat, maka penulis memilih *Empty Activity* seperti pada gambar 4.14*.* Setelah itu penulis menekan tombol *Next*.



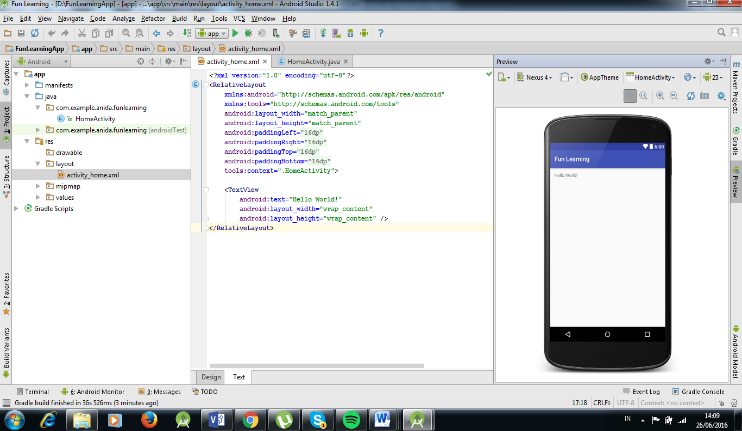
Gambar 4.14 Memilih Tampilan *Layout* *Activity*

1. Pada tahap ini, penulis menentukan nama dari *activity* yang telah dipilih pada langkah lima.c. Disini penulis memberi nama *MainActivity* kemudian penulis menekan tombol *Finish* seperti pada gambar 4.15.



Gambar 4.15 Mengatur Nama *Activity*

1. Setelah penulis menekan tombol *finish*, maka akan tampil lembar projek Android Studio seperti pada gambar 4.16.



Gambar 4.16 Tampilan Lembar Kerja

**4.3.2.1 Penambahan Library Pada Gradle**

*Gradle* adalah suatu fitur build automation yang open source. *Gradle*didesain untuk membuat multi-project yang sedang berkembang menjadi project berskala besar. File Gradle terdiri dari file settings.gradle dan build.gradle. build.gradle berisi pengaturan global mengenai aplikasi pada Android Studio. Pada aplikasi survei bangun gedung ini dibutuhkan *library* tambahan pada *gradle* android seperti berikut :

|  |
| --- |
| android {  compileSdkVersion 27  defaultConfig {  applicationId "com.example.alfhanrf.skripsihehe"  minSdkVersion 20  targetSdkVersion 27  versionCode 1  versionName "1.0"  testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  buildConfigField("String", "URI", "\"http://192.168.43.49/Surveygedung/\"")  }  buildTypes {  release {  minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro' |

Kode diatas adalah *gradle* yang digunakan dalam aplikasi, aplikasi survei bangun gedung menggunakan *compilesdkversion 27* dan *BuildConfigField* digunakan sebagai *base url* atau *link website*  yang telah dibuat sebagai *platform* basis data aplikasi.

|  |
| --- |
| dependencies {  implementation fileTree(dir: 'libs', include: ['\*.jar'])  implementation 'com.android.support:appcompat-v7:27.1.1'  implementation 'com.android.support:design:27.1.1'  implementation 'com.android.support:cardview-v7:27.1.1'  implementation 'com.android.support.constraint:constraint-layout:1.1.2'  implementation 'com.google.android.gms:play-services-maps:15.0.1'  implementation 'com.android.support:support-v4:27.1.1'  testImplementation 'junit:junit:4.12'  implementation 'com.jakewharton:butterknife:8.8.1'  annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1'  androidTestImplementation 'com.android.support.test:runner:1.0.2'  androidTestImplementation'com.android.support.test.espresso:espresso-core:3.0.2'  implementation 'com.squareup.retrofit2:retrofit:2.4.0'  implementation 'com.squareup.retrofit2:converter-gson:2.4.0'  implementation 'com.google.code.gson:gson:2.8.2'  implementation 'com.squareup.okhttp3:okhttp:3.10.0'  implementation 'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'  implementation 'com.squareup.okhttp3:logging-interceptor:3.10.0'  } |

*Dependencies* adalah *plugin* atau tambahan *library* yang digunakan untuk pembuatan aplikasi survei bangun gedung. *Library* yang sangat dibutuhkan pada aplikasi ini adalah 'com.squareup.retrofit2:retrofit:2.4.0' sebagai pembuat penghubung antara REST API *website* dan android. *Library* annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1' juga digunakan sebagai *butterknife* pada setiap activity android untuk menggantikan model deklarasi variabel.

**4.3.2.2 Pembuatan Class APIInterface**

Setelah membuat project baru android selanjutanya membuat kelas baru bernama APIInterface yang berfungsi sebagai REST API (*Application Programming Interface)* penghubung antara android dan *website.* Pada kelas APIInterface ini android memerlukan tambahan *library* baru bernama Retrofit. **Retrofit** adalah library Android yang dibuat oleh Squre yang digunakan sebagai REST Client pada Android, yang berfungsi untuk memudahkan dalam programing. Karena tidak perlu lagi untuk membuat method-method sendiri untuk menggunakan REST Client API dari backend. Berikut kode aplikasi yang terdapat pada kelas APIInterface.Java :

|  |
| --- |
| public interface APIInterface {  @POST("auth/login")  Call<JsonObject> login(@Query("user\_id")String user\_id, @Query("password") String password); |

Kode diatas adalah API untuk *login* pada android dengan menggunakan method *POST* dan diberinama login untuk mendapatkan user\_id dan mendapatkan password sebagai string.

|  |
| --- |
| @POST("auth/logout")  Call<RequestResponse> logout(  @Header("Authorization") String token,  @Header("User-ID") String userid); |

Kode diatas digunakan untuk *logout* pada aplikasi android. Dengan menggunakan method *POST* dan mendapatkan *Authorization* dan user\_id sebagai string. User-ID dan *authorization* merupakan sebuah header yang dibutuhkan aplikasi android untuk masuk kedalam aplikasi dan menggunakan basis data pada android.

|  |
| --- |
| @POST("auth/authenticate")  Call<RequestResponse> checkLogin(  @Header("Authorization") String token,  @Header("User-ID") String userid); |

Kode diatas digunakan untuk cek validasi login. API tersebut digunakan untuk mengambil kembali userid dan token yang telah ada didalam basis data dan mencocokan apabila *session login* pada database telah berakahir.

|  |
| --- |
| @GET("databangunan/list\_bangunan\_user")  Call<BangunanList> getSemuaBangunan(  @Header("Authorization") String token,  @Header("User-ID") String userid); |

Kode diatas digunakan mengambil seluruh data bangunan yang ada didalam database survei bangun gedung. API tersebut beranama getSemuaBangunan yang digunakan pada saat user membutuhkan seluruh data gedung di dalam tampilan *Mainactivity.* API tersebut memanggil kelas BangunList sebagai kamus data yang akan mendapatkan data dari basis data dan akan disimpan dalam BangunList tersebut.

|  |
| --- |
| @DELETE("databangunan/hapusdata/{id\_bangunan}")  Call<RequestResponse> deleteBangunan (  @Header("Authorization") String token,  @Header("User-ID") String userid,  @Path("id\_bangunan") String id\_bangunan); |

Pada kode diatas adalah API untuk menghapus data gedung yang telah dimasukan dengan android ke dalam basis data. API tersebut menggunakan method *DELETE* untuk menghapus pada android. Parameter yang ditambahkan pada API adalah id\_bangunan untuk mengarahkan id\_bangunan sebagai *primary key* untuk menghapus data.

|  |
| --- |
| @GET("databangunan/list\_bangunan\_petugas survei/{user\_id}")  Call<BangunanList> getBangunanPetugas survei(  @Header("Authorization") String token,  @Header("User-ID") String userid,  @Path("user\_id") String user\_id); |

Pada kode diatas digunakan untuk mengambil data yang hanya dimasukan oleh *user* tertentu. Dengan tambahan parameter user\_id sehingga data yang ditampilkan hanya data yang dimasukan oleh *user* yang bersangkutan.

|  |
| --- |
| @POST("databangunan/masukandata")  Call<RequestResponse> masukandata (  @Header("Authorization") String token,  @Header("User-ID") String userid,  @Body Bangunan bangunan); |

Berdasarkan kode diatas digunakan untuk memasukan data pada basis data dengan meggunakan method *POST* dengan android. Dalam memasukan data tersebut digunakan @Body Bangunan sebagai parameter body yang akan di masukan pada basis data.

|  |
| --- |
| @PUT("databangunan/updatedata/{id\_bangunan}")  Call<RequestResponse> updateBangunan (  @Header("Authorization") String token,  @Header("User-ID") String userid,  @Path("id\_bangunan") String id\_bangunan,  @Body Bangunan bangunan); |

Pada kode diatas digunakan untuk mengubah atau memperbarui data yang telah dimasukan kedalam basis data. Parameter yang digunakan adalah id\_bangunan yang digunakan sebagai *primary\_key* pada saat pengambilan data agar dapat mengubah data tersebut.

**4.3.2.3 Pembuatan Halaman SplashScreen**

Setelah membuat project baru android selanjutnya penulis membuat activity XML dan java untuk mendesain tampilan awal pada saat program aplikasi dijalankan yaitu Splashscreen.Berikut tahapan-tahapan pembuatan splashscreen :

1. Desain tampilan activity\_splashscreen.xml dengan menggunakan ContraintLayout agar widget terurut secara *vertical.* Didalam tag ConstraintLayout menggunakan android orientation *vertical* karena perangkat yang nantinya akan digunakan untuk aplikasi berorientasi *vertical.* Untuk mengatur tinggi dan lebar layer dengan menggunakan match\_parent supaya ukuran mengikuti ukuran layer. Fungsi yang digunakan pada xml yaitu android:gravity center agar tampilan berada ditangah layer. Penulis menggunakan *ImageView, Progresbar* dan *TextView* untuk mengisi konten dalam layar. Berikut kode aplikasi pada activity\_splasscreen.xml :

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent" |

1. Ditambahkan ImageView didalam tag ConstraintLayout. ImageView tersebut berfungsi untuk menampilkan gambar. Gambar yang digunakan untuk splashscreen adalah sbg.png kemudia untuk lebar dan tinggi dengan mengatur satuan dp. Berikut kode aplikasi pada activity\_splashscreen.xml :

|  |
| --- |
| <ImageView  android:id="@+id/imageView3"  android:layout\_width="200dp"  android:layout\_height="200dp"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginStart="8dp"  android:layout\_marginTop="32dp"  android:src="@drawable/ic\_menu\_camera"  app:layout\_constraintBottom\_toTopOf="@+id/textView6"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" /> |

1. Ditambahkan 2 TextView untuk menuliskan nama aplikasi dan untuk menuliskan nama pembuat aplikasi. Berikut kode aplikasi yang masih dalam activity\_splashscreen.xml :

|  |
| --- |
| <TextView  android:id="@+id/textView6"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="36dp"  android:layout\_marginEnd="8dp"  android:layout\_marginStart="8dp"  android:text="SISTEM INFORMASI SURVEI BANGUN GEDUNG"  android:textAlignment="center"  android:textColor="@android:color/black"  android:textSize="20sp"  android:textStyle="bold"  app:layout\_constraintBottom\_toTopOf="@+id/progressBar"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" />  <TextView  android:id="@+id/textView4"  android:layout\_width="wrap\_content"  android:layout\_height="0dp"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginStart="8dp"  android:textStyle="bold"  android:text=""  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" /> |

1. Ditambahkan 1 proggresbar digunakan untuk *loading screen* dan pengechekan *session use*r yang login di aplikasi survei bangun gedung. Berikut kode progress bar di activity\_splashscreen.xml :

|  |
| --- |
| <ProgressBar  android:id="@+id/progressBar"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginStart="8dp"  app:layout\_constraintBottom\_toTopOf="@+id/textView4"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" /> |

1. Setelah desain tampilan Splashscreen pada activity\_splashscreen.xml, kemudian memasukan kode pada activity Java yang bernama splashscreen.java yang sudah dibuat pada langkah diatas. Source code splashscreen.java mempunyai kelas splashcreen. Kemudia tampilan pembuka ini akan memanggil layout dan activity\_splashscreen.xml. lalu activity\_splashscreen akan muncul selama aplikasi mencoba koneksi ke dalam server dan koneksi ke server aplikasi. Setelah itu maka tampilan akan berpindah ke tampilan *login* dengan perintah *intent* dari kelas Splashscreen.this ke kelas Login.class. Intent merupakan sebuah *object* yang merupakan kunci berkomunikasi dengan *activity* lainnya. Di dalam *Splashscreen.java* juga terdapat method untuk pengechekan *userSession. UserSession* ini digunakan untuk mengambil data *login* yaitu IdUser dan UserToken untuk autentikasi mengakses basis data pada *website.* Apabila *user* belum *login* kedalam aplikasi maka aplikasi akan tertuju ke halaman *Login* apabila *user* sudah *login* maka aplikasi akan tertuju langsung kedalam *MainActivity.* Berikut kode skrip pada Splashscreen.Java :

|  |
| --- |
| public class Splashscreen extends BaseActivity {  UserSession userSession;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_splashscreen);  userSession = new UserSession(Splashscreen.this);  Thread timerThread = new Thread(){  public void run(){  try {  sleep(3000);  } |

Kode diatas digunakan deklarasi class UserSession dan insisialisai class UserSession kedalam Splashscreen.Java. dan loading akan berhenti dan memberikan pesan gagal pada saat sudah cek koneksi setelah 3 detik.

|  |
| --- |
| catch (InterruptedException e){  e.printStackTrace(); }finally {  if (userSession.isUserLoggedIn()){  APIInterface apiInterface = APIClient.getClientAuthorize().checkLogin(APIInterface.class);  Call<RequestResponse> call = apiInterface.checkLogin(userSession.getUserToken(), String.valueOf(userSession.getIdUser()));  call.enqueue(new Callback<RequestResponse>() { |

Kode diatas digunakan untuk memanggil API retrofit pada baris APIClient.getClientAuthorize().checkLogin(APIInterface.class); API tersebut digunakan untuk mengechek apakah *user* tersebut sudah *login* atau belum dan untuk pengecekan UserSession apakah *login user* sudah dalam waktu 12 jam terakhir atau tidak.

|  |
| --- |
| try {  if(response.isSuccessful())  Intent intent = new Intent(Splashscreen.this,MainActivity.class);  startActivity(intent);  }else{  sessionExpired();  Intent intent = new Intent(Splashscreen.this,Login.class);  startActivity(intent);  }  }catch (Exception e){  Toast.makeText(Splashscreen.this, "Password atau ID Salah",Toast.LENGTH\_SHORT).show();  }  }  @Override  public void onFailure(Call<RequestResponse> call, Throwable t) { showToast(t.toString());  }  });  }else{  Intent intent = new Intent(Splashscreen.this,Login.class);  startActivity(intent);  }  }  }  };  timerThread.start();  } |

Pada kode diatas digunakan untuk pengecekan dan untuk menentukan apakah *session user* sudah berakhir atau belum. Dan untuk pengecekan apakah id dan *password* yang dimasukan pada saat *form login* pada android benar atau tidak.

**4.3.2.4 Pembuatan Halaman Login**

Pembuatan halaman *login* digunakan untuk membatasi akses siapa saja yang dapat menggunakan aplikasi survei bangun gedung. Halaman ini akan ditampilkan setelah halaman *splashscreen* apabila sudah melewati kondisi pengecekan pada halaman tersebut. Pada pembuatan halaman *login* ini terdapat activity\_login.xml dan login.java. berikut kode pembuatan halaman *login*.

1. Seperti *splahscreen* halaman *login* juga memiliki file xml dan halaman java. Langkah pertama yang dilakukan adalah membuat tampilan untuk halaman login. Berikut kode login.xml

|  |
| --- |
| <ImageView  android:id="@+id/imageView3"  android:layout\_width="200dp"  android:layout\_height="200dp"  android:layout\_marginBottom="8dp"  android:layout\_marginEnd="8dp"  android:layout\_marginStart="8dp"  android:layout\_marginTop="32dp"  android:src="@drawable/ic\_menu\_camera"  app:layout\_constraintBottom\_toTopOf="@+id/guideline4"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />  <android.support.constraint.Guideline  android:id="@+id/guideline4"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:orientation="horizontal"  app:layout\_constraintGuide\_percent="0.5" />  <android.support.design.widget.TextInputEditText  android:id="@+id/txtPassword"  android:text="alfian123"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Password"  android:inputType="textPassword" />  </android.support.design.widget.TextInputLayout> |

Pada halaman login terdapat 1 *ImageView,* 2 *EditText*, 1 *button* dan 2 *TextView. Imageview* digunakan untuk menampilkan gambar dari logo aplikasi survei bangun gedung. Dan 2 *EditText* digunakan untuk mengisi user\_id dan *password. Button* digunakan untuk masuk kedalam aplikasi survei bangun gedung. *TextView* digunakan untuk menampilkan keterangan aplikasi survei bangun gedung dan menampilkan nama pembuat aplikasi.

1. Setelah tampilan layout *login* telah dibuat maka masukan kode pada *login.java* untuk menjalankan fungsi pada saat *user* sedang *login.* Berikut potongan kode dan penjesalan pada *login.java :*

|  |
| --- |
| public class Login extends BaseActivity implements View.OnClickListener {  @BindView(R.id.btnLogin) Button btnLogin;  @BindView(R.id.txtID) TextInputEditText txtID;  @BindView(R.id.txtPassword) TextInputEditText txtPassword; |

Berdasarkan kode diatas adalah awal dari kelas *login.java* kelas extends *BaseActivity* dan implement *ViewOnclickListener.* Dengan menggunakan ekstend pada header class maka *method* ata *function* yang terdapat pada *baseactivity* tidak perlu di inisialisasi pada kelas *login.* Untuk implement *View.OnclickListner* berfungsi untuk membuat *method* aksi kepada button yang akan diklik pada saat *login*. @BindView diatas adalah untuk deklarasi atribut atau widget yang terdapat pada activity\_login.xml.

|  |
| --- |
| private void login(){  mRegProgres.setTitle("Getting Data");  mRegProgres.setMessage("Please Wait...");  mRegProgres.setCanceledOnTouchOutside(false);  mRegProgres.show();  final String user\_id = txtID.getText().toString().trim();  String pass = txtPassword.getText().toString().trim();  Map<String, Object> jsonParams = new ArrayMap<>();  jsonParams.put("username", user\_id);  jsonParams.put("password", pass); |

Kode diatas adalah *method login* yang dibuat pada *login.java.* mRegProgres adalah progress bar yang terdapat pada *baseactivity. Progress bar* tersebut memberikan pesan “Getting Data” setelah itu android akan mengambil *string* pada masukan user\_id dengan *id* pada activity\_login.xml yaitu txtID dan mengambil *string password* dengan *id* txtPassword pada activity\_login.xml.

|  |
| --- |
| APIInterface.apiInterface= APIClient.getClientAuthorize().create(APIInterface.class);  Call<JsonObject> call = apiInterface.login(user\_id, pass);  call.enqueue(new Callback<JsonObject>() {  @Override  public void onResponse(Call<JsonObject> call, Response<JsonObject> response) {  try {  if(response.isSuccessful()){  Log.d("masuk", "asasas");  JsonElement jsonToken = response.body().get("token");  JsonElement jsonId = response.body().get("id");  JsonElement jsonNama = response.body().get("nama");  JsonElement jsonUserId = response.body().get("user\_id");  JsonElement jsonuserRole = response.body().get("role"); |

Setelah mendapatkan user\_id dan *password* berupa json, maka android akan memanggil API login yang telah dibuat pada kelas APIClient. Pada saat *login* akan mendapatkan *response* dari *website* berupa json UserId, UserToken, UserRole dan UserNama.

**4.3.2.5 Pembuatan Halaman MainActiviy**

Pada pembuataan *mainactivity* sama seperti *splashscreen* namun sudah tidak memerlukan untuk pembuatan file baru activity\_main.xml nya dan java class karena sudah secara default tersedia ketika membuat project baru. Sebelumnya tambahkan beberapa *user-permision* agar aplikasi Survei Bangun Gedung dapat berjalan. Berikut tambahan ­*user-permision* yang ditambahkan di dalam AndroidManifest.xml :

|  |
| --- |
| <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" /> <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" /> |

android.permission.ACCESS\_FINE\_LOCATION digunakan untuk memberikan perizinan tentang melakukan deteksi lokasi smartphone secara terperinci. android:name=android.permission.ACCESS\_NETWORK\_STATE digunakan untuk mengecheck apakah smartphone terdapat koneksi internet. android.permission.ACCESS\_COARSE\_LOCATION digunakan untuk melakukan akses lokasi Anda melalui WIFI yang sedan digunakan. android.permission.INTERNET digunakan untuk mengakses internet ketika melakukan download API Google Maps.

1. Langkah pertama dalam pembuatan MainActivity adalah mengatur tampilan awal, agar tampilan memenuhi ukuran layar ponsel, pada tampilan ini dibuat *width* dan *height* dengan *match\_parent,* dan berikan *id* sebagai identitas tampilan nanti jika tampilan ini dipanggil kembali oleh sebuah fungsi. id tampilan main\_activity adalah “drawer” dapat dilihat dari android:id=”+id/drawer\_layout”. alignParentTop berfungsi agar tampilan tepat berada dibawah header aplikasi. Dan android.support.v4.widget.DrawerLayout ini *library* yang digunakan pada menu aplikasi apabila menggunakan drawer atau tampilan yang bergeser pada tampilan main\_activity. Dan menambahkan *drawer* dengan *id* android:id="@+id/nav\_view"yang berfungsi untuk meletakan menu yang disediakan aplikasi. Include layout="@layout/app\_bar\_main" digunakan untuk memasukan tampilan app\_bar\_main kedalam tampilan MainActivity

|  |
| --- |
| <android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/drawer\_layout"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:fitsSystemWindows="true"  <include  layout="@layout/app\_bar\_main"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  <android.support.design.widget.NavigationView  android:id="@+id/nav\_view"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:layout\_gravity="start"  android:fitsSystemWindows="true"  app:headerLayout="@layout/nav\_header\_main"  app:menu="@menu/activity\_main\_drawer" />  </android.support.v4.widget.DrawerLayout> |

1. Selanjutnya kode dimasukan ke dalam *Mainactivity.class* untuk menjalankan fungsi-fungsi yang terdapat pada program yang terdapata pada tampilan MainActivity. Berikut kode yang yang terdapata pada MainActiviy.Java

|  |
| --- |
| private void addMarkers(){  APIInterface.apiInterface= APIClient.getClientAuthorize().create(APIInterface.class);  Call<BangunanList>call= apiInterface.getSemuaBangunan(userSession.getUserToken(),userSession.getIdUser());call.enqueue(new Callback<BangunanList>() { |

Kode diatas adalah untuk memanggil API getSemuaBangunan pada *class* APIInterface untuk memanggil seluruh data yang ada didalam basis data. Dengan menggunan refrensi dari *user* UserToke dan IdUser jika kedua refrensi tersebut valid maka *marker* akan muncul.

|  |
| --- |
| for (int i = 0;i< bangunan.size();i++){  mLatlng = new LatLng(Double.parseDouble(bangunan.get(i).getLatitude()),  Double.parseDouble(bangunan.get(i).getLongitude()));  Marker marker = mMap.addMarker(new MarkerOptions()  .position(mLatlng)  .title(bangunan.get(i).getNama\_bangunan())  .snippet("Cek Data Gedung"));  detailMarker.put(marker,bangunan.get(i));}  showToast("Mendapatkan Posisi Gedung")} |

Kode diatas menggunakan pengulangan untuk membaca semua data yang terdapat pada basis data, setelah itu data yang diambil hanya *latitude* dan *longitude* untuk menentukan posisi *marker*. Setelah *latitude* dan *longitude* sudah didapatkan apabila *marker* ditekan maka akan muncul nama bangunan dan keterangan untuk cek data gedung.

|  |
| --- |
| @Override  public void onFailure(Call<BangunanList> call, Throwable t) {  showToast("Gagal Mendapatkan Lokasi")}  }  });  mMap.setOnInfoWindowClickListener(new GoogleMap.OnInfoWindowClickListener() {  @Override  public void onInfoWindowClick(Marker marker) {  Intent i = new Intent(MainActivity.this,DetailBangun.class);  i.putExtra("bangunan",detailMarker.get(marker));  startActivity(i);  }  });  } |

Kode diatas apabila *marker* ditekan untuk kedua kalinya ditempat yang sama maka akan berpindah ke tampilan DetailBangun.class yang berisi keterangan tentang bangunan gedung yang telah di survei oleh petugas survei.

|  |
| --- |
| private void checkRole(){  userSession = new UserSession(MainActivity.this);  //Petugas survei = 2 , User Eksternal = 3  String id = userSession.getUserRoleStr();  Log.d("role", id);  if(id.equals("3")){  hideItemNavigation();  } |

Berdasarkan kode diatas adalah kode untuk pemisahan akses *user.* Terdapat 2 *user* yaitu petugas survei dan user eksternal. Setelah *role user* diketahui maka kode diatas memanggil *method* hideItemNavigation();

|  |
| --- |
| private void hideItemNavigation(){  NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view);  Menu nav\_Menu = navigationView.getMenu();  nav\_Menu.findItem(R.id.nav\_history).setVisible(false);  nav\_Menu.findItem(R.id.nav\_input\_data).setVisible(false);  } |

Kode diatas adalah *method* dari hideItemNavigtation() yang berfungsi untuk menyembunyikan menu item nav\_history dan nav\_input\_data apabila yang *login* ke dalam aplikasi adalah  *role* user eksternal.

|  |
| --- |
| private void logout(){  userSession = new UserSession(MainActivity.this);  mRegProgres.setTitle("Logout Account");  mRegProgres.setMessage("Please Wait...");  mRegProgres.setCanceledOnTouchOutside(false);  mRegProgres.show();  APIInterface apiInterface = APIClient.getClientAuthorize().create(APIInterface.class);  Call<RequestResponse> call = apiInterface.logout(userSession.getUserToken(),userSession.getIdUser()); |

Berdsarkan kode diatas digunakan untuk menampilkan progress bar dan melakukan *logout* dari aplikasi survei bangun gedung. Kode diatas memanggil API logout dengan menggunakan refrensi UserToken dan IdUser.

|  |
| --- |
| public boolean onNavigationItemSelected(MenuItem item) {  // Handle navigation view item clicks here.  int id = item.getItemId();  if( id == R.id.posisi\_bangunan) }  else if (id == R.id.nav\_input\_data) {  Intent surveibangun = new Intent(MainActivity.this, Spinner\_fungsi.class); surveibangun.putExtra(ConstantsUtils.ACTION, ConstantsUtils.ACTION\_NEW);  MainActivity.this.startActivity(surveibangun);  } else if (id == R.id.nav\_list\_survei) {  Intent listSurvei = new Intent(MainActivity.this, BangunanView.class);  MainActivity.this.startActivity(listSurvei); |

Berdasarkan kode diatas digunakan sebagai menu pilihan di dalam aplikasi survei bangun gedung, terdapat 5 menu untuk petugas survei dan 3 menu untuk user eksternal. Diantaranya adalah menu posisi gedung, input gedung, daftar bangun gedung, history input petugas survei dan *logout.*

|  |
| --- |
| } else if (id == R.id.nav\_logout) {  AlertDialog.Builder alert = new AlertDialog.Builder(MainActivity.this);  alert.setTitle("Logout");  alert.setMessage("Anda Yakin Akan Logout ? ");  alert.setPositiveButton("Ya", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  dialog.dismiss();  logout();  }  }).setNegativeButton("Tidak", new DialogInterface.OnClickListener() {  @Override  public void onClick(DialogInterface dialog, int which) {  dialog.dismiss();  }  }).create().show()} |

kode diatas adalah salah satu menu yaitu *logout*, apabila menu *logout* ditekan maka akan menjalankan *method logout* yang telah dibuat sebelumnya. Setelah itu akan muncul Dialog untuk menyatakan *logout* atau tidak. Apabila menekan “Ya” maka akan kembali ke menu *Login.* Apabila menekan tidak maka akan kembali kedalam dalam MainActivity.

**4.3.2.6 Pembuatan Halaman DetailBangun**

Setelah pembuatan detailbangunyang berfungsi untuk menampilkan *maps* dan titik setiap bangun gedung yang telah didata oleh petugas survei, maka selanjutnya pembuatan detailbangunyang berfungsi untuk menampilkan data yang telah dimasukan oleh petugas survei. Pada pembuatan halaman detailbangunini memiliki java *class*  dan memiliki file xml untuk mengatur tampilan nya. Berikut listing kode untuk halaman detailbangun:

1. Langkah pertama dalam pembuatan detailbangunadalah mengatur tampilan awal, agar tampilan memenuhi ukuran layar ponsel, pada tampilan ini dibuat *width* dan *height* dengan *match\_parent,* pada tampilan ini menggunakan *ScrollView* agar tampilan layar dapat di *scroll* untuk melihat informasi dan menggunakan *tag LinearLayout* yang berguna untuk mengatur tampilan secara *linear*, dan menambahkan *tag* *cardview* agar tampilan akan membentuk persegi sesuai dengan kategori yang telah ditentukan. Berikut listing progam activity\_detail\_bangun*.xml* :

|  |
| --- |
| <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" |

Kode diatas adalah tag *Scrollview* yang berguna agar tampilan layar dapat di *scroll* dengan memberika ukuran *width* dan *height* dengan *match\_parent* agar sesuai dengan tampilan layar pada android.

|  |
| --- |
| <LinearLayout  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <android.support.v7.widget.CardView  android:layout\_margin="8dp"  android:elevation="8dp"  app:cardCornerRadius="16dp"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <LinearLayout  android:layout\_margin="16dp"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"> |

Kode diatas adalah masih lanjutan pada tampilan desain menggunakan *tag* *LinearLayout* dan memberikan *tag CardView,* dan tag *cardCornerRadius* digunakan untuk memberikan efek melengkung pada setiap ujung *Cardview*.

|  |
| --- |
| <TextView  android:drawableLeft="@drawable/ic\_detail\_bangun"  android:drawablePadding="8dp"  android:text="Fungsi Bangun Gedung"  android:textColor="@color/colorPrimaryText"  android:textSize="18dp"  android:textStyle="bold"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content" />  <View  android:layout\_marginTop="8dp"  android:layout\_marginBottom="8dp"  android:background="@color/colorAccent"  android:layout\_width="match\_parent"  android:layout\_height="1dp"></View> |

Lanjutan pada tampilan *detailactivity* menambahkan tag *TextView* yang berguna untuk memberikan text pada tampilan android. Tulisan tersebut terdapat pada atribut android:text. Untuk *tag View* hanya memberikan garis bawah yang berguna sebagai pembatas antar judul setiap kategori *cardview* dan isi data yang terdapat pada *cardview.*

|  |
| --- |
| <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Penginput :"  android:textStyle="bold" />  <TextView  android:layout\_marginTop="4dp"  android:id="@+id/txtDetailPetugas survei"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="-"  android:textColor="@color/colorSecondaryText" /> |

Pada TextView diatas diberikan id txtDetailPetugas survei yang berguna untuk menampilkan data yang akan didapatkan dari *website* dan nama petugas survei akan muncul pada textview diatas.

1. Selanjutnya kode dimasukan ke dalam detailbangun*.class* untuk menjalankan fungsi-fungsi yang terdapat pada program yang terdapat pada tampilan detailbangun. Berikut kode yang yang terdapat pada detailbangun.java :

|  |
| --- |
| public class DetailBangun extends BaseActivity {  @BindView(R.id.txtDetailPetugas survei)  TextView txtpetugas survei;  @BindView(R.id.txtFungsiGedung)  TextView txtFungsiGedung;  @BindView(R.id.txtJenisGedung)  TextView txtJenisGedung;  @BindView(R.id.txtJenisFungsi)  TextView txtJenisFungsi; |

Diatas adalah kode java untuk detailbangun.java, @BindView diatas adalah library *butterknife* untu menggantikan kata *findviewbyid* ketika deklarasi variabel memanggil atribut pada tampilan activity\_detail\_bangun.xml.

|  |
| --- |
| @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_detail\_bangun);  ButterKnife.bind(this);  setTitle("Detail Bangunan");  Bundle data = getIntent().getExtras();  bangunan = data.getParcelable("bangunan"); |

Berdasarkan kode diatas adalah lanjutan dari kode detailbangun.java. kode diatas berfungsi sebagai pemanggil data pada class bangunan atau pada kamus data pada android di baris kode Bundle data = getIntent().getExtra(). Pada baris selanjutnya bangunan = data.getParcelable(“bangunan”) untuk memanggil satu bundle data pada class bangunan.

|  |
| --- |
| if (bangunan != null) {  id\_bangunan = bangunan.getId\_bangunan();  txtpetugas survei.setText(bangunan.getNama());  txtFungsiGedung.setText(bangunan.getFungsi\_bangunan());  txtJenisGedung.setText(bangunan.getJenis\_bangunan());  txtJenisFungsi.setText(bangunan.getJenis\_fungsi\_bangunan());  txtNamaDepar.setText(bangunan.getNama\_departemen());  txtAlmtDepar.setText(bangunan.getAlamat\_departemen());  txtNoIkmn.setText(bangunan.getNo\_ikmn()); |

Berdasarkan kode diatas digunakan untuk *set* data untuk pada atribut yang yang terdapat pada activity\_detail\_bangunan.xml yang telah dibuat. Pada saat data telah masuk kedalam android maka data akan masuk kedalam bangunan.class dan akan didapatkan di set pada textview.

**4.3.2.7 Pembuatan Halaman Form Survei**

Halaman survei yang dibuat digunakan untuk petugas survei dalam memasukan data gedung. Halaman form survei dibuat dengan nama *class* spinner\_fungsi.java. spinner\_fungsi merupakan *activity* yang memilik file berupa java *class* dan file berupa xml. Berikut kode form survei yang telah dibuat :

1. Pada halaman form survei memiliki banyak *widget* atau atribut yang dimasukan kedalam tampilan dikarenakan kebutuhan data yang akan dimasukan oleh petugas survei bangun gedung. Berikut tampilan kode form survei yang terdapat pada form xml yaitu activity\_spinner\_fungsi.xml:

|  |
| --- |
| <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"  tools:showIn="@layout/app\_bar\_main"  tools:context="com.example.alfhanrf.skripsihehe.Spinner\_fungsi"> |

Tampilan form survei bangung gedung menggunakan dasar layar *scrollview* sehingga tampilan dapat di*scroll*  sampai dengan konten terakhir pada layar. Menggunakan *width* dan *height* dengan ukuran *match\_parent* agar ukuran sesuai dengan layar *smartphone* yang digunakan.

|  |
| --- |
| <android.support.v7.widget.CardView  android:layout\_margin="8dp"  android:elevation="8dp"  app:cardCornerRadius="12dp"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <Spinner  android:id="@+id/Spinnerfungsigedung"  android:layout\_width="match\_parent"  android:entries="@array/fungsi\_gedung"  android:layout\_height="50dp" /> |

Pada tampilan form survei menggunakan *widget* atau atribut berupa *cardview* agar masukan data sesuai dengan kategori pada saat dimasukan oleh petugas survei, sehingga petugas dapat mebedakan setiap kategori pada saat memasukan data bangun gedung kedalam basis data. Kode diatas juga menggunakan Spinner sebagai pilihan yang telah ditentukan atau dikategorikan. Spinner tersebut memiliki *id* Spinnerfungsigedung yang akan dipanggil pada java class.

|  |
| --- |
| <android.support.design.widget.TextInputLayout  android:layout\_marginTop="8dp"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">  <android.support.design.widget.TextInputEditText  android:id="@+id/NoIkmn"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="Nomor IKMN"  android:inputType="number" />  </android.support.design.widget.TextInputLayout> |

Pada kode diatas menggunakan tambahan *library design widget texinput layout* agar tampilan untuk petugas layout lebih mudah. Kode diatas berfungsi memasukan text seperti diatas adalah NoIkmn. Tipe data masukan adalah number sehingga pada saat petugas survei melakukan pendaataan *keyboard* *smartphone* akan berubah menjadi angka.

|  |
| --- |
| <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="340dp"  android:orientation="vertical">  <fragment xmlns:android="http://schemas.android.com/apk/res/android"  android:name="com.google.android.gms.maps.SupportMapFragment"  android:id="@+id/map"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </LinearLayout> |

Kode diatas masih berada pada file spinner\_fungsi.xml, *tag* diatas memanggil *map fragment* agar *map* dapat ditampilkan dalam form survei gedung untuk menetukan lokasi atau titik bangun gedung dengan GPS ( *Global Positioning Services )* pada saat petugas survei melakukan pendataan posisi bangun gedung.

1. Selanjutanya adalah pembuatan *method* dan fungsi pada kelas spinner\_fungsi.java. kode tersebut digunakan untuk memasukan data surveyor kedalam basis data yang telah disediakan. Berikut kode spinner\_fungsi.java :

|  |
| --- |
| public class Spinner\_fungsi extends BaseActivity implements AdapterView.OnItemSelectedListener, OnMapReadyCallback {  @BindView(R.id.Spinnerfungsigedung)  Spinner fungsigedung;  @BindView(R.id.Spinnerfungsigedung1)  Spinner Spinnerfungsigedung1;  @BindView(R.id.Spinnerfungsigedung2)  Spinner Spinnerfungsigedung2;  @BindView(R.id.NamaDepar)  TextInputEditText NamaDepar; |

Pada kode diatas adalah kode header *class* Spinner\_fungsidengan *extends baseactivity* dan implement AdapeterView.OnItemSelectedListener yang digunakan untuk membuat *method* untuk menu *spinner* dan *OnMapReadyCallback* digunakan untuk memanggil *map* pada tampilan form survei. @BindView digunakan untuk deklarasikan *widget* yang terdapat file spinner\_fungsi.xml kode diatas mendeklarasikan 3 *spinner* dan 1 *TextInputEditText.*

|  |
| --- |
| ArrayAdapter<String> datafungsi = new ArrayAdapter<String>(this, android.R.layout.simple\_spinner\_item, fungsigedunglist); datafungsi.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);  fungsigedung.setAdapter(datafungsi); |

kode diatas adalah *adapter* untuk *spinner* yang digunakan, fngsi tersebut memanggil array dari fungsigedunglist yang terdapat pada folder values dan file string agar kategori dari array gedung tersebut masuk kedalam *spinner.*

|  |
| --- |
| fungsigedung.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  @Override  public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  if (position == 0) {  Spinnerfungsigedung1.setEnabled(true);  Spinnerfungsigedung1.setAdapter(datafungsibag1hunian);  Spinnerfungsigedung2.setEnabled(false);  refValue = 1;  } else if (position == 1) {  Spinnerfungsigedung1.setEnabled(true);  Spinnerfungsigedung1.setAdapter(datafungsibag1keagamaan);  Spinnerfungsigedung2.setEnabled(false);  refValue = 2; |

Fungsi kode diatas adalah *spinner* fungsi gedung apabila pada saat dipilih atau akan memasukan nilai berupa angka, angka tersebut digunakan untuk memberikan *id\_reference* kedalam basis data. Apabila dipilih salah nilai *spinner* maka akan terdapat kondisi dimana akan mengganti nilai *string-array* pada spinnerfungsi1 dan akan mengganti nilai pada string array spinnerfungsi2

|  |
| --- |
| Button gpsGetPosisi = (Button) findViewById(R.id.getPosisi);  gpsGetPosisi.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  locationManager=(LocationManager) getSystemService(Context.LOCATION\_SERVICE);  if(!locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER)) {  buildAlertMessageNoGPS();  }elseif (locationManager.isProviderEnabled(LocationManager.GPS\_PROVIDER)) {  getLocation();  if (ActivityCompat.checkSelfPermission(Spinner\_fungsi.this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED && ActivityCompat.checkSelfPermission(Spinner\_fungsi.this  Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  return;  }  mMap.setMyLocationEnabled(true); |

Kode diatas adalah fungsi mendapatkan latitude dan longitude posisi dari petugas survei. Ketika menekan button dapatkan posisi gedung, maka android akan meminta izin utuk mengakses GPS serta memanggil *method* getLocation(). Apabila GPS petugas survei gedung tidak menyala maka akan menajalakan *method* buildAlertMessageNoGPS();

|  |
| --- |
| private void getLocation()  {  if (ActivityCompat.checkSelfPermission(Spinner\_fungsi.this, Manifest.permission.ACCESS\_FINE\_LOCATION)  != PackageManager.PERMISSION\_GRANTED && ActivityCompat.checkSelfPermission(Spinner\_fungsi.this,  Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {  mMap.setMyLocationEnabled(true);  ActivityCompat.requestPermissions(Spinner\_fungsi.this, new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION}, REQUEST\_LOCATION);  } else {  Location location = locationManager.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);  LatLng target = new LatLng(location.getLatitude(), location.getLongitude()); |

Kode diatas adalah *method* getLocation() yang berfungsi untuk mendapatkan posisi latitude longitude posisi petugas survei bangun gedung. Kode diatas melakukan pengecekan apakah sudah mendapatkan perizinan untuk mendapatkan lokasi.

|  |
| --- |
| if (location != null) {  double latti = location.getLatitude();  double longi = location.getLongitude();  latitude = String.valueOf(latti);  longitude = String.valueOf(longi);  getLatitude.setText(latitude);  getLongitude.setText(longitude);  CameraPosition.Builder builder = new CameraPosition.Builder();  builder.zoom(18);  builder.target(target); this.mMap.animateCamera(CameraUpdateFactory.newCameraPosition(builder.build()));  mMap.addMarker(new MarkerOptions().position(new LatLng(location.getLatitude(),  location.getLongitude())).title("Posisi Anda"));  } else  Toast.makeText(this, "Tidak Mendapatkan Posisi", Toast.LENGTH\_SHORT).show();  }  } |

Kode diatas adalah untuk mendapatakan *latitude longitude* serta memberikan *marker* pada titik petugas survei, kode diatas apabila *location* tidak bernilai null maka akan mengambil titik *latitude* dan *longitude* kedalam form masukan posisi latitude dan longitude. apabila GPS tidak dapat mendapatkan posisi maka akan memberikan pesan “Tidak Mendapatkan Posisi”.

|  |
| --- |
| private void buildAlertMessageNoGPS() {  final AlertDialog.Builder builder = new AlertDialog.Builder(this);  builder.setMessage("Harap Menayalakan GPS")  .setCancelable(false)  .setPositiveButton("Nyalakan GPS ", new DialogInterface.OnClickListener() {  public void onClick(final DialogInterface dialog, final int id) {  startActivity(new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS));  }  })  .setNegativeButton("Tidak", new DialogInterface.OnClickListener() {  public void onClick(final DialogInterface dialog, final int id) {  dialog.cancel();  }  });  final AlertDialog alert = builder.create();  alert.show();  } |

Kode diatas adalah *method* buildAlertMessageNoGPS() yang berfungsi apabila GPS *smartphone* petugas survei bangun gedung tidak menyala makan *method* ini akan menjalankan fungsinya yaitu membuka *setting smartphone* untuk menyalakan GPS.

|  |
| --- |
| InputData.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (getIntent().getStringExtra(ConstantsUtils.ACTION).equals(ConstantsUtils.ACTION\_NEW)) {  if (fieldCheck()) {  final AlertDialog.Builder builder = new AlertDialog.Builder(Spinner\_fungsi.this);  builder.setMessage("Apakah Data yang Di Input Benar ?")  .setCancelable(false)  .setPositiveButton("Ya ", new DialogInterface.OnClickListener() {  public void onClick(final DialogInterface dialog, final int id) {  createData(parseData());  dialog.dismiss();  finish();  final AlertDialog alert = builder.create();  alert.show();  } else {  showToast("Ada Data Yang Belum diisi");  } |

Kode diatas ada *Listener* atau aksi yang akan dilakukan apabila *button* yang ber *id* InputData ditekan atau diklik. Pada saat tombol ditekan maka akan menjalankan *Method* fieldcheck, apabila kondisi fieldcheck dapat dilewati maka akan memunculkan dialog “Apakah Data yang DiInput Benar ?” apabila menekan “Ya” maka akan menjalankan *method* createData dan parseData.

|  |
| --- |
| private void createData(Bangunan bangunan) {  UserSession userSession = new UserSession(Spinner\_fungsi.this);  mRegProgres.setTitle("Sending Data");  mRegProgres.setMessage("Please Wait...");  mRegProgres.setCanceledOnTouchOutside(false);  APIInterface apiInterface = APIClient.getClientAuthorize().create(APIInterface.class);  Call<RequestResponse> call = apiInterface.masukandata(userSession.getUserToken(),  userSession.getIdUser(), bangunan);  call.enqueue(new Callback<RequestResponse>() {  @Override  public void onResponse(Call<RequestResponse> call, Response<RequestResponse> response) {  if (response.isSuccessful()) {  mRegProgres.dismiss();  showToast("Data Berhasil Di Inputkan");  }  } |

Berdasarkan kode diatas adalah *method* createData, *method* createData berfungsi untuk memasukan data dari android kedalam basis data yang ada di website. Pada saat menjalankan *method* ini akan memanggil API masukandata yang telah dibuat pada APIClient, apabila berhasil akan mendapatkan response yaitu IdUser dan UserToken dengan parameter yaitu bangunan. Jika data berhasil masuk kedalam database android akan memberikan pesan “Data Berhasil Di Inputkan”

|  |
| --- |
| @Override  public void onFailure(Call<RequestResponse> call, Throwable t) {  mRegProgres.dismiss();  showToast("Data gagal Dikirim : " + t.toString());  }  });  } |

Apabila data gagal masuk akan memberikan pesan “Data gagal Dikirim dan memberikan pesan error pada android.

|  |
| --- |
| private Bangunan parseData() {  Bangunan bangunan = new Bangunan();  UserSession userSession = new UserSession(this);  if (bangunan != null) {  bangunan.setId\_bangunan(bangunan.getId\_bangunan());  }  bangunan.setUser\_id(userSession.getUserId());  bangunan.setNama\_departemen(NamaBangun.getText().toString().trim());    bangunan.setJenis\_fungsi\_bangunan(Spinnerfungsigedung2.getSelectedItem().toString().trim());  bangunan.setId\_reference(refValue); bangunan.setAlamat\_departemen(AlmtDepartemen.getText().toString().trim());  bangunan.setNo\_ikmn(NoIkmn.getText().toString().trim());  bangunan.setNo\_hdno(NoHdno.getText().toString().trim());  bangunan.setTelpon(TelponBgnGedung.getText().toString().trim());  bangunan.setEmail(EmailBgnGedung.getText().toString().trim()); |

Kode diatas adalah *method* parseData, *method* tersebut digunakan untuk mengambil text dan pilihan *spinner* yang telah dimasukan kedalam *form* survei. setelah data dimasukan data akan dirubah menjadi string dan akan menghapus spasi yang tidak terpakai pada teks.

**4.3.2.8 Pembuatan *class* RecycleViewAdapterBangunan**

*RecycleView* ini berfungsi sebagai pengganti *listview* dan *gridview* pada android agar konten yang disediakan menjadi lebih rapih dan terstruktur. *RecyclerView* juga mempunyai animasi default sesuai standar Google Material Design saat menambahkan atau menghapus elemen. Sedangkan untuk mengatur posisi item pada list, *RecyclerView* menggunakan *LayoutManagers*. *RecycleView* menggunakan *ViewHolder* untuk menyimpan refrensi data yang akan ditampilkan. Pada aplikasi survei bangun gedung dibuat recycleview untuk menampilkan data bangun gedung secara *list* selain dengan cara melihat titik pada *maps.* Dengan menggunakan *recycleview* ini *user* lebih mudah mencari informasi bangun gedung dan dapat mencari dengan menggunakan fitur *search* yang telah tersedia. Berikut kode RecycleView pada aplikasi survei bangun gedung :

|  |
| --- |
| public class RecycleViewAdapterBangunan extends RecyclerView.Adapter<RecycleViewAdapterBangunan.ViewHolder> implements AdapterView.OnItemLongClickListener {  private static final String TAG = RecyclerView.class.getSimpleName();  private Context mContext;  private List<Bangunan> mList;  private List<Bangunan> bangunan;  private AdapterView.OnItemClickListener mListener;  private OnItemLongClickListener mLongListener;  UserSession userSession;  MainActivity mainActivity;  boolean dialogLongEnabled ; |

Pada kode diatas adalah header *class* dari recycleviewadapterbangunan, kelas tersebut digunakan untuk menambahkan *method* dari adapteronclicklonglistener. Pada kode diatas juga terdapat deklarasi variabel yang akan digunakan pada *class* recycleviewadapterbangunan.

|  |
| --- |
| public RecycleViewAdapterBangunan(Context context, List<Bangunan> bangunan,boolean dialogLongEnabled) {  this.mContext = context;  this.bangunan = bangunan;  this.dialogLongEnabled = dialogLongEnabled;  } |

Pada kode diatas adalah sebuah *method* yang bersifat public yang dapat dipanggil di lain kelas. *Method* tersebut menggunakan parameter context dimana context adalah deklarasi halaman fragment itu sendiri, lalu list<Bangunan> adalah kamus data yang telah dibuat pada android, serta Boolean dialogLongEnabled sebagai parameter untuk menentukan aksi ketika item ditekan dan ditahan untuk beberapa waktu.

|  |
| --- |
| @Override  public void onBindViewHolder(final ViewHolder holder, final int position) {  userSession = new UserSession(mContext);  final Bangunan item = bangunan.get(position);  holder.txtNamaBangunan.setText(item.getNama\_bangunan());  holder.txtNamaDepar.setText("Nama Departemen : " + item.getNama\_departemen());  holder.txtNamaPemilik.setText("Nama Pemilik : " + item.getNama\_pemilik\_tanah());  final ColorGenerator generator = ColorGenerator.MATERIAL; // or use DEFAULT  final int color2 = generator.getColor(item.getNama\_bangunan());  final TextDrawable.IBuilder builder = TextDrawable.builder()  .beginConfig()  .endConfig()  .round();  TextDrawable ic2 = builder.build(item.getNama\_bangunan().toUpperCase().substring(0, 1), color2);  holder.imgIconBangunan.setImageDrawable(ic2); |

kode tersebut digunakan untuk menampilkan apa saja item pada recycleview, terdapat inisialisasi class UserSession yang digunakan untuk mengambil UserId dan UserToken agar data yang terdapat pada basis data dapat digunakan dengan menggunakan 2 kunci tersebut. setelah mendapatkan 2 kunci untuk mendapatkan data maka holder atau item akan menampilkan nama bangunan, nama departemen dan nama pemilik tanah pada holder atau pada item recycleview. Color generator digunakan untuk membuat image pada holder dengan menggunakan fungsi substring untuk mengambil huruf terdepan dari nama banguna setelah itu akan dibuat image dengan menggunakan huruf tersebut dan dengan warna yang random.

|  |
| --- |
| holder.cvItemBangunan.setOnLongClickListener(new View.OnLongClickListener() {  @Override  public boolean onLongClick(final View v) {  userSession = new UserSession(mContext);  //Surveyor = 2 , User Eksternal = 3  String id = userSession.getUserRoleStr();  Log.d("role", id);  if(id.equals("2") && dialogLongEnabled == true) { |

Kode diatas adalah pemisahan hak akses setiap role, kode diatas adalah pemisahan *user* yang dapat memanggil *delete* dan *edit* data. OnLongClickListener adalah *method* yang digunakan untuk memberikan aksi ketika *list* data atau holder pada recycleview. Dan apabila *user role* bernilai 2 maka *method* dari OnLongClickListener dapat digunakan.

|  |
| --- |
| holder.cvItemBangunan.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  Intent i = new Intent(mContext, DetailBangun.class);  i.putExtra("bangunan", bangunan.get(position));  mContext.startActivity(i);  }  }); |

Berdasarkan kode diatas berfungsi apabila *list* item atau holder ditekan atau dipilih, maka akan berpindah kedalam *class* DetailBangun dan tampilan activity\_detail\_bangun.xml mengambil data bangungan dengan pilihan dari *id* position holder atau *list* data.

|  |
| --- |
| private void deleteData(Bangunan position) {  final BangunanView bangunanView = new BangunanView();  userSession = new UserSession(mContext);  final ProgressDialog mRegProgres = new ProgressDialog(mContext);  mRegProgres.setTitle("Processing Data");  mRegProgres.setMessage("Please Wait...");  mRegProgres.setCanceledOnTouchOutside(false);  mRegProgres.show();  userSession = new UserSession(mContext);  APIInterface apiInterface = APIClient.getClientAuthorize().create(APIInterface.class);  Call<RequestResponse> call = apiInterface.deleteBangunan(userSession.getUserToken(),userSession.getIdUser(), position.getId\_bangunan());  call.enqueue(new Callback<RequestResponse>() { |

Kode diatas adalah *method* untuk menghapus data oleh petugas survei yang memasukan data. Dalam kode tersebut berfungsi untuk menggunakan *progressbar* dengan memberikan pesan “Processing Data” . *method* tersebut memanggil API yang telah dibuat pada APIClient yaitu deleteBangunan dengan menggunakan header UserToken dan UserId sebagai kuncinya dan memberikan parameter *id* bangunan untuk menghapus data bangunan yang bersangkutan.

|  |
| --- |
| @Override  public void onResponse(Call<RequestResponse> call, Response<RequestResponse> response) {  mRegProgres.dismiss();  Toast.makeText(mContext,"DataBerhasilDi Hapus",Toast.LENGTH\_SHORT).show();  bangunanView.loadDataBangunan();  } |

setelah berhasil menghapus data maka akan memberikan pesan “Data Berhasil Dihapus” dan mengambil data terbaru dengan memanggil *method* yang terdapat pada bangunanView yaitu loadDataBangunan().

|  |
| --- |
| public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  @BindView(R.id.txtNamaBangunan)  TextView txtNamaBangunan;  @BindView(R.id.txtNamaDepar)  TextView txtNamaDepar;  @BindView(R.id.txtNamaPemilik)  TextView txtNamaPemilik;  @BindView(R.id.imgIconBangunan)  ImageView imgIconBangunan;  @BindView(R.id.cvItemBangunan)  CardView cvItemBangunan;  public ViewHolder(View itemView) {  super(itemView);  ButterKnife.bind(this, itemView);  } |

Kode datas adalah berguna untuk deklarasi *widget* atau atribut yang terdapat pada file item\_bangunan.xml untuk membentuk *list* item atau holder dari recycleview.

**4.3.2.9 Pembuatan Halaman BangunanView**

Halaman ini digunakan untuk menampilkan data yang telah dibuat pada RecycleViewAdapterBangunan. Pada *activity* ini memiliki file BangunanView.java dan activity\_bangunan\_view.xml. berikut listing kode yang terdapat pada halaman BangunanView :

1. Pembuatan tampilan halaman BangunanView ini hanya memanggil *widget* recycleview untuk menampilkan data yang dibuat pada recycleviewadapterbangunan. berikut kode yang terdapat pada file activity\_bangunan\_view.xml :

|  |
| --- |
| <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="com.example.alfhanrf.skripsihehe.BangunanFrag">  <!-- TODO: Update blank fragment layout -->  <android.support.v7.widget.RecyclerView  android:id="@+id/rvBangunan"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  </android.support.v7.widget.RecyclerView>  </FrameLayout> |

Berdasarkan kode di atas pada tampilan bangunanview menggunakan ukuran *width* dan *height* menggunakan *match\_parent.* Tampilan bangunanview ini menampilkan *fragment*. *Fragment* adalah sub-layar yang ada pada tampilan bangunanview. *Fragment* tersebut diber nama *id* rvBangunanyang menggunakan ukuran *match\_parent*.

1. Setelah tampilan bangunan view dibuat, kemudian masukan kode java yang terdapat pada BangunanView.java. yang berfungsi untuk menjalankan *method* yang dibutuhkan pada tampilan BangunanView. Berikut listing kode yang terdapat pada *class* BangunanView :

|  |
| --- |
| public class BangunanView extends BaseActivity{  private List<Bangunan> bangunan = new ArrayList<>();  private RecycleViewAdapterBangunan viewAdapterBangunan;  @BindView(R.id.recycleview)  RecyclerView recyclerView; |

Kode diatas adalah header kelas dari bangunanview yang menggunakan *extends* dari baseactivity sehingga tak perlu inisialisasi *class* pada saat menggunakan *widget* atau *method* yang terdapat pada baseactivity. Kode diatas adalah deklarasi variabel RecycleVieAdapterBangunan dan Bangunan sebagai ArrayList.

|  |
| --- |
| public void loadDataBangunan() {  UserSession userSession = new UserSession(BangunanView.this);  mRegProgres.setTitle("Mendaptkan Data");  mRegProgres.setMessage("Tunggu Sebentar... ");  mRegProgres.setCanceledOnTouchOutside(false);  mRegProgres.show();  APIInterface apiInterface = APIClient.getClientAuthorize().create(APIInterface.class);  Call<BangunanList> call = apiInterface.getSemuaBangunan(userSession.getUserToken(), userSession.getIdUser());  call.enqueue(new Callback<BangunanList>() {  @Override  public void onResponse(Call<BangunanList> call, Response<BangunanList> response) {  mRegProgres.dismiss();  bangunan = response.body().getSemuaBangunan();  viewAdapterBangunan = new RecycleViewAdapterBangunan(BangunanView.this, bangunan,false);  recyclerView.setAdapter(viewAdapterBangunan);  } |

Kode diatas adalah *method* untuk mengambil data bangunan dengan memanggil API yang terdapat pada *class* APIClient. API yang digunakan adalah getSemuaDataBangunan untuk mengambil seluruh data bangunan yang terdapat pada basis data bangungedung. Apabila data berhasil didapatkan dan masuk kedalam android, data tersebut akan diatur oleh holder dan recycleviewadapterbangunan. Pengambilan data dapat dilakukan dengan menggunakan kunci UserToken, dan IdUser.

**4.3.2.10 Pembuatan Halaman BangunanViewHistory**

Pembuatan halaman ini memiliki kesamaan dengan BangunanView, pada tampilan atau struktur yang ditampilkan pada layar, perbedaan pada tampilan ini adalah kategori data yang ditampilkan. Data yang ditampilkan adalah data yang telah dimasukan oleh petugas survei. Pada tampilan ini digunakan untuk menghapus dan mengubah data yang telah dimasukan oleh petugas survei. Berikut kode yang terdapat pada BangunanViewHistory.java :

|  |
| --- |
| private void loadDataBangunan() {  final UserSession userSession = new UserSession(BangunanViewHistory.this);  mRegProgres.setTitle("Mendapatkan Data Dirimu");  mRegProgres.setMessage("Tunggu Sebentar... ");  mRegProgres.setCanceledOnTouchOutside(false);  mRegProgres.show();  APIInterface apiInterface = APIClient.getClientAuthorize().create(APIInterface.class);  Call<BangunanList> call = apiInterface.getBangunanSurveyor(userSession.getUserToken(),  userSession.getIdUser(), userSession.getUserId()); |

Pada kode diatas adalah *method* loadDataBangun untuk tampilan BangunanViewHistory. Perbedaan data yang ditampilkan adalah terdapat parameter UserId pada tampilan BangunanViewHistory sehingga data yang akan ditampilkan pada *list* atau holder adalah data yang dimasukan berdasarkan UserId petugas survei.

**4.3.2.11 Pembuatan *Class* UserSession**

*Class* UserSession ini dibuat sebagai kamus untuk kunci yang dibutuhkan untuk mendapatkan data, menghapus data, mengedit data, dan memasukan data. UserSession ini menyimpan data *user* berupa nama, *role,* user\_id, id\_user dan token. Sehingga jika sebuah parameter membutuhkan data *user* tersebut dapat digunakan dengan inisialisasi *class* UserSession memanggil *method* yang terdapat dalam *class.* Berikut kode yang terdapat pada *class* UserSession :

|  |
| --- |
| public class UserSession {  private SharedPreferences preferences;  private SharedPreferences.Editor editor;  private Context mContext;  private static final String PREFER\_NAME = "SurveyBangunGedung";  private static final String IS\_USER\_LOGIN = "LoginStatus";  private static final String KEY\_ID\_USER = "id\_user";  private static final String KEY\_USER\_ROLE = "user\_role";  private static final String KEY\_USER\_ID = "user\_id";  private static final String KEY\_USER\_TOKEN = "token";  private static final String KEY\_USER\_NAME = "nama"; |

Kode diatas adalah deklarasi variabel bertipe *string* dan diberikan sebuah *key* untuk pemanggilan data *user.* Setiap data *user* yang di deklarasikan memiliki *key* yang berbeda seperti KEY\_ID\_USER yang digunkan untuk id\_user, KEY\_USER\_NAME yangi digunakan untuk nama.

|  |
| --- |
| public void setLoginSession(String id, String token, String nama, String userRole, String user\_id ) {  editor.putBoolean(IS\_USER\_LOGIN, true);  editor.putString(KEY\_USER\_ID, user\_id);  editor.putString(KEY\_ID\_USER, id);  editor.putString(KEY\_USER\_TOKEN, token);  editor.putString(KEY\_USER\_NAME, nama);  editor.putString(KEY\_USER\_ROLE, userRole);  editor.commit();  } |

Kode diatas adalah *method* untuk melakukan menyimpan informasi *user* pada saat melakukan *login* di android.

|  |
| --- |
| public int getUserID() {  return preferences.getInt(KEY\_ID\_USER, 0);  }  public int getUserRole() {return preferences.getInt(KEY\_USER\_ROLE, 0);  }  public String getUserToken() {  return preferences.getString(KEY\_USER\_TOKEN, null);  }  public String getUserId() {  return preferences.getString(KEY\_USER\_ID, null);  }  public String getIdUser() {  return preferences.getString(KEY\_ID\_USER, "null");  }  public String getUserName() {  return preferences.getString(KEY\_USER\_NAME, null);  }  public String getUserRoleStr() {  return preferences.getString(KEY\_USER\_ROLE, null);  }  public boolean isUserLoggedIn() {  return preferences.getBoolean(IS\_USER\_LOGIN, false);  }  } |

Kode diatas adalah variabel yang menyimpan data *user* sehingga jika API memerlukan data yang menggunakaninformasi *user* dapat menggunakan *method* getIdUser atau

getUserToken tergantung dari parameter yang dibutuhkan.

**4.3.2.12 Pembuatan *Class* BaseActivity**

BaseActivity digunakan sebagai dasar fungsi atau method yang sering digunakan pada *class* lain agar *method* tersebut tidak perlu lagi dibuat pada *class* yang membutuhkan. Berikut kode yang terdapat pada BaseActivity :

|  |
| --- |
| Public class BaseActivity extends AppCompatActivity {  private Boolean connStatus;  protected ProgressDialog mRegProgres;  @Override  protected void onCreate(@Nullable Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  checkInternet();  mRegProgres = new ProgressDialog(this);  mRegProgres.setTitle("Getting Data");  mRegProgres.setMessage("Please Wait...");  mRegProgres.setCanceledOnTouchOutside(false);  } |

Berdsarkan kode diatas adalah header dari *class* BaseActivity dan deklarasi variabel yang yang digunakan pada *class* tersebut. mRegProgres = new ProgressDialog(this) digunakan untuk mendeklarasikan bahwa mRegProgres adalah *widget* ProgressDialog. Dan pada baris kode mRegProgres.setTitle("Getting Data") yaitu memberikan pesan “Getting Data “ ketika menjalankan ProgressDialog.

|  |
| --- |
| private void checkInternet(){  ConnectivityManager ConnectionManager =(ConnectivityManager) getSystemService(Context.CONNECTIVITY\_SERVICE);  NetworkInfo networkInfo=ConnectionManager.getActiveNetworkInfo();  if(networkInfo != null && networkInfo.isConnected()==true ) {  connStatus = true;  }  else {  connStatus = false;  showToast("No Internet Connection");  }  }  protected void showToast(String message){  Toast.makeText(this, message, Toast.LENGTH\_SHORT).show();  }  } |

Kode diatas adalah *method* checkInternet, *method* tersebut digunakan untuk mengecek koneksi intenet, apabila internet dalam keadaan mati, maka android akan memberikan pesan “No Internet Connection”. Kode *method* showToast digunakan untuk memberikan pesan tanpa perlu mengetikan kode *Toast*.

**4.3.2.13 Pembuatan Class Bangunan**

*Class* Bangunan digunakan sebagai penyimpan data yang didapatkan oleh android dari basis data yang disediakan oleh *website.* Pada *class* bangunan menggunakan *implement* Parcelable. Parcelable tersebut adalah fitur yang berfungsi untuk mengirimkan secara bersamaan kepada *intent* atau *activity.* Berikut ini adalah kode *class* Bangunan.java :

|  |
| --- |
| public class Bangunan implements Parcelable {  @SerializedName("user\_id")  @Expose  private String user\_id;  @SerializedName("nama")  @Expose  private String nama;  @SerializedName("id\_bangunan")  @Expose  private String id\_bangunan;  @SerializedName("id\_reference")  @Expose  private int id\_reference; |

Kode diatas adalah header dari *class* Bangunan yang mengimplementasikan parcelable. Fungsi kode diatas adalah untuk mendeklarasikan variabel apa saja yang diperlukan untuk data yang akan diolah oleh android.

|  |
| --- |
| public Bangunan (String nama, String id\_bangunan, int id\_reference,String fungsi\_bangunan, String jenis\_bangunan, String jenis\_fungsi\_bangunan, String nama\_departemen, String alamat\_departemen, String no\_ikmn, String no\_hdno, String telpon, String email, String date\_input, String nama\_bangunan, String alamat\_bangunan,String klasifikasi\_bangunan, String jumlah\_lantai\_bangunan,  String luas\_lantai\_bangunan, String luas\_basement, String ketinggian\_bangunan,  String date\_selesai, String nama\_pemilik\_tanah, String no\_ipt,  String no\_bkt, String jenis\_kepemilikan\_tanah, String alamat\_tanah,  String luas\_tanah, String data\_peruntukan\_resmi,  String kdb, String klb, String kdh, String ktb, String nama\_departemen\_dulu, String alamat\_departemen\_dulu, String no\_ikmn\_dulu, String no\_hdno\_dulu,String telpon\_dulu, String email\_dulu, String no\_imb\_terdahulu, String no\_islf\_terdahulu, String latitude, String longitude){  this.nama = nama;  this.id\_bangunan = id\_bangunan;  this.id\_reference = id\_reference;  this.fungsi\_bangunan = fungsi\_bangunan;  this.jenis\_bangunan = jenis\_bangunan; |

Kode diatas adalah *method* Bangunan yang menggunakan parameter data yang akan digunakan oleh android. Data tersebut akan dideklarasikan pada baris this.nama = nama dan seterusnya.

|  |
| --- |
| public String getNama() {  return nama;  }  public void setNama(String nama) {  this.nama = nama;  }  public String getId\_bangunan(){return id\_bangunan;}  public void setId\_bangunan (String id\_bangunan) {this.id\_bangunan = id\_bangunan;} |

Kode diatas adalah variabel yang menyimpan setiap data yang telah didatapakan android dari basis data. Variabel tersebut menggunakan parameter sesuai dengan nama yang disimpan dalam bentuk String.

**4.3.2.13 Pembuatan *Class* Logout**

*class* ini dibuat untuk keluar dari akun dalam aplikasi survei bangun gedung di android. Berikut kode yang terdapat pada logout.java :

|  |
| --- |
|  |

**4.4 Uji Coba**

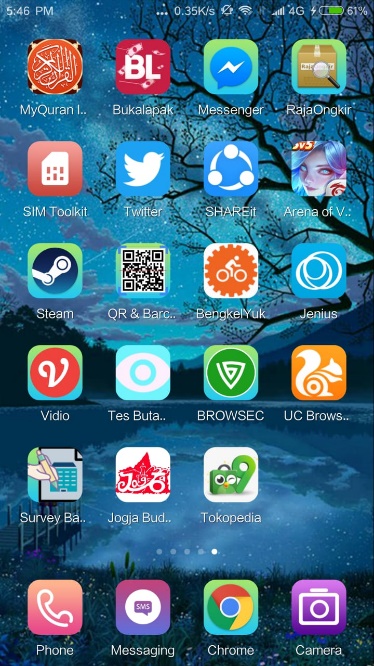
Berikut uji coba yang dilakukan penulis dalam pengujian aplikasi :

**4.4.1 Uji Coba Aplikasi**

Pada skenario uji coba aplikasi, pengguna diminta untuk menjalankan aplikasi. Berikut scenario uji coba :

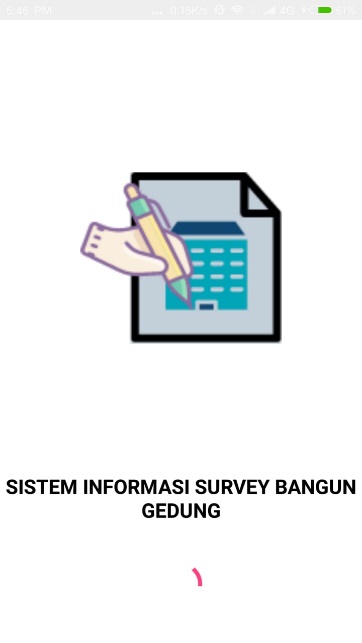
1. Pengguna membuka aplikasi dengan menekan icon pada smartphone

seperti gambar 4.17



Gambar 4.17 Tampilan Icon Aplikasi pada Ponsel

1. Kemudian akan tampil *Splashscreen* aplikasi seperti gambar 4.18



Gambar 4.17 Tampilan Icon Aplikasi pada Ponsel

1. Setelah tampilan splashscreen, maka muncul tampilan *login* seperti

gambar 4.18



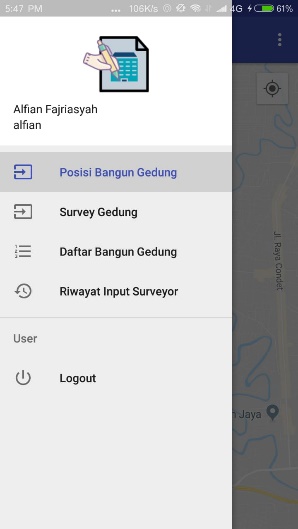
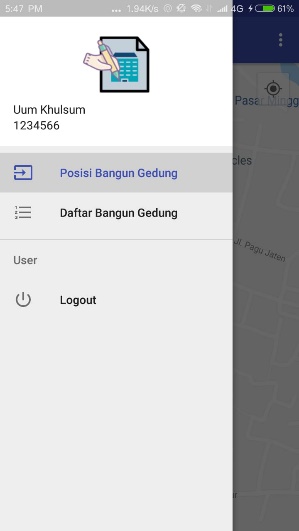
Gambar 4.18 Tampilan *Login* Aplikasi pada Ponsel

1. Saat *user* berhasil *login* akan muncul menu utama yaitu *maps* dan titik bangun gedung yang telah di survei seperti pada gambar 4.19



Gambar 4.19 Tampilan Utama Aplikasi pada Ponsel

1. Apabila *user* menakan menu yang terdapat pada sebelah kiri atas akan muncul tampilan menu seperti pada gambar 4.20(A) dan 420(B).

1. (B)

Gambar 4.20 (A) *Login* aplikasi dengan *role* petugas survei, (B) *Login* aplikasi dengan *role* user eksternal