**POLÍTICAS DE DESARROLLO SOFTWARE**

COMISIÓN PERMANENTE

* QA

**Comisión:**

Patricio Michael Paccha Angamarca

Tabla de contenido

[ESTÁNDARES DE DESARROLLO DE SOFTWARE 2](#_Toc146706055)

[Propósito 3](#_Toc146706056)

[Destinatario 3](#_Toc146706057)

[Buenas Prácticas 3](#_Toc146706058)

[Idioma 4](#_Toc146706059)

[Sangría y Formateo de Código 4](#_Toc146706060)

[Documentación 4](#_Toc146706061)

[Análisis de código 5](#_Toc146706062)

[Impresión innecesaria de excepciones 6](#_Toc146706063)

[Manejo de expresiones lógicas con operador lógico “?” 6](#_Toc146706064)

[Uso de Paréntesis 6](#_Toc146706065)

[Variables 6](#_Toc146706066)

[Constantes 7](#_Toc146706067)

[Asociaciones 7](#_Toc146706068)

[Excepciones 7](#_Toc146706069)

[Declaraciones 8](#_Toc146706070)

[Accesibilidad Web 8](#_Toc146706071)

[Resumen 9](#_Toc146706072)

[ESTÁNDARES DE DESARROLLO DE BASE DE DATOS 10](#_Toc146706073)

[Generalidades 12](#_Toc146706074)

[NOMENCLATURA 13](#_Toc146706075)

[Reglas 13](#_Toc146706076)

[Instancia 13](#_Toc146706077)

[Base de Datos 13](#_Toc146706078)

[Esquema 14](#_Toc146706079)

[Tablas 14](#_Toc146706080)

[Columnas 14](#_Toc146706081)

[Vistas 15](#_Toc146706082)

[Procedimientos almacenados 15](#_Toc146706083)

[Funciones 16](#_Toc146706084)

[Triggers 16](#_Toc146706085)

[Índices 17](#_Toc146706086)

[Sinónimo 17](#_Toc146706087)

[Secuencia 18](#_Toc146706088)

[Primary Key Constraint 18](#_Toc146706089)

[Unique Constraint 18](#_Toc146706090)

[Foreign Keys Constraint 18](#_Toc146706091)

[Defaults constraint 19](#_Toc146706092)

[Check Constraint Constraint 19](#_Toc146706093)

[IDENTIFICADORES 19](#_Toc146706094)

[Parámetros 20](#_Toc146706095)

[Variables 20](#_Toc146706096)

[DOCUMENTACIÓN 21](#_Toc146706097)

[Encabezado: 21](#_Toc146706098)

[Nomenclatura de Scripts de bases 22](#_Toc146706099)

[Snippet : Log de ejecución de script 23](#_Toc146706100)

[BUENAS PRÁCTICAS 24](#_Toc146706101)

[Variables columnas y campos 24](#_Toc146706102)

[Claves primarias 24](#_Toc146706103)

[Palabras reservadas 24](#_Toc146706104)

[Sangría 24](#_Toc146706105)

[Uso de espacios en blanco 24](#_Toc146706106)

[Comentarios de código 24](#_Toc146706107)

[Diagrama de entidad relación y diccionario de datos 25](#_Toc146706108)

[Uso de NOCOUNT 25](#_Toc146706109)

[Uso de sentencias SQL en el cliente 25](#_Toc146706110)

[Optimización de consultas 25](#_Toc146706111)

[Uso de Triggers 25](#_Toc146706112)

[Store Procedures 25](#_Toc146706113)

[Uso alias en las consultas 25](#_Toc146706114)

[Prohibición de uso select \* 26](#_Toc146706115)

[Recomendaciones Adicionales 26](#_Toc146706116)

[AUDITORÍA DE LA BASE DE DATOS 27](#_Toc146706117)

[Nivel de Servidor 27](#_Toc146706118)

[Nivel de Base de Datos 27](#_Toc146706119)

[MANEJO DE ERRORES 28](#_Toc146706120)

[Excepciones en Procedimientos Almacenados 28](#_Toc146706121)

[Registro de Log de Auditoría 29](#_Toc146706122)

[REFERENCIAS 29](#_Toc146706123)

[USER EXPERIENCE AND USER INTERFACE 30](#_Toc146706124)

[OBJETIVO 31](#_Toc146706125)

[ÁMBITO 31](#_Toc146706126)

[ANTECEDENTE 31](#_Toc146706127)

[INTRODUCCIÓN 31](#_Toc146706128)

[GUIDELINES 33](#_Toc146706129)

[Material Desing 33](#_Toc146706130)

[Bold, graphic, intentional 34](#_Toc146706131)

[Style 34](#_Toc146706132)

[Layout 34](#_Toc146706133)

[Usability 35](#_Toc146706134)

[UX & UI & Material Desing 35](#_Toc146706135)

[Referencias 35](#_Toc146706136)

[MANUAL DE INSTALACIÓN, CONFIGURACIÓN Y PARAMETRIZACIÓN 36](#_Toc146706137)

[OBJETIVO 37](#_Toc146706138)

[ÁMBITO 37](#_Toc146706139)

[MÓDULO O FUNCIONALIDAD 37](#_Toc146706140)

[INSTALACIÓN 37](#_Toc146706141)

[CONFIGURACIÓN 38](#_Toc146706142)

[PARAMETRIZACIÓN 38](#_Toc146706143)

[ANEXOS: 40](#_Toc146706144)

[Metodología de desarrollo y artefactos 42](#_Toc146706145)

[Estándares de desarrollo 42](#_Toc146706146)

[Artefactos: 42](#_Toc146706147)

[Plataforma 42](#_Toc146706148)

1. Control del documento
   1. **Documentos de referencia**
   * Registro Oficial No. 87 Normas de Control Intemo
   * Proceso de Gestión de la Demanda de Sistemas de Información
   * Desarrollo / Mantenimiento de Sistemas de Información
   * Plan Estratégico de Tecnologías de la Información: revisión
   1. **Revisiones**
   2. **Aprobaciones**
2. Definición de términos

# ESTÁNDARES DE DESARROLLO DE SOFTWARE

ESTÁNDARES DE DESARROLLO DE SOFTWARE

## Estándares

[**https://www.oracle.com/technetwork/java/codeconventions-150003.pdf**](https://www.oracle.com/technetwork/java/codeconventions-150003.pdf)

## Propósito

Describir un conjunto de normas básicas a seguir por parte del equipo de desarrollo de software al momento de codificar una aplicación para la Faculta de Ingeniería de Sistemas (FIS) con la finalidad de facilitar, entender, mantener y mejorar el código[[1]](#footnote-2)

## Buenas Prácticas

* Uso de profile (inspeccionado de código) según acorde al lenguaje de programación
* El nombre a de la entidades y atributos deben ser descriptivos evitando abreviaciones
* Las entidades deben estar nombradas en singular, haciendo el uso de UpperCamelCase
* Los atributos de las entidades deben estar nombrados con el uso de lowerCamelCase.

**Ejemplo:**

Entidad => MovimientoIncidente

Atributo => razonIncidente

* Las entidades deben implementar Serializable.
* Las interfaces deben llevar el prefijo I
* Si la entidad pertenece o está contenida por capas arquitectónicas se debe nombrar con el nombre de la entidad y el sufijo de la capa, y una clase la cual implementa dicha interfaz con el nombre de la interfaz más el sufijo Impl.

**Ejemplo:**

Intefaz => IEmpleado

Servicio Interfaz => IServicioSorteoConfiguracion

Servicio => UsuarioDAO (Usuario pertenece a la capa Data Access Object - DAO)

Implementación => UsuarioDAOImpl

* Crear agrupamiento semántico si las entidades mantienen relación.

**Ejemplo:**

Entidad => Usuario (entidad padre)

Entidad => UsuarioPago (entidad hijo, subclase, relación)

* Usar tipos de datos objetos, esto en referencia a las primitivas.

**Ejemplo:**

int (NO)

Integer (**SI**)

### Idioma

* Se usará el idioma español para nombrar clases, interfaces, variables. Excepto para los beans tales como getters y setters.

### Sangría y Formateo de Código

* Archivos de código deben estar en formato UTF-8
* Se requiere que para toda clase e interfaz se deja sangrías que faciliten la lectura de código.
* El código debe ser formateado

### Documentación

* Siempre se escribe en tercera persona.
* Los caracteres especiales tales como tildes y eñes se han de codificar con su código HTML correspondiente.
* Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de cada método.
* Se tienen que evitar el uso de caracteres especiales dentro de los comentarios, así como el uso de cajas u otro tipo de gráfico creado mediante códigos ASCII.
* Se deberá siempre comentar las interfaces y funciones; así como las declaraciones de funciones y funciones de las interfaces y clases respectivamente.
* Documentar las clases e interfaces con el autor(@author) y versión(@version)
* Existen 2 tipos de comentarios: Implementación (/\*...\*/ y //) y Documentación (/\*\*...\*/). Solo los Comentarios de Documentación son mostrados por JavaDoc.
* Se usará el siguiente formato para la generación de documentación de cada método.

**/\*\***

\* Indica si la pila está vacia

\* @param nombreParametro

\* @return true indica pila vacía

**\*/**

* Se recomienda borrar líneas de código que estén comentadas y no tenga utilidad.
* No se aceptarán impresión de mensajes a consola System.out.println
* Se exigirá la documentación del código fuente en clases e interfaces apegado a las siguientes señalizaciones:
* Tag de documentación @author
* Tag de documentación @date
* Tag de documentación @parameters
* Tag de documentación @return
* Encabezado de cada archivo de código.

**Ejemplo:**

/\*\*

 \* ©2024, All Rights Reserved.

 \* (EPN) ESCUELA POLITÉCNICA NACIONAL

\* (FIS) FACULTA DE INGENIERÍA EN INFORMÁTICA

 \*

\* @version:

 \* @author :

 \* @date   : año.mes.dia

 \*\*/

### Análisis de código

* Se debe utilizar PMD (calidad de código) y CPD (código duplicado) para la evaluación de la calidad de código. No se debe tomar en cuenta el “Avoid modifiers which are implied by the context” cuando PMD alerte sobre algún error en la programación.
* El código deberá ser validado con pruebas de carga que el código implementado tenga un rendimiento óptimo.
* Se recomienda no poseer clases de más de 2000 líneas[[2]](#footnote-3).

### Impresión innecesaria de excepciones

* Cuando se trate de excepciones relacionadas al manejo y consumo de servicios web de terceros así como conexiones con Alfresco se deberá atrapar dicha excepción e imprimir únicamente el e.getMessage() más un mensaje simple y descriptivo que indique el nombre de la clase y la función donde se suscitó la excepción.

**Ejemplo**:

log.error(this.getClass().getCanonicalName() + " -> " + mensajeDescriptivo + " -> nombreFuncion() -> [" + e.getMessage() + "]");

* Para todas las demás excepciones genéricas se imprimirá e, e.printStackTrace().

**Ejemplo:**

log.error("Error al buscar tipos de acción usadas en recusación de demandas", e);

### Manejo de expresiones lógicas con operador lógico “?”

* Si una expresión contiene operador lógico “?” (operador ternario), la expresión deberá ser precedida y finalizada por paréntesis.

**Ejemplo**

(x >= 0) ? x : -x;

return (condition ? x : y);

### Uso de Paréntesis

* De acuerdo con la complejidad de una expresión se debe usar paréntesis para facilitar la lectura del código en especial cuando se usa varios operadores y así entender claramente la precedencia.

**Ejemplo**

if (a == b && c == d || e != f && h == i) (NO)

if ((a == b && c == d) ||( e != f && h == i)) (SI)

### Variables

* Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para las de los métodos. No se permite el uso de caracteres especiales; tanto para clases, funciones y variables. Se usará en lo posible nombres completos y descriptivos con significado funcional, las excepciones a ésta regla son las variables utilizadas en bucles for(en donde se necesite conocer la posición), para esos casos se permite utilizar i, j, k, l y siempre en ese orden de anidamiento. Para el manejo de listas de objetos se debe usar for each.

**Ejemplo**

diaCalculo, fechaIncoporacion (SI)

dC, DCal, fI, FI… (NO)

### Constantes

* Para representar constantes se usará variables de tipo static. Los nombres de constantes de clases deberían escribirse todo en mayúsculas con las palabras separadas por guion bajo (“\_”). Todas serán declaradas como public static final.

**Ejemplo**

public static final String PROPERTY\_URL\_SERVICIO = "urlServicio";

* Por requerimiento del *framework Angular* las constantes no podrán ser estáticas por lo que para este caso habrá dicha excepción[[3]](#footnote-4).

**Ejemplo:**

public String OCULTO = "H";

* Se usará el patrón singleton para la creación de instancias de constantes

**Ejemplo:**

public class Constantes{

private static Constantes instance;

public static Constantes getInstance() {

if (instance == null) {

instance = new Constantes();

}

return instance;

}

}

### Asociaciones

* Se debe usar fetch para las asociaciones de tipo lazy.

**Ejemplo:**

@ManyToOne(fetch=FetchType.LAZY)

### Excepciones

* Se debe atrapar en los DAOS al menos las excepciones NoResultException, NonUniqueResultException
* En la inserción de entidades se deberá atrapar la siguiente excepción EntityExistsException
* Se utilizará log4j de Apache. No es recomendado las librerías slf4j ni logging.

**Ejemplo**

private final Logger log = Logger.getLogger(this.getClass());

### Declaraciones

* Para la declaración de las variables no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean.
* Las declaraciones de variables se realizarán al inicio de la función o clase según corresponda. Serán inicializadas en null y cuando se necesite utilizar, se realizará la instanciación.
* La declaración de las variables locales a una clase, método o bloque de código se realizan al principio de este y no justo antes de necesitarse la utilización de la variable, la única excepción a esta regla son las variables que gestionan los bucles FOR, las variables de avance de bucles FOR no podrán ser codificadas de ninguna manera fuera de la propia sentencia del bucle.
* La duplicidad de los nombres de variables en diferentes niveles dentro de la misma clase se tiene que evitar.

### Accesibilidad Web

* En las xhtmls se deberá setear el lenguaje. lang="es" en el tag html.
* La página deberá tener el tag title con una referencia descriptiva del propósito de la página.
* Toda imagen <h:graphicImage debe tener el atributo ALT y con mensaje. No se debe dejar en blanco.
* En los eventos JS siempre deberán llamar a una función donde se trate la información. No ingresar líneas de código JS en el evento.
* Siempre en cada <p:inputText <h:outputText se debe usar el atributo title.
* <title> no puede estar vacío.
* Usar el templates para que la aplicación sea responsive
* Cuando pulse la tecla TAB para navegar por los componentes que requieren de interacción con el usuario; debe poseer una secuencia lógica.

## Resumen

Se utilizará como estándares de desarrollo Java Code Convention[[4]](#footnote-5). A continuación, se muestra ejemplos.

|  |  |  |
| --- | --- | --- |
| **No.** | **Módulos** | **Plataforma** |
| Paquetes | Los paquetes serán únicos, de alto nivel y escritos en minúsculas identificando la funcionalidad del paquete tal como “dao”, ”model”, “servicio”, “util” etc. Esta convención deberá especificar el nombre del proyecto. En este caso “esatje”. El paquete que implemente una interfaz tendrá el sufijo Impl. | ec.gob.epn.app.rest  ec.gob.epn.app.rest.impl |
| Clases | Los nombres de las clases serán sustantivos con notación Cammel Case. Deberán ser simples y descriptivos. Evitar usar acrónimos y abreviaciones. Usar nombres en singular. Las clases que implementen una interfaz deberán poseer el sufijo Impl. | class Abogado;  class AbogadoDAO;  class AbogadoRestfulImpl; |
| Interfaces | Las interfaces deben poseer nombres con notación Cammel Case. Las interfaces de los servicios rest deben tener el sufijo “Restful”. Usar nombres en singular. | public interface AbogadoRestful |
| Métodos | Los métodos deben empezar con un verbo en infinitivo. Se usará Cammel Case para las palabras compuestas. La primera letra será minúscula. | listarMateria(); |
| Variables | Se usará notación Cammel Case y se usará los tipos de datos como objetos y no como tipos de datos primitivos. No se permite el uso “\_” para palabras compuestas, excepto para las constantes | int //NO  Integer //SI  long //NO  Long //SI |
| Constantes | Se usará la clase de Constantes del paquete ec.gob.epn.app.util, para definir constantes necesarias de los diferentes módulos. Podrían contener “\_” para palabras compuestas y deberán ser mayúsculas. | public static final String ESTADO\_ELIMINADO = "ELIMINADO"; |
| Reportes | Para la reportería se usará IReport junto con el jar template.jar. el nombre del reporte deberá empezar con la iniciales rpt  En caso de que se use iText se utilizará los estándares anteriormente descritos | rptActaPorJudicatura.jrxml |

# ESTÁNDARES DE DESARROLLO DE BASE DE DATOS

Versión 1.0

**CONTROL DE CAMBIOS:**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Autor** | **Referencia del cambio/revisión** |
| 01.ago.2023 | Patricio Paccha | Borrador |
|  | Carlos Anchundia | Borrador |
|  | Monserrate | Revisión |
|  |  |  |

**APROBACION:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisado por** | **Observaciones** | **Sumilla** |
|  |  |  |  |
|  |  |  |  |

ESTÁNDARES PARA EL DESARROLLO DE BASE DE DATOS

El propósito del presente documento es el de definir un patrón de normalización que deberá ser utilizado al momento de crear y actualizar objetos de base de datos (tablas, triggers, procedimientos almacenados, funciones, etc.). De este modo, se fomenta la facilidad de mantenimiento y la estandarización de elementos en la base de datos que se maneja en la institución enmarcada a los lineamientos generales de la Arquitectura General de TICs[1].

Microsoft recomienda el uso de la ISO 8859-1 para evitar los problemas de conversión de caracteres entre los sistemas; además contiene caracteres adecuados para manejar la mayor parte de idiomas que comprende América y la parte Oeste de Europa, incluso haciendo uso, de ser necesario, de tipos Unicode en algunos campos tales como nchar, nvarchar y ntext.[2].

## Generalidades

* Normaliza las tablas para asegurar que no exista duplicidad de datos y se aproveche al máximo el almacenamiento en las tablas.
* Usar campos de tipo bit para almacenar valores booleanos envés de enteros o varchar puesto que consume almacenamiento innecesariamente.
* usar tipos Unicode. Use siempre: nchar, nvarchar y ntext
* Definir filegroup específicos o particiones para entidades con almacenamiento extenso
* Imágenes y columnas de tipo blob no deben estar definidas en tablas frecuentemente requeridas para evitar problemas de rendimiento. Estos datos deben ser definidos en tablas separadas relacionadas por un identificador.

## NOMENCLATURA

## Reglas

* Usar nombres en singular para todos los objetos a crear
* Se debe declarar los nombres en español
* No usar palabras reservadas en los nombres de los objetos, en el caso de ser absolutamente necesario usar [], por ejemplo [Year]
* Los nombres de los objetos de base de datos utilizarán el **<sufijo>** (en mayúsculas) y estarán compuestos por palabras con la primera letra en mayúscula (**UpperCamelCase**). Se debe evitar el uso de abreviaciones que dificulten la comprensión.
* Si se requiere separar el nombrado de los objetos debe usar el underscore: “\_”
* Las palabras reservadas del lenguaje TRANSACT – SQL deben estar en mayúsculas.

### Instancia

**Reglas:**

Debe estar relacionado con el aplicativo en mayúsculas.

**Formato:**

<APLICATIVO>

**Ejemplo:**

APP\_SATJEU07

### Base de Datos

**Reglas:**

El nombre en mayúsculas y separado por underscore.

El nombre físico de la base de datos -Master Database Files y Log Database Files- debe corresponder al alias de la aplicación seguido de guion bajo más las palabras Data y Log respectivamente.

**Formato:**

BDD:<APLICATIVO>

MDF: <ALIAS>\_Data

LDF: <ALIAS>\_Log

**Ejemplo:**

BDD: EXPEDIENTE\_ELECTRONICO

MDF: EEL\_Data

LDF: EEL\_Log.

### 

### Esquema

**Reglas:**

Nombre en acrónimo o nemónico.

**Formato:**

<ESQUEMA>

**Ejemplo:**

ADM

### Tablas

**Reglas:**

Nombre en singular y **UpperCammelCase**.

Si la tabla es relacional considere que primero es el nombre de la tabla principal.

**Formato:**

<TableName>

**Ejemplo:**

CategoriaJudicatura

### Columnas

**Reglas:**

Nombre en singular, **UppercammelCase** y no usar espacios, ni guiones.

Para las claves primarias se debe usar el prefijo **Id** unido al <NombreTabla>

No sobrecargar la descripción. Es decir, si estamos en una tabla Provincia y necesitamos la columna nombre; debemos usar el campo nombre y no nombreProvincia.

Las columnas que sirven como referencia como un Foreign Key deberán ser nombradas como esta en la Tabla que hace referencia.

Ajustar al máximo el tamaño de los campos para no desperdiciar espacio.

**Formato:**

<ColumnName>

**Ejemplo:**

IdProvincia à primary Key

IdUbicacion à campo de la tabla Provincia

IdPais à en la tabla Provincia hace referencia a Id de la tabla País

### Vistas

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **VS\_**

**Formato:**

VS\_<NombreVista>

**Ejemplo:**

VS\_BuscarJuiciosReasignacion

### Procedimientos almacenados

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **PA\_** más el **acrónimo del aplicativo (3 letras mayúsculas)** más underscore el ‘nombre del procedimiento almacenado’

El ’nombre del procedimiento almacenado’ debe ser empezar con un verbo en infinitivo.

Los parámetros inician con i (entrada)/ o (salida) junto Nombre en singular, **UpperCammelCase**

**Formato:**

PA\_<ACRONIMO >\_<VerboNombreProcedimiento >

**Ejemplo:**

PA\_EEL\_ **Resortear**Firmante

PA\_EEL\_ **Obtener**Firmante

**Notas:**

Los nombres de los Stored Procedures NO deben comenzar con SP\_, esto porque generalmente el SQL piensa que son system procedures y los busca primero en la base de datos Master

Se recomienda usar procedimientos almacenados cuando sea estrictamente necesario. Se debe desarrollar toda la lógica de negocio en clases **java**, en los paquetes de servicio[3].

No usar prefijo xp\_, para extended stored procedures, cual es un prefijo reservado para identificar system extended stored procedures. No se permiten sentencias SQL en el cliente. Esto genera baja en el performance de los sistemas.

Evitar en lo posible usar cursores, dado que éstos consumen muchos recursos

## Funciones

**Reglas:**

Nombre en singular, **UpperCammelCase** sin espacios iniciando con el prefijo **FU\_** más underscore el ‘NombreFunción’

El ‘NombreFunción’ debe ser empezar con un verbo en infinitivo.

Los parámetros inician con i (entrada)/ o (salida) junto Nombre en singular, **UpperCammelCase**

**Formato:**

FU\_< VerboNombreFunción>

**Ejemplo:**

FU\_ConvertirFirmantePonente

**Notas:**

Se recomienda usar funciones cuando sea estrictamente necesario. Se debe desarrollar toda la lógica de negocio en clases **java**, en los paquetes de servicio[3].

## Triggers

**Reglas:**

Usar el prefijo **TR\_** más **el nombre de la tabla** seguido de guion bajo y a continuación la **acción** (Insert, Delete, Update) y nombre en singular, **UpperCammelCase**

**Formato:**

TR\_<TableName>\_<ActionNombreDescriptivo>

**Ejemplo:**

TR\_Judicatura\_UpdateDeleteAuditado

**Notas:**

<Action> : Debe especificar la acción principal que desarrolla

Insert

Update

Delete

## Índices

**Reglas:**

Usar el prefijo **IX\_** más el nombre de la tabla seguido de guion bajo y a continuación el nombre del campo afectado

**Formato:**

IX\_<TableName>\_<Column>

**Ejemplo:**

IX\_MovimientoJuicio\_IdJudicaturaDestino

**Notas:**

Por cada tabla transaccional importante no exceder más de 12 índices.

Requerimientos de reportes, consultas o inserciones deberá utilizar los índices existentes.

## Sinónimo

**Reglas:**

Iniciará con SYN\_ más el nombre de la tabla o seudónimo más el nombre en singular, **UpperCammelCase**

**Formato:**

SYN\_<TableName>

**Ejemplo:**

SYN\_Provincia

**Nota:**

Se utilizará sinónimos sólo para los objetos de los que no seamos propietarios; es decir a las tablas a las cuales solo se tiene permiso de lectura.

## Secuencia

**Reglas:**

Tendrán como prefijo la palabra “**SEQ**” seguido de underscore más el nombre en singular, **UpperCammelCase**

**Formato:**

SEQ\_<TableName>

**Ejemplo:**

SEQ\_Provincia

## Primary Key Constraint

**Reglas:**

Usar el prefijo PK\_ mas el nombre de la columna.

**Formato:**

PK\_<NombreColumna>

**Ejemplo:**

PK\_IdProvincia

## Unique Constraint

**Reglas:**

Usar el prefijo UK\_ seguido por el nombre del campo. En caso de tener varios campos usar **UpperCammelCase** para denotar cada uno de ellos.

**Formato:**

UK\_<NombreColumnas>

**Ejemplo:**

UK\_LoginCedula

## Foreign Keys Constraint

**Reglas:**

Usar el prefijo FK\_ seguido por el nombre de campo. En caso de tener varios campos usar **UpperCammelCase** para denotar cada uno de ellos.

**Formato:**

FK\_<NombreColumna>

**Ejemplo:**

FK\_IdProvincia

## Defaults constraint

**Reglas:**

Usar el prefijo DF\_ seguido por el nombre de la tabla más underscore con el nombre del campo seguido de guión bajo con el valor a ser asignado.

**Formato:**

DF\_<TableName>\_<ColumnName>\_<Value>

**Ejemplo:**

DF\_Judicatura\_Nombre\_Nulos

## Check Constraint Constraint

**Reglas:**

Usar el prefijo CK\_ seguido por el nombre de la tabla más underscore con el nombre del campo seguido de guión bajo con el valor a ser evaluado.

**Formato:**

CK\_<TableName>\_<ColumnName>\_<Value>

**Ejemplo:**

CK\_MovimientoJuicio\_Cantidad\_NoNegativo

### IDENTIFICADORES

Todos los nombres de identificadores de variables, constantes, cursores y demás, deben ser escritos usando **LowerCammelCase**

**Formato**:

<nombreParámetro>

**Ejemplo:**

cedulaEcuatoriana.

## Parámetros

**Reglas:**

Para un mejor entendimiento de la lógica de los programas, los parámetros deberán llevar la sintaxis **prefijo** + **UpperCammelCase**

**Formato**:

<tipodato><NombreParámetro>

**Ejemplo:**

iCedulaEcuatoriana. à parámetro de entrada

**Nota:**

|  |  |
| --- | --- |
| **Tipo de Parámetro** | **Identificador** |
| Input/ Entrada | i |
| Output/salida | o |
| Input Output | io |

**Ejemplo:**

CREATE PROCEDURE PA\_EEL\_Resortear (

@iCedula VARCHAR(10) INTPUT,

@oSecretario INT OUTPUT )

AS

BEGIN

<Declaración de variables>

<Sentencias SQL>

END

## Variables

**Reglas:**

La declaración de variables deberá llevar la sintaxis **lowerCammelCase**

**Formato:**

<nombreParámetro>

**Ejemplo:**

Declaración de una variable para manejar el número de autorización de SRI de tipo numérico.

Declare NumeroAutorizacion integer;

## DOCUMENTACIÓN

## Encabezado:

* Los scripts deben tener encabezado
* Los procedimientos y funciones deben tener encabezado

**Ejemplo:**

/\*

 \* ©2024, All Rights Reserved.

 \* (EPN) ESCUELA POLITÉCNICA NACIONAL

\* (FIS) FACULTA DE INGENIERÍA EN INFORMÁTICA

\*

\* Este código está protegido por las leyes y tratados internacionales de derechos de

\* autor por lo que el uso, reproducción o distribución del código total o parcial no

\* autorizada se atendrá a las sanciones contempladas en la ley con su máximo rigor.

\*

\* [App] : Auditoria-Alert, App2, …

\* [Developer]

\* Author :

\* Descripction :

\*

\* [SQM-Tester]

\* Author :

\* Date :

\* Observation :

\* Tested : [ ] Stress o Stress concurrente [ ] ETL

\* [ ] DataBase Object [ ] Log & Audit

\* [ ] Quality Data

\*/

--

-- do code

--

## Nomenclatura de Scripts de bases

Para que un script de bases de datos sea válido, debe cumplir con las siguientes recomendaciones:

1. Los scripts de modificación de vistas, procedimientos y funciones se crean en un solo directorio “\_DATABASE” sin numeración y con la siguiente nomenclatura:

**FIS-<DATABASE>-<SHEMMA>**-<Vista, función, procedimiento>.sql”

**Ejemplos:**

FIS-BADALEX1-CLEX1-vs\_CN\_SorteoFirmantes.sql

2. El script debe ser enviado en un archivo con extensión “.sql”

3. Los nombres de los scripts deben tener la siguiente sintaxis:

**Sintaxis:**

##-**FIS**-BaseDeDatos-SHEMMA-ModuloProposito-AÑO-MES-DIA.sql

**Ejemplos:**

05-FIS-Badalex-ReportesInternosSPIngresoCausas-2016-JUN-17.sql

06-FIS-Adm-ArchivoCrearRolesParaArchivo-2016-AGO-28.sql

07-FIS-Adm-TramiteAlterTableJuicios-2016-SEP-15.sql

4. Mantener la secuencia del número de script y las fechas no son relevantes

5. Verificar que el valor de la variable @Filename definida en el script, debe ser idéntico al nombre físico del archivo.

**Ejemplo:**

exec sp\_AddLogScript '@Filename', '03-FIS-Badalex-Archivo-2016-SEP-07.sql'

6. Para almacenar evidencia que el script haya sido ejecutado especificadamente en la creación de objetos de base de datos deben utilizarse las sentencias:

**Ejemplo:**

exec sp\_AddLogScript ‘acción realizada sobre el objeto’

Esto no es necesario dentro de objetos de base de datos como sp, vista y funciones

## Snippet : Log de ejecución de script

Considerar la que el script tenga log de ejecución, el manejo de errores y transacciones.

***Ejemplo:***

-- do code à procedure, function, ...

BEGIN TRAN

BEGIN TRY

-- Body script …

--

COMMIT TRAN

END TRY

BEGIN CATCH

ROLLBACK TRAN

END CATCH

go

## BUENAS PRÁCTICAS

## Variables columnas y campos

usar tipos Unicode. Use siempre: nchar, nvarchar y ntext

## Claves primarias

No se permite el uso de claves primarias compuestas. Se recomienda usar secuenciales.

Elegir columnas con tipos enteros (o sus variantes) para indexar. Una columna con tipo varchar puede causar problemas de rendimiento.

## Palabras reservadas

Todas las palabras reservadas deberán escritas ser en mayúsculas.

## Sangría

Las instrucciones serán alineadas de acuerdo con la profundidad los diferentes bloques que se vayan construyendo. Se puede utilizar alguna herramienta para dar formato al código.

## Uso de espacios en blanco

Siempre incluya un espacio entre cada identificador y separador.

**Ejemplo:**

En lugar de esto:

WHILE (totalSales < maximumSales AND companyType='NEW') LOOP

Escriba esto:

WHILE (totalSales < maximumSales AND companType = 'NEW') LOOP

## Comentarios de código

Se debe tomar en cuenta las siguientes consideraciones para comentar el código TRANSACT SQL.

* No se debe comentar cada línea del código
* Comentar solamente las partes importantes del código.
* Los comentarios deben ser completos y coherentes
* Los comentarios deben ser escritos usando la sintaxis, gramática, puntuación y ortografía correcta.
* No use un comentario para reafirmar las acciones de una pieza de código.

El código SQL podría comentarse adicionalmente en las siguientes instancias:

* Antes de cada estructura de loop
* Antes de cada bloque BEGIN…END
* Antes de cada expresión lógica condicional (IF <condición> THEN).
* Antes de cualquier otra sentencia que se considere significativa e importante

## Diagrama de entidad relación y diccionario de datos

Generar para la BDD un Diagrama ER y un Diccionario de Datos

Script que crean objetos de base de datos deben ser documentados

## Uso de NOCOUNT

En los Store Procedures se debe tener en cuenta que SET NOCOUNT ON elimina la notificación del nro. de registros afectados por cada sentencia SQL lo cual incrementa el performance. Esto exceptúa los procedimientos que, por la naturaleza de su función, requieran esta funcionalidad.

## Uso de sentencias SQL en el cliente

No se deberán utilizar sentencias SQL en el cliente. Esto genera una gran baja en la performance de los sistemas.

## Optimización de consultas

* Se procurará elegir en la cláusula WHERE aquellos campos que formen parte de la clave primaria.
* Si se desea consultar por campos pertenecientes a índices compuestos es mejor utilizar todos los campos que componen el índice.
* Cuando se utilizan varias tablas dentro de la consulta hay que tener cuidado con el orden empleado en la cláusula FROM. Si deseamos saber cuántos alumnos se matricularon en el año 1996 y escribimos: FROM Alumnos, Matriculas WHERE Alumno.IdAlumno = Matriculas.IdAlumno AND Matriculas.Año = 1996 el gestor recorrerá todos los alumnos para buscar sus matriculas y devolver las correspondientes. Si escribimos FROM Matriculas, Alumnos WHERE Matriculas.Año = 1996 AND Matriculas.IdAlumno = Alumnos.IdAlumnos, el gestor filtra las matrículas y después selecciona los alumnos, de esta forma tiene que recorrer menos registros.

## Uso de Triggers

Los triggers deberán ser utilizados solo en caso de que se quiera hacer alguna acción en cascada sobre la integridad referencial o cuando se quiera evitar que una clave primaria sea modificada o si la funcionalidad del aplicativo así lo requiere, tomando en cuenta que el uso de estos degrada según sea el caso la velocidad de actualización de las tablas.

## Store Procedures

En la programación de Store Procedures se debe utilizar mayúsculas para las sentencias propias y palabras reservadas del SQL.

## Uso alias en las consultas

* Si se utiliza varias tablas en la consulta siempre se debe especificar a qué tabla pertenece cada campo, con ello se ahorra tiempo al gestor en localizar a la tabla pertenece el campo.

SELECT distinct ofe.ofe\_id,

ofe.pac\_id,

ofe.ofe\_fecha\_desde,

ofe.ofe\_fecha\_hasta,

ofe.ofe\_estado,

ofe.ofe\_version\_registro,

cen.pra\_id

FROM tsga\_oferta ofe,

tsga\_oferta\_centro\_ejecucion oce,

tsga\_centro\_stu\_ejecucion cen

WHERE ( (ofe.ofe\_id = oce.ofe\_id)

AND (cen.cee\_id = oce.cee\_id)

AND (ofe.ofe\_id = en\_id))

En el caso de varias referencias a una misma tabla agregar un número secuencial luego del prefijo de la tabla, esto aplica también para subconsultas o selects anidados.

## Prohibición de uso select \*

No utilizar nunca SELECT \* porque el gestor debe leer primero la estructura de la tabla antes de ejecutar la sentencia

## Recomendaciones Adicionales

Para que un script de bases de datos sea de calidad, debe cumplir con las siguientes recomendaciones:

1. El script debe contener las descripciones del caso al inicio de archivo, detallando si contiene varias acciones (creaciones, modificaciones de objetos de base de datos) o únicamente una descripción antes de la creación o modificación del objeto.
2. Tomar en consideración de que si se crea o modifica un objeto debe cerrarse con GO
3. En general para cambios de estructura, gestión de datos deben realizarse las verificaciones de caso pueden utilizar la cláusula IF EXISTS
4. En cambios de datos utilizar transacciones para asegurar que se cumpla lo indicado, así mismo tener en consideración que si la aplicación controla las transacciones ya no sería necesario en el gestor de base de datos.
5. En inserciones de datos en tablas deben especificar las columnas de las tablas para el mapeo el cual les ayudara a asegurar que los datos se registrar en la columna especificada (mapeo), en las inserciones masivas a partir de una selección deben contar con el nombre de columna correspondiente.
6. En transacción de datos deben asegurar que los datos que estén modificando o eliminando (cambio de estado) sea el correcto recomendado que utilicen la pk para su modificación.
7. Finalmente tener presente que los scripts que se envié deben asegurar la información.

## AUDITORÍA DE LA BASE DE DATOS

SQL Server proporciona varias características útiles para auditar las actividades y los cambios en el sistema de SQL Server. Estas características permiten a los administradores implementar una estrategia de defensa optimizada para los riesgos de seguridad específicos de su entorno.

SQL Server permite crear auditorías a nivel de servidor o a nivel de base de datos. La configuración es sencilla ya que solo se requiere especificar donde se almacenará la auditoría (en un archivo, la bitácora de eventos de aplicaciones o la bitácora de eventos de seguridad), que objeto se desea auditar (Base de datos, Esquema, Tablas, Funciones, Procedimientos Almacenados o Vistas) y para cual Principal (entidades que pueden solicitar recursos de SQL Server). [4]

Se debe tomar en cuenta que el costo de la auditoría es la degradación del rendimiento de servidor de la BDD.

## 

## Nivel de Servidor

En las especificaciones de auditoría de servidor debemos indicar el objeto de auditoría que deseamos rastrear. Por ejemplo, en el caso de un objeto de auditoría que rastree los login fallidos, debemos crear una especificación que busque los logins que no son exitosos. Para más detalle revisar [4]

## 

## Nivel de Base de Datos

En este tipo se especifica la clase de objeto se utiliza para indicar lo que se quiere auditar. Las opciones son: Objeto, Base de datos o Esquema. Seleccione objeto para auditar Tablas, Funciones, Procedimientos Almacenados o Vistas. Para más detalle revisar [4]

## MANEJO DE ERRORES

## Excepciones en Procedimientos Almacenados

En caso de requerirlo, los procesos deberán levantar la excepción que será manejada dentro de un proceso principal a nivel e aplicación y no a nivel de base de datos, por lo tanto todos los procesos de consulta, actualización o inserción deben tener manejo de excepciones para que se pueda lanzar una excepción que pueda ser entendida por un usuario o desarrollador.

**Algoritmo de Excepción**

WHEN exception1 THEN

Executable\_statements1;

WHEN exception2 THEN

Executable\_statements2;

. . .

WHEN OTHERS THEN

Otherwise\_code;

END

Para el manejo de excepciones considerar:

BEGIN TRY

-- do something

--

-- RAISERROR ('Error raised in TRY block.', -- Message text.

16, -- Severity.

1 -- State.

); -- Forzar un salto

END TRY

BEGIN CATCH

-- ROLLBACK

-- considerer that you catch of error

-- ERROR\_MESSAGE(),

-- ERROR\_LINE()

-- ERROR\_SEVERITY(),

-- ERROR\_STATE();

-- RAISERROR (@ErrorMessage, -- Message text.

-- @ErrorSeverity, -- Severity.

-- @ErrorState -- State.

-- );

END CATCH

Considerar los niveles de anidamiento para la captura de errores con el respectivo retorno de mensajes del error para la captura o administración de este.

## Registro de Log de Auditoría

De ser necesario, se deberá establecer a nivel de proyecto, los registros de auditoría a nivel de tablas, para lo cual se deberá crear un paquete para centralizar estas funcionalidades.

## REFERENCIAS

[2] J. Gabillaud, SQL Server 2008: Administración de una base de datos con SQL Server Management Studio. Ediciones ENI, 2010.

[4] «SQL Server Audit (motor de base de datos)». [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/cc280386.aspx. [Accedido: 30-mar-2015].

# USER EXPERIENCE AND USER INTERFACE

Versión 1.0

**CONTROL DE CAMBIOS:**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Autor** | **Referencia del cambio/revisión** |
| 01.ago.2023 | Patricio Paccha | Borrador |
|  | Carlos Anchundia | Borrador |
|  | Monserrate | Revisión |
|  |  |  |

**APROBACION:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisado por** | **Observaciones** | **Sumilla** |
|  |  |  |  |
|  |  |  |  |

UX & UI

**USER EXPERIENCE AND USER INTERFACE**

## OBJETIVO

Crear un lenguaje visual estándar que sintetice los principios del diseño de interfaz gráfica innovadora y tecnológica para el Sistema de Tramite Automático Ecuatoriano – SATJE.

## ÁMBITO

* User Experiencia
* User Interface
* Material Desing

## ANTECEDENTE

La interfaz de usuario (**UI**) se conceptualiza como el entorno donde el usuario interactúa con el sistema, puede ver y tocar. En la actualidad los programas de software la interfaz de usuario suele ser grafica GUI accesibles mediante la web.

La Experiencia de Usuario (**UX**) por contrapartida es un concepto más intangible que acompaña, pero excede a la UI. La UX (acrónimo de Experiencia de Usuario en inglés) se emplea para describir aspectos dentro de un concepto controvertido descritas en la norma ISO 9241-210

## INTRODUCCIÓN

La arquitectura de Experiencia de Usuario describe las funciones de la interfaz de usuario y su usabilidad considerando la experiencia de una persona con un sistema incluyendo el diseño industrial, el diseño gráfico, la interfaz, la interacción física y la manual. En este contexto se refleja varias disciplinas del diseño UX contempla:

* Diseño visual
* Diseño de la interfaz
* Diseño de navegación
* El diseño de interacción (IxD)

Seleccionar un esquema que refleje elementos subjetivos para determinar el campo de trabajo la usabilidad que todo sistema debería contener se realiza considerando el diagrama del panal de experiencia de usuario de Peter Morville.

  
Fig. 1. Panal de la Experiencia de usuario de Peter Morville

Por lo tanto, en el presente el diseño de UI se enfoca en la funcionalidad requerida por los usuarios y asegurarse de que la interfaz tenga elementos que son de fácil acceso, se entienden y utilizan para facilitar esas acciones. Entonces, al diseñar la UX se busca el proceso de aumentar la satisfacción y lealtad del usuario mediante la mejora de la usabilidad, facilidad de uso, y el placer proporcionado en la interacción del usuario y el producto.

La Usabilidad juega un papel esencial en que las experiencias sean eficaces y agradables, pero existen factores como la Arquitectura de Información, el diseño de interacción, accesibilidad, diseño visual, etc. que también tienen su lugar. La usabilidad es un subconjunto de la UX muy cerca del centro donde estaría lo más esencial: la utilidad, pero con otros niveles por encima. En palabras de Lee McCormick: La experiencia de usuario es igual a un viaje en carretera. La Usabilidad es el trazado de la carretera y la señalización.

## GUIDELINES

La metáfora Material Design es el referente considerado como base para la implementación del lenguaje de interfaz gráfica open-source y proporciona un estándar probado e implementado en varios entornos para expresar la teoría unificadora de un espacio racionalizado y un sistema de movimiento.

Material design es un concepto, una filosofía, un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal. En un inicio creado por Google, enfocándolo en sistemas moviles, web y cualquier plataforma. El encargado de crear material design, y máximo responsable de diseño en Google a nivel de UI & UX.

Al incluir una nueva dimensión se hace más fácil para el usuario la comprensión de la jerarquía. De un primer vistazo es mucho más fácil de entender casi todos los detalles es fácilmente utilizable, independientemente del dispositivo para el que estemos diseñando. Las animaciones son un punto fuerte a usar para extrapolar a otros puntos del diseño porque vienen incorporadas y eliminan la necesidad de manipular manualmente.

Proporciona una experiencia unificada a través de todos los dispositivos que hace que, a la larga, más usuarios se sumarán a la facilidad de uso.

Finalmente, material design es un invento de Google, por lo que está intrínsecamente ligado a él y al estar tan claramente marcadas las directrices, el desarrollador no puede usar tanto la creatividad y se debe regir al uso y estándar definido.

## Material Desing [[5]](#footnote-6)

Material Design es un sistema unificado que combina teoría, recursos y herramientas para crear experiencias digitales basado en la realidad táctil, inspirado en el estudio del papel y la tinta, pero tecnológicamente avanzado y abierto a la imaginación y la magia.

Las superficies y los bordes de *material* proporcionan pistas visuales que se basan en la realidad. El uso de atributos táctiles familiares ayuda a los usuarios a comprender rápidamente las posibilidades. Sin embargo, la flexibilidad de *material* crea nuevas posibilidades que reemplazan a aquellas en el mundo físico, sin romper las reglas de la física.

Los fundamentos de la luz, la superficie y el movimiento son clave para transmitir cómo los objetos se mueven, interactúan y existen en el espacio y en relación entre sí. La iluminación realista muestra costuras, divide el espacio e indica partes móviles.

## Bold, graphic, intentional

Los elementos fundamentales del diseño basado en la impresión (tipografía, cuadrículas, espacio, escala, color y uso de imágenes) guían los tratamientos visuales. Estos elementos hacen mucho más que complacer a la vista del usuario. Crean jerarquía, significado y enfoque. Las elecciones de color deliberadas, las imágenes de borde a borde, la tipografía a gran escala y el espacio en blanco intencional crean una interfaz gráfica y audaz que sumerge al usuario en la experiencia amigable.

El énfasis en las acciones del usuario hace que la funcionalidad central sea inmediatamente evidente y proporciona puntos de referencia para el usuario.

* Environment & Elevation & shadows
* Style
* Layout
* Components
* Patterns
* Growth & communications
* Usability
* Resources

## Style

[Color](https://material.io/guidelines/style/color.html)

[Icons](https://material.io/guidelines/style/icons.html)

[Imagery](https://material.io/guidelines/style/imagery.html)

[Typography](https://material.io/guidelines/style/typography.html)

[Writing](https://material.io/guidelines/style/writing.html)

## Layout

[Principles](https://material.io/guidelines/layout/principles.html)

[Units & measurements](https://material.io/guidelines/layout/units-measurements.html)

[Metrics & keylines](https://material.io/guidelines/layout/metrics-keylines.html)

[Structure](https://material.io/guidelines/layout/structure.html)

[Responsive UI](https://material.io/guidelines/layout/responsive-ui.html)

[Split screen](https://material.io/guidelines/layout/split-screen.html)

## Usability

[Accessibility](https://material.io/guidelines/usability/accessibility.html)

[Bidirectionality](https://material.io/guidelines/usability/bidirectionality.html)

## UX & UI & Material Desing

La usabilidad y la funcionalidad son conceptos que van más allá de la estética, lo más importante es que el diseño debe funcionar. Es necesario ser conscientes de cuándo debemos recurrir a su uso y cuándo no. Material Design tiene definidas normas para casi todos los detalles y se mantienen independientemente del tamaño de pantalla. Precisamente esa transversalidad es su punto fuerte.

## Referencias

* <http://uxdesign.com/ux-defined>
* <http://uxdesign.com/ux-defined-2>
* <https://uxpa.org/resources/definitions-user-experience-and-usability>
* <http://uxmyths.com/post/1533970267/myth-27-ux-design-is-about-usability>
* <http://www.uxisnotui.com/>
* <http://www.workbridgeassociates.com/blog/posts/2013/10/17/workbridge-san-francisco-ui-ux-whats-the-difference-in-terms-of-design>
* [Buley, Leah (2013) The User Experience Team of One: A Research and Design Survival Guide. Rosenfeld Media](http://www.amazon.co.uk/The-User-Experience-Team-One/dp/1933820187)
* [Moule, Jodie (2012) )Killer UX Design. SitePoint](http://www.amazon.co.uk/Killer-UX-Design-Jodie-Moule/dp/0987153099/ref=sr_1_1?s=books&ie=UTF8&qid=1412437688&sr=1-1&keywords=killer+ux+design)

# MANUAL DE INSTALACIÓN, CONFIGURACIÓN Y PARAMETRIZACIÓN

Versión 1.0

**CONTROL DE CAMBIOS:**

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Autor** | **Referencia del cambio/revisión** |
| 01.ago.2023 | Patricio Paccha | Borrador |
|  | Carlos Anchundia | Borrador |
|  | Monserrate | Revisión |
|  |  |  |

**APROBACION:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisado por** | **Observaciones** | **Sumilla** |
|  |  |  |  |
|  |  |  |  |

**MANUAL DE INSTALACIÓN, CONFIGURACIÓN Y PARAMETRIZACIÓN**

**NOMBRE SISTEMA/MODULO**

## OBJETIVO

* Describir la instalación y configuración del módulo

## ÁMBITO

* Descripción del componente a desplegar

## MÓDULO O FUNCIONALIDAD

|  |  |
| --- | --- |
| **Fecha** |  |
| **Descripción** |  |
| **Motivo** | **[ ]** Nuevo  **[ ]** Actualización  **[ ]** Eliminación |
| **Integrador** |  |

## INSTALACIÓN

1. **Instrucciones**

…

1. **Prerrequisitos**

…

1. **Componentes y ramas en git/repositorio**

…

1. **Dependencias modulares/producto**

…

1. **Referencias documentales para usuarios**

…

1. **Referencias documentales técnica**

…

1. **Observaciones**

…

## CONFIGURACIÓN

1. **Instrucciones**

…

1. **Prerrequisitos**

…

1. **Componentes y ramas en git/repositorio**

…

1. **Dependencias modulares/producto**

…

1. **Referencias documentales para usuarios**

…

1. **Referencias documentales técnica**

…

1. **Observaciones**

…

## PARAMETRIZACIÓN

1. **Instrucciones**

…

1. **Prerrequisitos**

…

1. **Componentes y ramas en git/repositorio**

…

1. **Dependencias modulares/producto**

…

1. **Referencias documentales para usuarios**

…

1. **Referencias documentales técnica**

…

1. **Observaciones**

…

|  |  |  |
| --- | --- | --- |
| **Elaborado por:** |  | **Revisado por:** |
| Nombre  CARGO |  | Nombre  CARGO |

**Nota:**

Cada liberación se debe realizar con el respectivo PDF

Las secciones que no aplican deben ser eliminadas

# ANEXOS:

* Ninguno por ahora.

# Estándares de desarrollo

Codificación

Base da datos

Organización

# Metodología de desarrollo y artefactos

No se condiciona la aplicación de una metodología de desarrollo debido a diversidad de la naturaleza de las posibles soluciones, trabajo en equipo o problemática a tratar. Sin embargo, es posible recomendar distintos procesos de desarrollo, se instan el seguimiento de prácticas, y se exigen el establecimiento de hitos para seguimiento y control sobre el avances o riesgos de proyectos.

Para esto existen (1) recomendaciones sobre procesos metodológicos que pueden ser aplicados en cada proyectos, (2) prácticas requeridas que deben cumplirse, pero que dependerán del proceso metodológico a utilizar, y (3) reuniones que deben ser incluidas en los cronogramas de cada proyecto.

## Recomendación sobre procesos metodológicos

### Ejemplos de Ciclos de vida secuenciales

* V-Model
* IBM RESTful API Design
* API-First Design
* Spec-Driven Development

### Ejemplos de Ciclos de vida evolutivos

* RAD
* RUP/UP
* IBM Garage
* Kanban/Scrum

## Prácticas requeridas

Las prácticas son requeridas, sin embargo, dependerán de los procesos de desarrollo elegidos.

### Especificación de requerimientos

* Casos de uso
* Historias de usuario, Carácterísticas (Features), Criterios de aceptación, Pruebas de unidad, Prototipos, etc,
* Diagramas complementarios arquitectonicos: C4, Componentes, estado, despliegue, secuencia, actividades, clases, MER, Diccionarios, etc.
* Formularios estandarizados de visionado

### Aseguramiento y Control de calidad

* Especificación y reporte de casos de prueba
* Reporte sobre análisis estático de código
* Reporte sobre complimiento de pruebas automatizadas para TDD/ATDD
* Especificación de procesos de CI/CD

## Actividades de seguimiento y control

Dentro del cronograma de actividades se deberá considerar las siguientes sesiones de trabajo en la que participaran todos los actores de la CPGAA

### Reuniones Kickoff

* Se deberá realizar una reunión inicial para dar a conocer la visión y propósito del proyecto.

### Reuniones de iteración o progreso

* Se deberán realizar diversas reuniones para reportar sobre el avance del proyecto.

### Reuniones de Showcase/Demo

* Será necesario realizar una demostración del producto/servicio de forma parcial o total.

### Reuniones de revisión

* Se deberá considerar actividades de revisión o evaluación con cliente o usuarios finales

### Reuniones de entrenamiento o transferencia de conocimiento

* Al finalizar el desarrollo de la aplicación se deberá realizar el proceso de capacitación o transferencia de conocimiento. La capacitación a usuarios finales podrá ser realizada a través de recursos alternativos de acuerdo a las consideraciones de cada proyecto.

# Consideraciones de la infraestructura

Ambientes

Repositorios

Seguridad

Integración

## Estándares de desarrollo

* Base de datos,
* Código (Profile por tipo de lenguaje cada uno de los que lleva un TICs),
* UI

## Artefactos:

* Diccionario de datos (Diagrama de base de datos)
* Diagrama arquitectónico/Modelo C4/4+1 (Integración)

(patmic)

## **Plataforma**

* Virtual Desktop Interface (DGIP), solicitar al personal del FIS (director de TIC, subdecano )
  + Linux/Win
  + minikube
* Repositorio de productos
* Buenas prácticas para los ambientes dev,test,pro

**TEMAS próxima reunión:**

- Informe de la comisión: resultados de los artefactos realizados

- ver estados de las TICs propuestas anteriormente

- Prácticas profesionales

- Gestión de plataforma

Puntos del informe:

* Documentos de la unidad
* Documento navegable

(definir perfil)

)<https://www.google.com/imgres?imgurl=https%3A%2F%2Fjarroba.com%2Fwp-content%2Fuploads%2F2012%2F03%2Flogica.png&tbnid=Cus7jEQMnmLqCM&vet=10CAIQxiAoAGoXChMI0JronYitgQMVAAAAAB0AAAAAEAY..i&imgrefurl=https%3A%2F%2Fjarroba.com%2Fmodelo-41-vistas-de-kruchten-para-dummies%2F&docid=bdvS0birhtWAdM&w=500&h=299&itg=1&q=modelo%204%2B1&ved=0CAIQxiAoAGoXChMI0JronYitgQMVAAAAAB0AAAAAEAY>

1. https://docs.oracle.com/javase/specs/ [↑](#footnote-ref-2)
2. https://www.oracle.com/technetwork/java/codeconventions-150003.pdf [↑](#footnote-ref-3)
3. https://angular.io/guide/styleguide [↑](#footnote-ref-4)
4. https://www.oracle.com/technetwork/java/codeconventions-150003.pdf [↑](#footnote-ref-5)
5. **https://material.io/guidelines/#** [↑](#footnote-ref-6)