**Universidad Tecnológica de la Riviera Maya**

**T.S.U. en Tecnología de la Información y Comunicaciones.**

**Calidad en el Desarrollo de Software**

23/01/2017

**Docente: Cristian Bernal Bera**

**Alumno: Cruz Alcántara Jorge Luis**

Contenido

[Planeación 3](#_Toc474909892)

[Objetivos 3](#_Toc474909893)

[Alcance 3](#_Toc474909894)

[Requerimientos 3](#_Toc474909895)

[Interfaz 5](#_Toc474909896)

[Codificación 9](#_Toc474909897)

[Definición de líneas de código 9](#_Toc474909898)

[Convención de nombres 9](#_Toc474909899)

[Nombre de archivos 10](#_Toc474909900)

[Nombre de clases 10](#_Toc474909901)

[Nombre de funciones / métodos 10](#_Toc474909902)

[Nombre de variables / propiedades 11](#_Toc474909903)

[Nombre de constantes 11](#_Toc474909904)

[Nombres de las tablas y datos 12](#_Toc474909905)

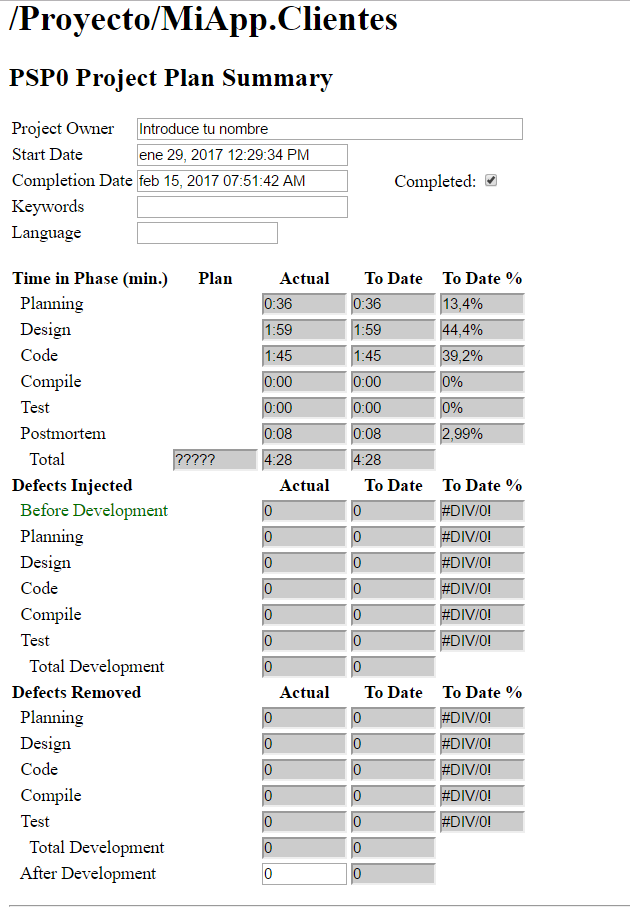
[Estructuras de control 12](#_Toc474909906)

[If 12](#_Toc474909907)

[Clases 13](#_Toc474909908)

[Ejemplo: 13](#_Toc474909909)

[Postmortem 14](#_Toc474909910)

[ 14](#_Toc474909911)

[Análisis de Datos 15](#_Toc474909912)

[Tiempo en Fase a la Fecha 15](#_Toc474909913)

# Planeación

## Objetivos

El objetivo de este proyecto es crear una aplicación para un restaurante donde puedan registrar clientes y dependiendo de las áreas se puedan registrar diversas categorías.

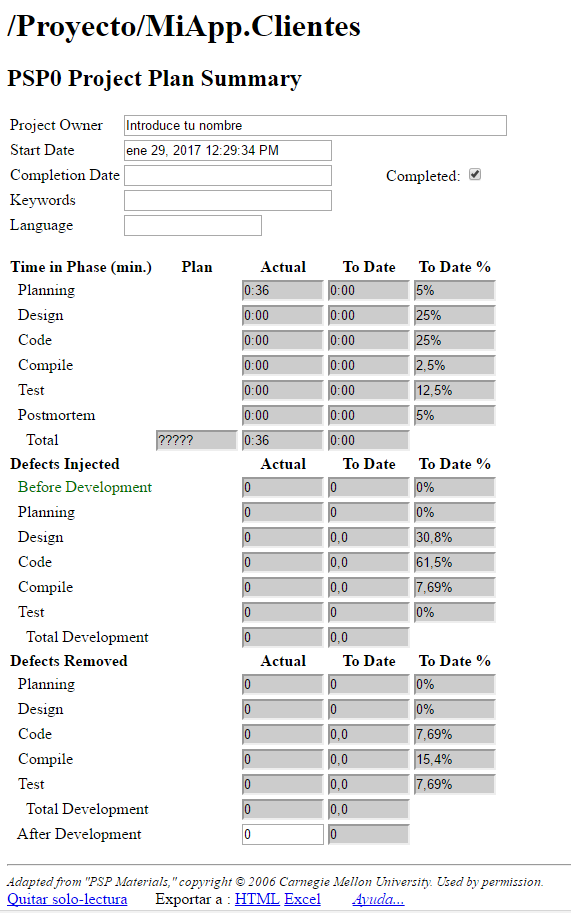
El sistema consiste en datos casuales donde se llevará un control de los datos ya que serán clientes de entrada por salida y no requerirán más que sus datos comunes.

## Alcance

Se necesitará de un diseño sencillo donde se registre fácilmente al cliente y sus datos sea de fácil localización en el momento que se necesiten, cuando se inicia la aplicación se le pedirán los datos de dichos clientes donde se guardaran en una base de datos y después de que dicho cliente vuelva a regresar sea autocompletado.

## Requerimientos

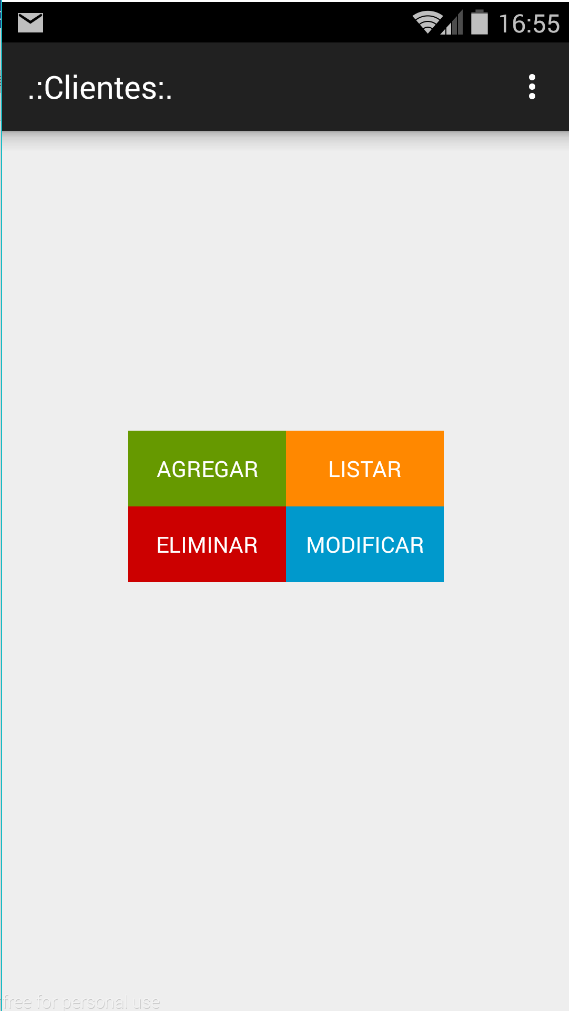
Los requisitos para que el cliente pueda ser registrado son nombre, apellidos, teléfono y un correo electrónico, donde el nombre y apellido son obligatorios, el teléfono de tipo alfanumérico y el correo electrónico no se podrá repetir ni escribir otro formato que no sea el de un correo, si no cuenta con correo electrónico puede quedar en blanco.



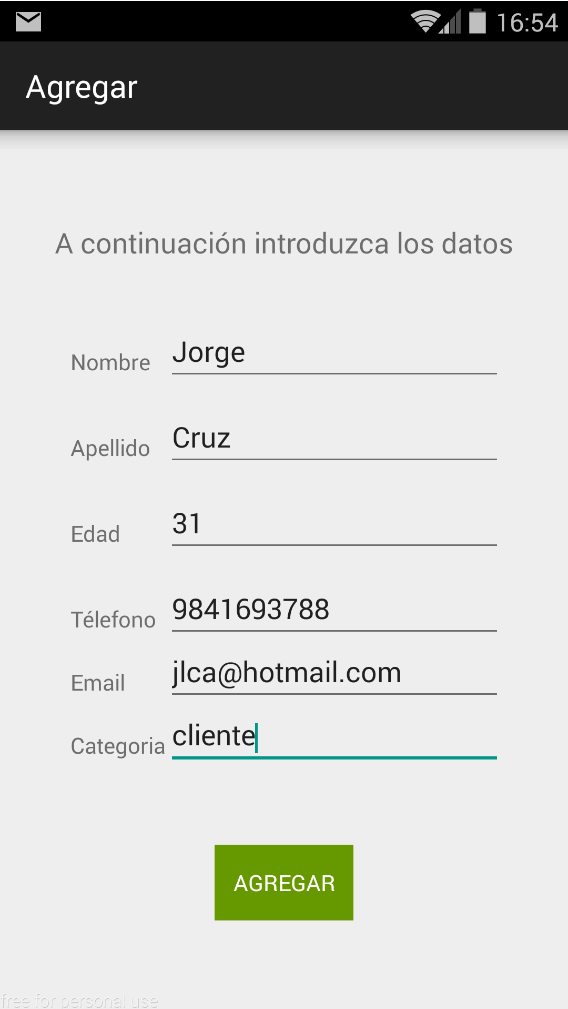
# Interfaz

La interfaz de inicio tendrá cuatro botones básicos en los que se incluye el CRUB, agregar, eliminar, mostrar y modificar.

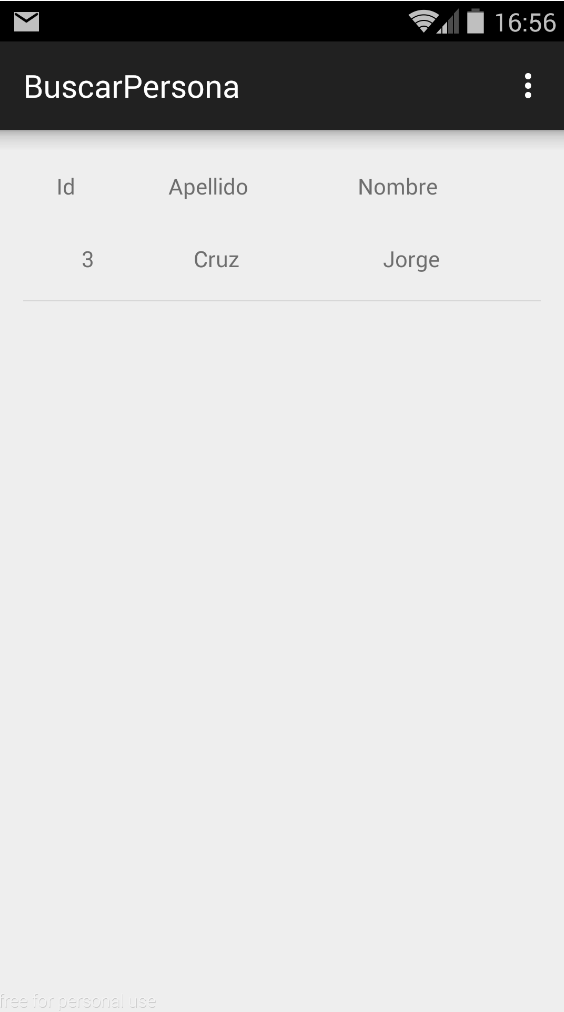
Desde aquí se ira accediendo a los datos necesarios para las consultas.



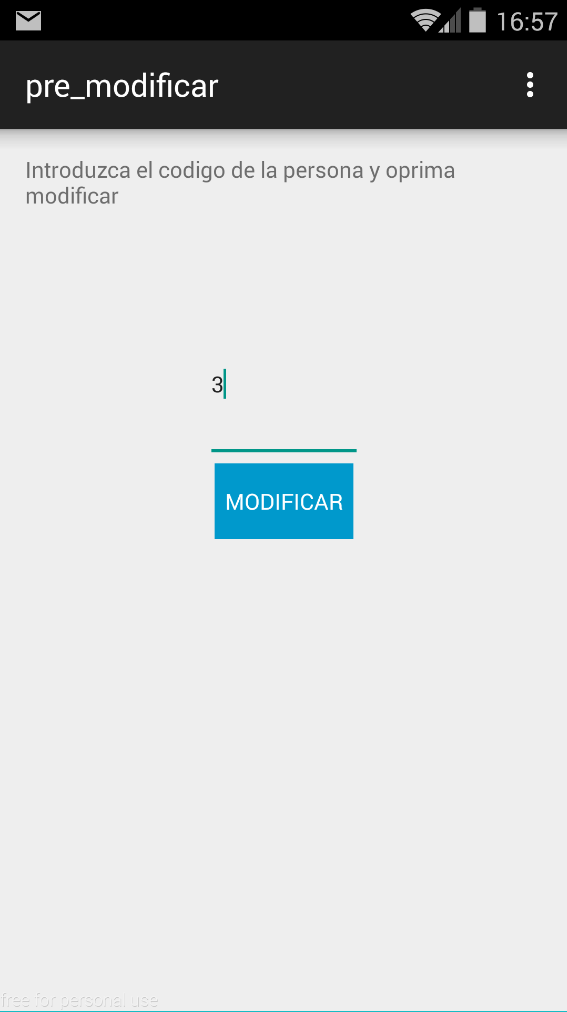
En el menú agregar se encuentran los datos nombre, apellidos, edad, teléfono, correo y categoría.



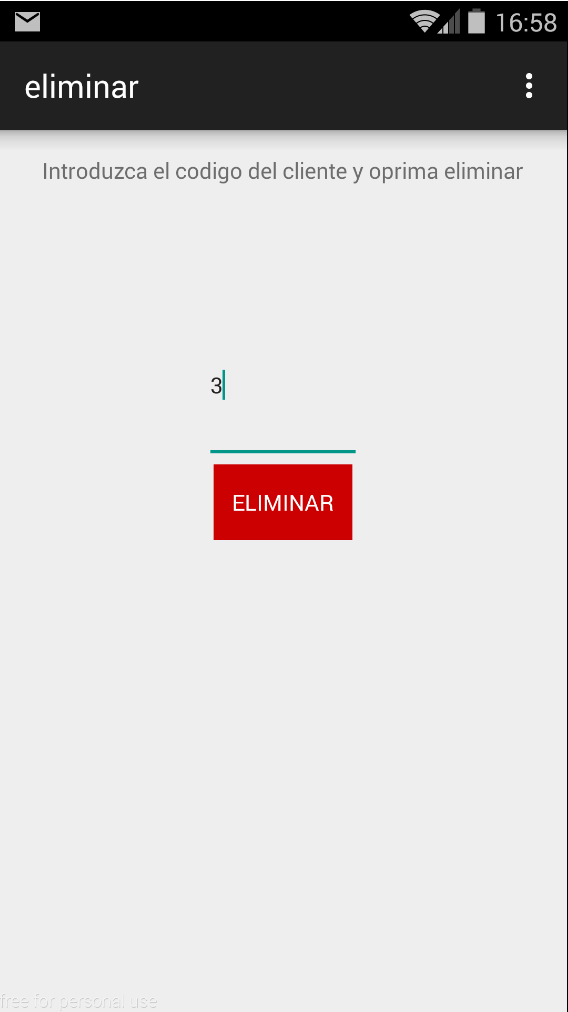
La pantalla donde se listan los clientes tendrá solo dos datos principales que son el nombre y el apellido, incluyendo su número de ingreso que será autoincrementadle.



Pantalla para modificar clientes, accediendo con el id de cada cliente.



Eliminar Cliente



# Codificación

## Definición de líneas de código

El lenguaje a utilizar es Java para Android con una base de datos interna perteneciente al IDE Android Studio 2.1.

## Convención de nombres

### Nombre de archivos

El nombre de cada clase o archivo que será definido en Android Studio será escrito con la primera letra en mayúscula, por ejemplo BuscarCliente, Clientes y los subarchivos estarán escritos en minúsculas.

### Nombre de clases

Las clases de preferencias se deben de escribir en inglés, pero por falta de este dominio y para un mejor entendimiento se escribirán con el nombre perteneciente a su función.

Ejemplo:

*public class Agregar extends ActionBarActivity {*

*}*

Esta clase lo que ara es agregar un cliente.

### Nombre de funciones / métodos

Los nombres de funciones deberán de estar escritos en minúsculas de preferencia y ser llamados por su definición.

*public void confirmacion(){*

*AlertDialog.Builder dlgAlert = new AlertDialog.Builder(this);*

*dlgAlert.setMessage("Se ha agregado exitosamente!");*

*dlgAlert.setTitle("Agregar Cliente");*

*dlgAlert.setPositiveButton("Ok",*

*new DialogInterface.OnClickListener() {*

*public void onClick(DialogInterface dialog, int which) {*

*//dismiss the dialog*

*}*

*});*

*dlgAlert.setCancelable(true);*

*dlgAlert.create().show();*

*}*

### Nombre de variables / propiedades

Estos serán los nombres de las variables que se definirán…

*BDClientes bdClientes;*

*EditText nombre\_input;*

*EditText apellido\_input;*

*EditText edad\_input;*

*EditText telefono\_input;*

*EditText email\_input;*

*EditText categoria\_input;*

*private static final int DATABASE\_VERSION = 1;*

*private static final String DATABASE\_NAME = "clientes.db";*

*public static final String TABLA\_CLIENTES = "clientes";*

*public static final String COLUMN\_ID = "\_id";*

*public static final String COLUMN\_NOMBRE = "nombre";*

*public static final String COLUMN\_APELLIDO = "apellido";*

*public static final String COLUMN\_EDAD = "edad";*

*public static String COLUMN\_TEL = "telefono";*

*public static String COLUMN\_EMAIL = "email";*

*public static String COLUMN\_CATEGORIA = "categoria";*

*private int \_id;*

*private String \_nombre;*

*private String \_apellido;*

*private int \_edad;*

*private String \_telefono;*

*private String \_email;*

*private String \_categoria;*

### Nombre de constantes

Una constante es un valor que no debe cambiar y tiene que ser bien identificado para ser leído por cualquier programador, este ayudara a la legibilidad.

Ejemplo:

*Clientes cliente = new Clientes;*

### Nombres de las tablas y datos

Los nombres de las tablas son llamados en plural y los campos en singular, serán definidos de la siguiente manera…

*DATABASE\_NAME = "clientes.db";*

*TABLA\_CLIENTES = "clientes";*

*COLUMN\_ID = "\_id";*

*COLUMN\_NOMBRE = "nombre";*

*COLUMN\_APELLIDO = "apellido";*

*COLUMN\_EDAD = "edad";*

*COLUMN\_TEL = "telefono";*

*COLUMN\_EMAIL = "email";*

*COLUMN\_CATEGORIA = "categoria";*

## Estructuras de control

### If

*//Listar por ID*

*if (c != null) {*

*c.moveToFirst();*

*}*

*//Listar a todos los clientes*

*if (c != null) {*

*c.moveToFirst();*

*}*

*//Simplificar declaración \* 5*

*if (id == R.id.action\_settings) {*

*return true;*

*}*

*return super.onOptionsItemSelected(item);*

## Clases

### Ejemplo:

*public class Agregar extends ActionBarActivity {*

*}*

*public class BDClientes extends SQLiteOpenHelper{*

*}*

*public class BuscarCliente extends ActionBarActivity {*

*}*

*public class Clientes {*

*}*

*public class eliminar extends ActionBarActivity {*

*}*

*public class ListarClientes extends CursorAdapter {*

*}*

*public class MainActivity extends ActionBarActivity {*

*}*

*public class modificar extends ActionBarActivity {*

*}*

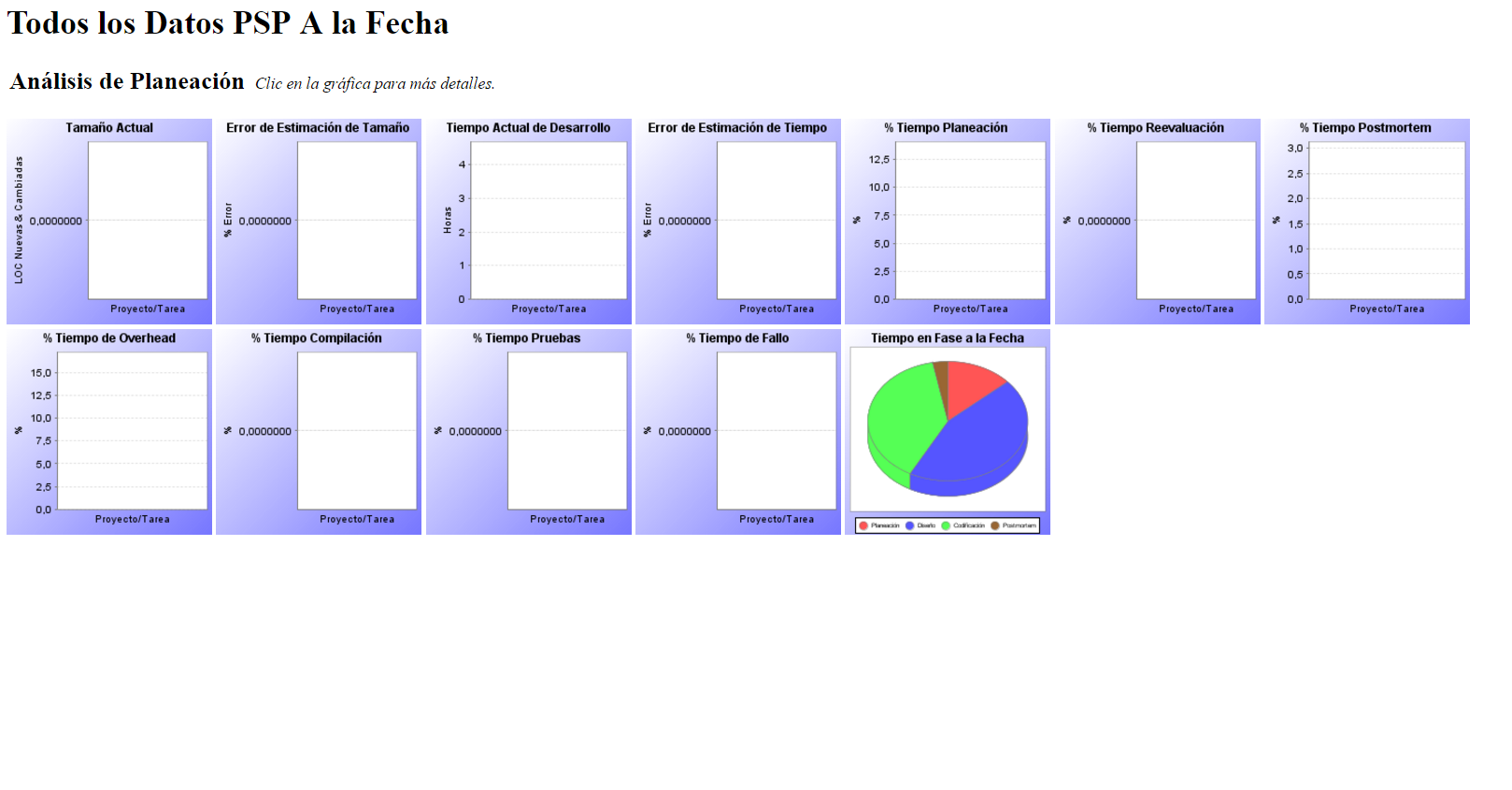
*public class pre\_modificar extends ActionBarActivity {*

*}*

# Postmortem

### 

# Análisis de Datos



# Tiempo en Fase a la Fecha

