****

Estándar de programación

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 08/07/2020 | 1.0 | Versión 0.1 Estándar de programación | Jesús Condori |

Revisión y aprobación

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Rol | Firma | Observaciones |
| Renzo Moreno | Jefe del proyecto, diseñador |  |  |
| Aldo Lopez | Desarrollador |  |  |
| Jesús Condori | Analista, Téster |  |  |

**ÍNDICE**

[Revisión y aprobación 1](#_Toc45116052)

[1 OBJETIVO 3](#_Toc45116053)

[2 ALCANCE 3](#_Toc45116054)

[3 DEFINICIONES 3](#_Toc45116055)

[3.1 Metodología de Programación 3](#_Toc45116056)

[3.2 Sistema 3](#_Toc45116057)

[3.3 Aplicación 3](#_Toc45116058)

[4 RESPONSABILIDAD 3](#_Toc45116059)

[4.1 De los analistas programadores 3](#_Toc45116060)

[4.2 De los programadores 4](#_Toc45116061)

[4.3 Del analista supervisor 4](#_Toc45116062)

[5 DESARROLLO 4](#_Toc45116063)

[5.1 Bases de datos 4](#_Toc45116064)

[5.1.1 Tablas 5](#_Toc45116065)

[5.1.2 Columnas 6](#_Toc45116066)

[5.2 Laravel 7](#_Toc45116067)

[5.2.1 Generalidades 7](#_Toc45116068)

[5.2.2 Convenciones de Capitalización 7](#_Toc45116069)

[5.2.3 Para Tipos 8](#_Toc45116070)

[5.2.3.1 Métodos 8](#_Toc45116071)

[5.2.3.1 Propiedades 8](#_Toc45116072)

[5.2.3.1 Eventos 8](#_Toc45116073)

[5.2.3.1 Campos 9](#_Toc45116074)

[5.2.3.1 Parámetros 9](#_Toc45116075)

[5.2.4 Controles 9](#_Toc45116076)

[5.3 Android 10](#_Toc45116077)

[5.3.1 Nombrado de archivos 10](#_Toc45116078)

[5.3.2 Estándar Java 12](#_Toc45116079)

[5.3.3 Estándar XML 14](#_Toc45116080)

# 1 OBJETIVO

El objetivo del presente documento es definir los estándares de programación en cuanto a denominación de variables, tablas, columnas, herramientas de interfaz, etc. con el fin de desarrollar una programación fácil de entender por cualquier programador y contar con una herramienta de fácil documentación de los sistemas de información.

# 2 ALCANCE

El estándar de programación se utilizará en todos los desarrollos de software realizados para el “Maquipuray”.

# 3 DEFINICIONES

## 3.1 Metodología de Programación

Permite realizar mantenimientos en menor tiempo y costo, migrar código consistentemente, habilita a los programadores moverse entre proyectos diferentes y asegura la comunicación técnica entre desarrolladores.

## 3.2 Sistema

Conjunto de elementos que interactúan entre sí para lograr un objetivo común.

## 3.3 Aplicación

Es un programa que se descarga e instala en el dispositivo móvil de un usuario

# 4 RESPONSABILIDAD

## 4.1 De los analistas programadores

Cada analista programador deberá utilizar el estándar de programación en los desarrollos de software posteriores a la fecha de aprobación del mismo.

## 

## 4.2 De los programadores

Cada programador deberá seguir el estándar de programación según quedó expresado por los analistas en los documentos respectivos y según indica el presente manual.

## 4.3 Del analista supervisor

Controlar la aplicación del estándar de programación en los desarrollos de software posteriores a la fecha de aprobación del mismo.

**4.4 Del jefe de proyecto**

Asegurarse que el estándar de desarrollo sea utilizado en todos los proyectos de software.

# 5 DESARROLLO

## 5.1 Bases de datos

Esta sección es una guía genérica para el nombramiento de objetos comunes en DBMS. Aunque las bases de datos tienen aún más objetos que los discutidos en este documento, se cubren los más importantes, que son:

* Tablas
* Columnas
* Restricciones
* Vistas
* Procedimientos Almacenados

Para todos los objetos de la base de datos:

* Limitar el nombre a 50 caracteres.
* Evitar el uso de subguiones, excepto cuando esté especificado en este documento.
* Use sólo letras o subguiones. Trate de evitar el uso de números y limite el uso de subguiones para cumplir con los estándares.
* Nombre de las tablas están con abreviación, seguido de un guion bajo con el nombre de la tabla.
* Use una letra como el primer carácter del nombre. (No comience los nombres con subguiones o números).
* Limite el uso de abreviaciones.
* Limite el uso de acrónimos.
* Haga los nombres leíbles (No deben sonar muy enrevesados).
* Evite el uso de espacios en los nombres incluso si el sistema los permite.

### 5.1.1 Tablas

Al nombrar las tablas de su base de datos, considere los otros pasos en el proceso de desarrollo. Tenga en cuenta que lo más probable es que tenga que utilizar los nombres que le da a sus tablas varias veces como parte de otros objetos, por ejemplo, los procedimientos, los activadores o las vistas pueden contener referencias al nombre de la tabla. Es por ello que se deben mantener los nombres tan simples y cortos como sea posible.

* Nombres en singular. Los nombres de las tablas deben estar expresados de manera singular, por ejemplo “Empleado” en vez de “Empleados”. Esta regla es aplicable porque las tablas son patrones para almacenar una entidad como una entrada.
* **No use Prefijos**. No use prefijos a menos que sean necesarios para ayudar organizar las tablas en grupos relacionados o para distinguirlos de otro grupo de tablas. Usualmente los prefijos causarán que tipee una gran cantidad de caracteres innecesarios.
* **Notación**. Para todos los nombres de las tablas, iniciaran con la abreviación “tbl” en minúscula seguido de un guion bajo y posteriormente el nombre de la tabla en minúscula.
* **Caracteres especiales**.

Tampoco uses números en los nombres de tus tablas. Esto generalmente apunta a un modelo de datos mal diseñado o tablas con particiones irregulares. Tampoco uses espacios en los nombres de tus tablas. Si bien la mayoría de los sistemas de bases de datos pueden manejar nombres que incluyen espacios, sistemas como MySQL Server requiere que agregue corchetes alrededor del nombre cuando haga referencia a él (como [nombre de la tabla] por ejemplo), lo que va en contra de la regla de mantener las cosas lo más cortas y simples posible

* Tablas de Intersección. Las tablas de intersección, que manejan muchas a muchas relaciones, deben nombrarse concatenando los nombres de las tablas que tienen una relación de uno a muchos con la tabla de unión. Por ejemplo, podría tener las tablas "Doctor" y "Paciente". Como los médicos pueden tener muchos pacientes y los pacientes pueden tener muchos médicos (especialistas), usted necesita una tabla para guardar los datos de esas relaciones en una tabla de unión.

### 

### 5.1.2 Columnas

Al nombrar las columnas, tenga en cuenta que estas son miembros de la tabla, por lo que no necesitan ninguna mención del nombre de la tabla en el nombre.

Todos los nombres de las columnas de las tablas utilizaran la notación LowerCamelCase, igual que la anterior con la excepción de que la primera letra es minúscula.

* **Llaves Primarias.** Para los campos que son la llave primaria de la tabla e identifican a cada entrada de la table, el nombre debería ser simplemente “id” + [nombreDeTabla] (ej. en una tabla de Cliente, el campo de la llave primaria debería llamarse idCliente. Un prefijo es agregado ya que “id” es una *keyword* (palabra clave) en SQL Server, y se tendría que encerrar en corchetes al referirse a él.
* **Llaves Foráneas.** Los campos de llaves foráneas deberán tener el mismo nombre que tienen en la tabla padre, de donde el campo es la llave primaria. Por ejemplo, en la tabla de Clientes, el campo de la llave primaria debe llamarse “idCliente”. En una tabla de Pedidos donde el id del cliente es guardada, el nombre deberá llamarse también “idCliente”. Esto hace que el código SQL sea mucho más legible (“File inner join Directory on File.FileId = Directory.FileId”), mientras que este ejemplo tiene mucha información repetitiva y confusa (File inner join Directory on File.FileId\_Pk = Directory.FileId\_Fk).
* **Llaves Compuestas.** Si se tienen tablas con llaves compuestas, es recomendable que se cree una columna de tipo *identity* para ser usada como la llave primaria de la tabla.
* **Prefijos.** No prefija sus campos con "fld\_" o "Col\_", ya que debería ser obvio en las sentencias de SQL qué elementos son columnas (antes o después de la cláusula FROM). Tampoco utilice un prefijo de tipo de datos para el campo, por ejemplo, "IntCustomerId" para un tipo numérico o "VcName" para un tipo varchar. Estos "obstruyen" nuestros nombres y agregan poco valor; la mayoría de los campos de enteros se pueden identificar fácilmente como tales y los campos de caracteres deberían ser revisados por su longitud en el “Object Browser” de todas formas
* **Nombres según el tipo de dato.** Los campos tipo “Bit“ deben recibir nombres booleanos afirmativos como "EstaEliminado", "TienePermiso" o "EsValido" para que el significado de los datos en el campo no sea ambiguo; los nombres booleanos negativos son más difíciles de leer cuando se verifican los valores en T-SQL debido a los dobles negativos (por ejemplo, "No use NoTienePermiso"). Si el campo contiene información de fecha y / o hora, la palabra "Fecha" u "Hora" debe aparecer en algún lugar del nombre del campo. A veces también es apropiado agregar la unidad de tiempo al nombre del campo, especialmente si el campo contiene datos como números enteros ("3" o "20"). Estos campos deben nombrarse como "RuntimeHours" o "ScheduledMinutes"

**Tamaño del campo**. Los nombres de los campos no deben tener más de 50 caracteres y todos deben ser lo menos extensos posible. Sin embargo, no se debe sacrificar la legibilidad por brevedad y evitar el uso de abreviaturas a menos que sea absolutamente necesario.

## 5.2 Laravel

Esta sección describe reglas y recomendaciones para desarrollar aplicaciones y bibliotecas de clases utilizando el lenguaje Laravel. El objetivo es definir pautas para imponer un estilo y un formato coherentes y ayudar a los desarrolladores a evitar errores y fallas comunes.

### 5.2.1 Generalidades

* Siempre use los nombres LowerCamelCase*.*
* Evite nombres que sean solo mayúsculas o solo minúsculas.
* No utilice nombres que comiencen con un número.
* Siempre elija nombres significativos y específicos.
* Variables y propiedades deben describir una entidad no su tipo o tamaño.
* Evite usar abreviaturas a menos que el nombre completo sea excesivo.
* Evitar abreviaturas de más de 5 caracteres.
* Cualquier abreviatura debe ser ampliamente conocida y aceptada.
* Use mayúsculas para las abreviaturas de dos letras, y LowerCamelCase para abreviaturas más largas.
* No incluya el nombre de la clase padre dentro de un nombre de propiedad. Ejemplo: Empleado.Nombre no Empleado.EmpleadoNombre

### 5.2.2 Convenciones de Capitalización

Las pautas en esta sección presentan un método simple para usar el caso que, cuando se aplica de manera consistente, hace que los identificadores de tipos, miembros y parámetros sean fáciles de leer.

### 5.2.3 Para Tipos

Los tipos se componen de miembros: métodos, propiedades, eventos, constructores y campos. Las siguientes secciones describen las pautas para nombrar miembros de tipo.

#### 5.2.3.1 Métodos

Debido a que los métodos son el medio para tomar medidas, las pautas de diseño requieren que los nombres de los métodos sean verbos o frases verbales. Seguir esta guía también sirve para distinguir los nombres de los métodos de los nombres de propiedad y de tipo, que son frases nominales o adjetivas.

Ejemplo:

**public static function PromocionListarCategoria($idCategoria)  
{  
 return DB::*table*('tbl\_promocion as p')  
 ->select(  
 'p.idPromocion',  
 'p.idNegocio',  
 'p.codigoPromocion',  
   
 )**

#### 5.2.3.1 Propiedades

A diferencia de otros miembros, a las propiedades se les debe dar nombres de frases o adjetivos. Esto se debe a que una propiedad se refiere a datos, y el nombre de la propiedad refleja eso. Lower Camel Case siempre se utiliza para nombres de propiedades.

* Nombre las propiedades utilizando un sustantivo, frase sustantivo o adjetivo.

#### 5.2.3.1 Eventos

Los eventos siempre se refieren a alguna acción, ya sea que esté sucediendo o que haya ocurrido. Por lo tanto, al igual que con los métodos, los eventos se nombran con verbos, y el tiempo verbal se usa para indicar la hora en que se produce el evento.

* Nombre los eventos con un verbo o una frase verbal. Por ejemplo: Coloreado, Cliqueado, Activado, etc.
* Asigne nombres de eventos con un concepto de antes y después, utilizando los tiempos presente y pasado. Por ejemplo: Cerrando o Cerrado.

#### 5.2.3.1 Campos

Las pautas de nomenclatura de campos se aplican a los campos estáticos públicos y protegidos. Los campos internos y privados no están cubiertos por las directrices, y los campos de instancia pública o protegida no están permitidos por las directrices de diseño del miembro.

* Use Lower Camel Case en nombres de campo.
* Nombre los campos usando un sustantivo, frase sustantivo o adjetivo.

#### 5.2.3.1 Parámetros

Más allá de la razón obvia de la legibilidad, es importante seguir las pautas para los nombres de los parámetros porque los parámetros se muestran en la documentación y en el diseñador cuando las herramientas de diseño visual proporcionan la funcionalidad de navegación de clase.

* Considere el uso de nombres basados en el significado de un parámetro en lugar del tipo de parámetro.

### 5.2.4 Controles

La siguiente tabla muestra las abreviaturas aceptadas para controles estándar encontrados en el marco de trabajo. Laravel

|  |  |
| --- | --- |
| Control | Abreviación |
| Button | btn |
| CheckBox | cb |
| CheckBoxList | cbl |
| DropDownList | ddl |
| HiddenField | hdn |
| Label | lbl |
| Panel | pnl |
| TextBox | txt |
| DataGrid | dtg |
| ListBox | lst |
| RadioButton | rdo |
| RadioButtonList | rdl |
| Table | tbl |
| GridView | gv |
| FormView | fv |

## 5.3 Android

### 5.3.1 Nombrado de archivos

**Archivos de clases**

Los nombres de las clases son escritos en **UpperCamelCase**.

Para las clases que extienden un componente de Android, el nombre de la clase debe de terminar con el nombre del componente; por ejemplo: **SignInActivity, SignInFragment, ImageUploaderService**.

**Archivos de resources (recursos)**

Los archivos de recursos son escritos en **lowercase\_underscore**.

* **Archivos Drawable**

Convenciones de nombre para drawables:

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Prefijo** | **Ejemplo** |
| Action Bar | ab\_ | ab\_stacked.png |
| Button | btn\_ | btn\_send\_pressed.png |
| Dialog | dialog\_ | dialog\_top.png |
| Divider | divider\_ | divider\_horizontal.png |
| Icon | ic\_ | ic\_star.png |
| Menu | menu\_ | menu\_submenu\_bg.png |
| Notification | notification\_ | notification\_bg.png |
| Tabs | tab\_ | tab\_pressed.png |

Convenciones para iconos:

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Prefijo** | **Ejemplo** |
| Icons | ic\_ | ic\_star.png |
| Launcher Icons | ic\_launcher | ic\_launcher\_calendar.png |
| Menu Icons y Action Bar Icons | ic\_menu | ic\_menu\_archive.png |
| Status Bar Icons | ic\_stat\_notify | ic\_stat\_notify\_msg.png |
| Tab Icons | ic\_tab | ic\_tab\_recent.png |
| Dialog Icons | ic\_dialog | ic\_dialog\_info.png |

Convenciones para selectores de estado:

|  |  |  |
| --- | --- | --- |
| **Estado** | **Sufijo** | **Ejemplo** |
| Normal | \_normal | btn\_order\_normal.png |
| Presionado | \_pressed | btn\_order\_pressed.png |
| Enfocado | \_focused | btn\_order\_focused.png |
| Deshabilitado | \_disabled | btn\_order\_disabled.png |
| Seleccionado | \_selected | btn\_order\_selected.png |

* **Archivos Layout**

Los archivos de diseño deben coincidir con el nombre de los componentes de Android para los que están destinados, pero se debe mover el nombre del componente de nivel superior al principio. Por ejemplo, si estamos creando un diseño para SignInActivity, el nombre del archivo de diseño debe ser activity\_sign\_in.xml.

|  |  |  |
| --- | --- | --- |
| Componente | Clase | Layout |
| Activity | UserProfileActivity | activity\_user\_profile.xml |
| Fragment | SignUpFragment | fragment\_sign\_up.xml |
| Dialog | ChangePasswordDialog | dialog\_change\_password.xml |
| AdapterView item |  | item\_person.xml |
| Partial layout |  | partial\_stats\_bar.xml |

Un caso ligeramente diferente es cuando estamos creando un diseño que va a ser inflado por un adaptador, por ejemplo, para rellenar un ListView. En este caso, el nombre del diseño debe comenzar con item\_. Tenga en cuenta que hay casos en que estas reglas no serán posibles de aplicar. Por ejemplo, al crear archivos de diseño que están destinados a formar parte de otros diseños. En este caso debes usar el prefijo partial\_.

* **Archivos Menu**

Al igual que en los archivos layout, los archivos de menú deben coincidir con el nombre del componente. Por ejemplo, si estamos definiendo un archivo de menú que se va a utilizar en UserActivity, el nombre del archivo debe ser activity\_user.xml Una buena práctica es no incluir la palabra menú como parte del nombre porque estos archivos ya están ubicados en el directorio del menú.

* **Archivos Values**

Los archivos de recursos en la carpeta **values** deben ser plurales, por ejemplo, strings.xml, styles.xml, colors.xml, dimens.xml, attrs.xml

### 5.3.2 Estándar Java

**Definición de campos y denominación.**

Los campos deben definirse en la parte superior del archivo y deben seguir las reglas de denominación que se enumeran a continuación.

* Los nombres de los campos **privados** y **no estáticos** comienzan con **m**.
* Los nombres de los campos **privados** y **estáticos** comienzan con **s**.
* Otros campos comienzan con una letra minúscula.
* Los campos finales estáticos (**constantes**) son TODAS\_LETRAS\_MINUSCULA.

Ejemplo:

public class MiClase {  
 public static final int UNA\_CONSTANTE = 42;  
 public int campoPublico;  
 private static MiClase sSingleton;  
 int mPaquetePrivado;  
 private int mPrivado;  
 protected int mProtegido;  
 }

**Tratar las siglas como palabras**

|  |  |
| --- | --- |
| **Bien** | **Mal** |
| XmlHttpRequest | XMLHTTPRequest |
| getCustomerId | getCustomerID |
| String url | String URL |
| long id | long ID |

**Orden de miembros de Clase**

No existe una única solución correcta para esto, pero usar un orden lógico y coherente mejorará la capacidad de aprendizaje y legibilidad del código. Es recomendable utilizar el siguiente orden:

1. Constantes
2. Campos o atributos
3. Constructores
4. Métodos sobrecargados (públicos o privados)
5. Métodos públicos
6. Métodos privados
7. Clases internas o interfaces

Si su clase está extendiendo un componente de Android como una Actividad o un Fragmento, es una buena práctica ordenar los métodos de anulación para que coincidan con el ciclo de vida del componente. Por ejemplo, si tiene una Actividad que implementa onCreate (), onDestroy (), onPause () y onResume (), el orden correcto es:

public class MainActivity extends Activity {  
  
 @Override  
 public void onCreate() {}  
  
 @Override  
 public void onResume() {}  
  
 @Override  
 public void onPause() {}  
  
 @Override  
 public void onDestroy() {}  
  
}

**Constantes de texto, nombres y valores**

Muchos elementos del SDK de Android, como SharedPreferences, Bundle o Intent, utilizan un enfoque par de clave-valor, por lo que es muy probable que incluso para una aplicación pequeña tenga que escribir muchas constantes de cadena.

Al usar uno de estos componentes, debe definir las claves como campos finales estáticos y deben tener el prefijo como se indica a continuación.

|  |  |
| --- | --- |
| Elemento | Prefijo del campo |
| SharedPreferences | SHARED\_PREF |
| Bundle | FAVORITE\_COUNT |
| Fragment Arguments | FAVORITE\_ID |
| Intent Extra | WEBSERVICE\_URL |

Ejemplo:

// Tenga en cuenta que el nombre del campo es el mismo que el nombre para evitar problemas de duplicación  
**public static final** String ***SHARED\_PREF*** = **"shared\_pref"**;  
**public static final** String ***FAVORITE\_COUNT*** = **"favorite\_count"**;  
**public static final** String ***FAVORITE\_ID*** = **"favorite\_id"**;  
**public static final** String ***WEBSERVICES\_URL*** = **"http:/..."**;

### 5.3.3 Estándar XML

**Nombramiento de recursos**

Los identificadores y los nombres de los recursos se escriben en minuscula\_guionbajo.

**Nombramiento de ID’s**

Las ID deben tener un prefijo con el nombre del elemento en minúscula y guión bajo.

|  |  |
| --- | --- |
| **Elemento** | **Prefijo** |
| TextView | text\_ |
| ImageView | image\_ |
| Button | button\_ |
| Menu | menu\_ |

Ejemplo ImageView:

<ImageView

android:id="**@+id/image\_profile**"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content" />

Ejemplo Menu:

<menu>  
 <item  
 android:id="**@+id/menu\_done**"  
 android:title="Done" />  
</menu>