GUIA METODOLÓGICA
PARA LOS DESARROLLADORES
YO CUIDO LO PÚBLICO ADMINISTRADOR
SOLUCIONES MOVILES 4
PROYECTO FÁBRICA DE SOFTWARE GRUPO 2

Soluciones y Servicios Tecnológicos
Dirección de Gobierno en línea

Bogotá, D.C, abril de 2014

FORMATO PRELIMINAR AL DOCUMENTO

Título:	**GUIA METODOLÓGICA PARA LOS DESARROLLADORES**				
Fecha elaboración aaaa-mm-dd:	2014-02-11				
Sumario:	Este documento brinda pautas y recomendaciones a los desarrolladores para que sean tenidas en cuenta durante el proceso de construcción de software de la aplicación Yo Cuido Lo Público Administrador del proyecto Soluciones Móviles 4				
Palabras Claves:	Código fuente, mensajes, pautas, recomendaciones.				
Formato:	DOC		Lenguaje:		Español
Dependencia:	Ministerio de Tecnologías de la Información y las Comunicaciones: Dirección de Gobierno en línea – Soluciones y Servicios Tecnológicos.				
Código:	GLFS2-SM4-GM	Versión:	2.0	Estado:	Aprobado
Categoría:					
Autor (es):	Cristina Cortes Albadan Líder Técnico UT Software Works		Firmas:		
Revisó:	Mónica Monroy Consultor Procedimientos y herramientas de Interventoría Consorcio S&M Jorge Santiago Moreno Dirección de Gobierno en línea Luisa Medina Dirección de Gobierno en línea Fernando Segura Asesor Secretaría de Transparencia				
Aprobó:	Luis Felipe Galeano Arquitecto IT Consorcio S&M Rafael Londoño Dirección de Gobierno en línea				
Información Adicional:	No Aplica				
Ubicación:	El archivo magnético asociado al documento está localizado en el 24 – SOLUCIONES MOVILES 4 en la siguiente ruta: 03. Fase de Ejecucion / 02. Diseno / 01. Diseno Detallado / 02. Arquitectura				

CONTROL DE CAMBIOS

VERSIÓN	FECHA	No. SOLICITUD	RESPONSABLE	DESCRIPCIÓN
1.0	2014-02-11	No aplica	UT Software Works	Creación del documento
1.1.	2014-02-28	No aplica	UT Software Works	Ajustes solicitados por Interventoría Comentarios sección 4.1 Mensajes documentación técnica a código fuente, 8. Depuración y Validación De Código Fuente, 9. Consideraciones especiales de la solución y 10. Terminología
2.0	2014-04-24	No aplica	UT Software Works	Aprobación del documento

TABLA DE CONTENIDO

LISTA DE TABLAS

DERECHOS DE AUTOR

CRÉDITOS

En un trabajo conjunto entre los consultores de la Dirección de Gobierno en línea, Secretaria de Transparencia, las firmas Consorcio S&M y la Unión Temporal Software Works, se ha generado el presente documento siguiendo los estándares establecidos en el Sistema de Gestión de Calidad de la Dirección de Gobierno en línea para el proyecto: **IMPLEMENTACIÓN DE SOLUCIONES TECNOLÓGICAS, BAJO EL MODELO DE FÁBRICA DE SOFTWARE PARA LAS INICIATIVAS DEL PLAN VIVE DIGITAL A CARGO DEL PROGRAMA AGENDA DE CONECTIVIDAD Y LA EVOLUCIÓN DE LAS SOLUCIONES QUE SOPORTAN LA ESTRATEGIA DE GOBIERNO EN LÍNEA GRUPO 2.**

Este documento fue revisado y aprobado por los consultores y profesionales de la Dirección de Gobierno en línea, previa validación de la empresa interventora del contrato Consorcio S&M.

1. AUDIENCIA

Este documento está dirigido a los integrantes de los equipos de la entidad Secretaría de Transparencia, Dirección de Gobierno en línea, el Consorcio S&M y la Unión Temporal UT Software Works que participan en el proyecto. Este documento es aplicable a la solución Yo Cuido Lo Público Web del proyecto Soluciones Móviles 4 el cual debe ser conocido por los miembros de los equipos del proyecto: **IMPLEMENTACIÓN DE SOLUCIONES TECNOLÓGICAS, BAJO EL MODELO DE FÁBRICA DE SOFTWARE PARA LAS INICIATIVAS DEL PLAN VIVE DIGITAL A CARGO DEL PROGRAMA AGENDA DE CONECTIVIDAD Y LA EVOLUCIÓN DE LAS SOLUCIONES QUE SOPORTAN LA ESTRATEGIA DE GOBIERNO EN LÍNEA GRUPO 2.**

2. INTRODUCCIÓN

La guía metodológica presentada en este documento brinda a los equipos de desarrollo de la fábrica de software una serie de pautas y recomendaciones metodológicas y técnicas para que sean tenidas en cuenta durante el proceso de construcción de software.

Las recomendaciones presentadas son basadas principalmente en las buenas prácticas de la industria y en la experiencia en proyectos de desarrollo de software de las empresas que conforman la Unión Temporal Software Works.

El objetivo final de establecer una guía metodológica es hacer las soluciones más entendibles, es por esto que se debe establecer estándares y guiar a los desarrolladores de software para cumplir con buenas prácticas en el proceso de construcción que mejoran principalmente los niveles de usabilidad y mantenibilidad de las aplicaciones desarrolladas.

3. PAUTAS PARA PRESENTACIÓN DE MENSAJES

A continuación se describen algunas pautas y recomendaciones a la hora de presentar mensajes a los usuarios finales mediante la interfaz de usuario de la aplicación:

3.1 MENSAJES INFORMATIVOS O DE CONFIRMACIÓN

- Deben manejar un formato estándar en toda la aplicación. Esto incluye tipo y tamaño de letra, color y posición y forma de presentación (label, messageBox, div layer, etc.)
- Los mensajes deben presentarse en idioma español.
- La redacción debe ser clara y concisa. No se deberían superar los 200 caracteres.

3.2 MENSAJES DE VALIDACIÓN DE CAMPOS

- Deben manejar un formato estándar en toda la aplicación. Esto incluye tipo y tamaño de letra, color y posición y forma de presentación (label, messageBox, div layer, etc.)
- Los mensajes deben presentarse en idioma español, con colores o un símbolo de advertencia que indiquen el campo que está causando el error de validación en caso de no tener texto asociado al error.
- La redacción debe ser clara y concisa. No se deberían superar los cien (100) caracteres.
- Para aplicaciones web, las validaciones en lo posible deben hacerse del lado del cliente (por ejemplo con JavaScript).

3.3 MENSAJES DE ERROR O DE EXCEPCIÓN

- Debe manejar un formato estándar en toda la aplicación. Esto incluye tipo y tamaño de letra, color y posición y forma de presentación (label, messageBox, div layer, etc.)

- Los mensajes de error, en lo posible, deben presentarse en idioma español e indicar claramente al usuario el problema que se está presentando.

- Los mensajes de excepción no deben brindar información técnica del error al usuario final tal como el fragmento de código que generó la excepción o el nombre del objeto en la base de datos.

- Se recomienda que durante el tiempo de pruebas, estabilización y producción del sistema se habilite un log con el detalle de los errores que se presentan en la ejecución de la aplicación para que sean analizados periódicamente por el equipo de desarrollo o el administrador del sistema.

4. PAUTAS PARA CODIFICACIÓN

4.1 MENSAJES DOCUMENTACIÓN TÉCNICA A CÓDIGO FUENTE

Todo desarrollador debe garantizar que el código generado por él sea entendible y quede documentado de manera que cualquier otro desarrollador pueda comprender dicho código y sea capaz de realizar modificaciones sobre el mismo. Para esto, se recomienda al desarrollador el uso de plantillas XML para la generación de documentación técnica en el código fuente. Estas plantillas deben incluir la siguiente información:

- Resumen o descripción del elemento que se está documentando
- Para los métodos o procedimientos se debe documentar el objetivo de cada parámetro y valor de retorno esperado.

Algunos datos adicionales a incluir en la documentación del código son:

- Autor del cambio
- Fecha de creación o modificación del elemento que se está documentando
- Versión

Una ventaja de documentar el código fuente utilizando este estándar es que la documentación técnica del proyecto se pueda generar posteriormente en diferentes formatos utilizando herramientas de generación automática de documentación como NDoc para Administrador Yo Cuido Lo Público Web y javadoc para los Servicios Web.

4.2 NOMENCLATURA DE NOMBRAMIENTO DE CÓDIGO FUENTE

Es importante que el equipo de trabajo del proyecto, en cabeza del líder técnico, establezca un estándar de nomenclatura para el código fuente que desarrollan. Éste puede variar dependiendo la naturaleza misma del proyecto, el lenguaje de programación a utilizar y si se está trabajando sobre un framework de desarrollo o sobre alguna plataforma base, ya que generalmente éstas ya tienen definido una nomenclatura y resulta conveniente adaptarse a ella. Sin importar que la metodología sea muy conocida, se deberá elaborar un documento donde se

explique claramente la nomenclatura seleccionada y como se debe entender o interpretar.

Las convenciones de nombres hacen los diseños y programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre un elemento de código, por ejemplo, cuando es una constante, un paquete, una clase, un procedimiento almacenado o una tabla, que puede ser útil para entender el código rápidamente.

Algunos tipos de estilos de nomenclatura son los siguientes:

"c" = camelCase, forma de escribir una palabra donde su primera letra está en minúsculas, y la primera letra de las subsiguientes palabras en mayúsculas, ejemplo: codigoMunicipio.

"P" = PascalCase, forma de escribir una palabra donde la primera letra está en mayúscula, y la primera letra de las subsiguientes palabras igualmente en mayúscula, ejemplo: CodigoMunicipio.

"_" = Prefijo con raya inferior. (ejemplo: codigo_municipio)

"X" = No aplicable o libre, varias palabras separadas por espacio en blanco. (ejemplo: codigo municipio)

"min" = Minúsculas. (ejemplo: codigo)

"May"= Mayúsculas. (ejemplo: CODIGO)

4.3 LEGIBILIDAD DEL CÓDIGO FUENTE

El desarrollador debe promover el uso de mecanismo que ayuden a mantener un código entendible y legible. Se recomiendan los siguientes:

- Utilizar sangría, es decir desplazar bloques de código hacia la derecha insertando espacios o tabuladores de acuerdo al nivel de anidamiento. Esto teniendo en cuenta que los compiladores suelen ignorar dichos espacios en blanco y sí se mejora notablemente la legibilidad del código.

- Los nombres de las variables, constantes, clases, métodos, y en general cualquier elemento deben estar relacionados con nombres del negocio de manera que fácilmente se identifique su función dentro del código sin necesidad de acudir a comentarios adicionales que expliquen su objetivo.

- Las clases o procedimientos deben estar agrupados en paquetes lógicos de trabajo o espacios de nombre.

- Algunos lenguajes de programación permiten escribir grandes fragmentos de código en una sola línea, en lo posible evite esta práctica ya que dificulta la legibilidad. Utilice espacios y saltos de línea adecuadamente.

- Se deben seguir los lineamientos correspondientes a la arquitectura planteada para cada solución respetando los patrones de diseño establecidos y las responsabilidades de cada componente. Esto implica por ejemplo no escribir lógica de negocio o de acceso a datos en capas de presentación.

Ejemplo de definición de un estándar de nomenclatura de objetos:

Tabla 1. Estándar nombres de objetos código fuente

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
Paquetes	Min	El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, debe comenzar con las dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981, seguido del nombre de dominio de alto nivel, actualmente com, edu, gov, net, org. Seguido del nombre de la Organización, seguido de las letras que identifican el componente. Y los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada aplicación. Las cuales pueden indicar sub-componentes, servicios generales, sub-proyectos o máquinas.	co.gov.gel.urnavirtual.registro co.gov.pec.portal
Clases	P	Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivas. Usar palabras completas, evitar acrónimos	class Auditor; class EstadisticoVital;

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
		y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL or HTML). El nombre de cada una de las clases, debe especificar el objeto en español para la cual fue construido	
Interfaces	P	Los nombres de las interfaces siguen la misma regla que las clases.	interface ConsultaTablasModificadas, interface ModificaNacidoVivo;

Métodos	C	Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	calcularSaldo();
			calcularDigivoVerificador();
			leerTarifa();

Atributos / Variables de las clases	C	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres underscode "_" o signo del dólar "$". Los nombres de las variables deben ser cortos pero con significado. La elección del nombre de una variable puede ser un mnemónico, designado para indicar a un observador casual su función. Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales o en los ciclos for. Nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.	int i; char c; float tasaCrecimiento; int numeroCertificado;
Constantes	MAY	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas	static final int TASA = 4; static final int DIAS_MAXIMO = 9999;

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
		separando las palabras con un subguión ("_"). (Las constantes ANSI se deben evitar, para facilitar su depuración.)	
Formularios web	P	Los nombres de las páginas o interfaces de usuario deben comenzar con un verbo, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de los formularios simples y descriptivos. Usar palabras completas, evitar acrónimos y abreviaturas.	ConsultarSolictudIss.xxx AplicarDv.xxx CalcularSaldoCuenta.xxx ConsultarDatosRepresentante.xxx Donde xxx puede ser jsp o aspx.
Procesos	X	PRX.Y.Z: Nombre del proceso comienza con las letras "PR", seguido de un numero consecutivo que comienza en 1, seguido de un punto y otro consecutivo que comienza en 1, seguido de un punto y otro consecutivo que comienza en 1, esto se utiliza para indicar el proceso general y los sub procesos que conforman el proceso general y así sucesivamente si se descompone en mas sub procesos, seguido del nombre del proceso, máximo cuatro palabras necesarios para describir el proceso.	PR1 Consultar Hechos Vitales Nacido Vivo. PR2.Administración Sistema PR2.1 Administrar usuarios PR2.1.1 Dar de alta usuario PR2.1.2 Dar de baja usuario
Requerimientos	X	RFX: Nombre del requerimiento funcional comienza con las letras "RF", seguido de un numero consecutivo que comienza en 1, seguido del nombre del requerimiento (palabras necesarios para describir el requerimiento)	RF1 Ingreso al sistema RF2 Evaluar solicitudes
Requerimiento no funcional	X	RNFX: Nombre del requerimiento no funcional comienza con las letras "RNF", seguido de un numero consecutivo que comienza en 1, seguido del	RNF1 Manejo de grandes volúmenes de información RNF2 Manejo de concurrencia

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
		nombre del requerimiento (palabras necesarios para describir el requerimiento)	
Caso de Uso	X	CUX: Nombre del caso de uso comienza con las letras “CU”, seguido de un nemotécnico de dos (2) letras que indica el módulo al que pertenece y luego seguido de un número consecutivo que comienza en 01, seguido del nombre del caso de uso, el cual comienza con un verbo en infinitivo. En caso de metodología donde se usen Historias de Usuario, cada equipo por demanda definirá el estándar que corresponde acorde con la solución, por ejemplo puede tratarse de historias de usuario para lo cual se utilizará HUX, donde X corresponde al número de la historia de usuario.	CU-WB-02 Consultar Tabla de Referencia CU-WB-02 Crear Columna Tabla de Referencia CU-LD-01 Ejecutar paquetes de trabajo
Actores	X	Los nombres de los actores son sustantivos y comienzan con la primera letra de cada palabra en mayúsculas y las demás letras en minúsculas, cada palabra esta separa por un espacio en blanco	Administrador Entidad Funcionario MPS Médico
Casos de prueba	X	CP-CUX-99: Nombre del caso de prueba, comienza con las letras "CP", seguido de guion (-), seguido por el código del caso de uso (asociado al caso de prueba), seguido guion (-), seguido de un consecutivo que comienza en 1 (un caso de uso puede tener varios casos de prueba), seguido del nombre del caso de prueba.	CP-CU1-1 Probar ingreso al sistema. CP-CU2-1 Probar presentación de solicitud
Diagramas de secuencia	X	SQ-CUX-99: Nombre del diagrama de secuencia, comienza con las letras "SQ", seguido de guion (-), seguido por el código del	SQ-CU1-1 Configurar parámetros del log SQ-CU1-2 Configurar parámetros bitácora

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
		caso de uso (asociado al diagrama de secuencia), seguido guion (-), seguido de un consecutivo que comienza en 1 (un caso de uso puede tener varios diagramas de secuencia uno por cada flujo o escenario), seguido del nombre del caso de uso.	
Diagramas de actividad	X	AC-CUX-99: Nombre del diagrama de actividad, comienza con las letras "AC", seguido de guion (-), seguido por el código del caso de uso (asociado al diagrama de actividad), seguido guion (-), seguido de un consecutivo que comienza en 1 (un caso de uso puede tener varios diagramas de actividad, uno por cada flujo o escenario), seguido del nombre del caso de uso.	AC-CU1-1 Configurar parámetros del log AC-CU1-2 Configurar parámetros bitácora
Diagramas	X	Los nombres de los demás diagramas son iguales a los nombres de los paquetes a los que pertenecen, si hay más de uno se puede cambiar el nombre o agregar un numero consecutivo	Solicitante Servicios al solicitante Servicios al funcionario

4.4 PRUEBAS UNITARIAS

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione adecuadamente por separado.

La recomendación al momento de diseñar las pruebas unitarias es que cumplan las siguientes características:

- Automatizable: programada en alguna herramienta
- Completas: cubrir la mayor cantidad de código.
- Repetibles: no se deben crear pruebas que solo pueden ejecutarse una vez.

- Independientes: la ejecución de una prueba no debe afectar la ejecución de otra.

- Responsables: las pruebas deben ser consideradas con la misma importancia que el proceso de codificación, deben aplicarse en las situaciones que lo ameritan siendo responsables en su uso, con la misma profesionalidad, documentación, etc.

Si el proyecto incluye la creación de pruebas unitarias se recomienda al desarrollador el uso de herramientas donde se puedan codificar las pruebas unitarias más relevantes y automatizar su ejecución en proyectos independientes. Ejemplos de estos programas son JUnit, NUnit, PHPUnit.

En un desarrollo guiado por pruebas incluso se involucra una práctica que propone escribir primero las pruebas unitarias y se verifica que la prueba falle, luego se implementa el código que hace que la prueba pase satisfactoriamente. La idea es que los requisitos sean traducidos a pruebas, de este modo, cuando las pruebas pasen se garantizan que el software cumple con los requisitos que se han establecido al menos de manera unitaria.

4.5 ASPECTOS DE SEGURIDAD DE LA APLICACIÓN

El desarrollador debe garantizar no solo la funcionalidad definida sino aquellos requerimientos no funcionales como la seguridad. Algunas recomendaciones básicas son:

- Los mensajes de excepción no deben brindar información técnica del error al usuario final, tal como el fragmento de código que generó la excepción o el nombre del objeto en la base de datos. Esta información es utilizada como flanco de ataques potenciales a la aplicación.

- Asegúrese de compilar en modo Release al momento de pasar la aplicación a producción

- Nunca establezca una contraseña nula o que sea igual al usuario. Utilice contraseñas de al menos ocho (8) caracteres de longitud, que incluya números al menos una letra mayúscula y al menos un carácter especial.

- Verifique que los datos de entrada se ajusten totalmente al formato y tipo esperado. (esto ayuda a prevenir ataques de Cross-Site Scripting y SQL Injection)

- En el caso de que la aplicación web, o una sección concreta, requiera autenticación o identificación mediante usuario y contraseña hay que cuidar que en todas las páginas afectadas se realiza la comprobación de identidad,

puesto que un error muy común es realizar esa validación simplemente en la página de entrada.

- Es necesario que el desarrollador conozca los posibles riesgos derivados de una incorrecta programación y sepa cómo evitarlos. Se sugiere por lo tanto que se programe con el líder de seguridad o encargado del tema una charla informativa para concientizar al equipo en este aspecto y se ejecute un plan se seguridad para el proyecto.

- Se recomienda consultar sobre las vulnerabilidades de seguridad reportadas para la plataforma o lenguaje de programación.

Para los proyectos de la fábrica de software de Gobierno en línea es muy importante considerar las pautas mencionadas en el manual de interoperabilidad de GEL, tal como el cumplimiento de GEL-XML. De igual manera se recomienda al equipo de desarrollo y pruebas verificar las recomendaciones y pautas descritas en los lineamientos básicos de seguridad para el desarrollo y la guía de pruebas OWASP VER 3.0 Se recomienda al equipo del proyecto acordar con el cliente la necesidad puntual de cumplimiento de éstos estándares y documentar el criterio de aceptación para tal fin, para ver estos documentos se encuentran en: http://disico.com.co:8086/confluence/ en la carpeta 00-Fábrica de Software / Módulo Documentos / Administración / 1.Iniciación / Documentos Referencia.

5. PAUTAS PARA ESTÁNDARIZAR BASES DE DATOS

En este capítulo se hace referencia a los criterios que se deben tener en cuenta por parte de los miembros del equipo de trabajo al momento de crear objetos de la base de datos tales como el nombramiento adecuado, la actualización de la documentación y la seguridad, para lograr un uso adecuado y facilitar el ingreso de nuevos miembros al equipo de trabajo.

5.1 NOMENCLATURA DE NOMBRAMIENTO DE OBJETOS DE BASE DE DATOS

Tabla 2. Ejemplo estándar nombres de objetos bases de datos

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
Paquetes	Min	Los nombres de los packages son similares a los nombres de los stored procedures, pero se les antepone las tres letras PKG seguido de "_".	pkg_gov_urnavirtual pkg_gov_pec_portal
Tablas, Vistas	C	Los nombres consisten en combinaciones de mayúsculas y minúsculas que expresen de forma clara y descriptiva la función que cumple la tabla o vista y en singular.	Usuario UsuarioTramite
Procedimientos almacenados	C	Los nombres consisten en combinaciones de mayúsculas y minúsculas que expresen de forma clara y descriptiva la función que cumple el Procedimiento Almacenado. Deben comenzar por las dos letras de la aplicación, seguida de un verbo en infinitivo y seguido de la entidad que está utilizando o la función que realiza el modulo, cada palabra está separada por un underscode (_).	ISS_Obtener_Tarifa
Campos	Min	Los nombres deben ser simples, representativos e intuitivos, el campo clave de una tabla de nombrarse con el prefijo Id más el nombre de la tabla. Campos que representen la misma entidad, deben estar nombrados de la	nombre_usuario id_Notaria fecha_creacion

TIPOS IDENTIFICADOR	ESTILO	REGLAS PARA NOMBRAR	EJEMPLOS
		misma manera en todas las tablas de un esquema. Por ejemplo nombrar la clave de la tabla Sales en una tabla como Id_SalesId y en otra Key_SalesKey es incorrecto. No se prefijar sistemáticamente TODOS los campos de una tabla con el nombre de la tabla o una abreviación del mismo. Esto agrega un nivel de redundancia y complejidad al sistema que no es necesario.	
Constraints	C	Se utilizan los mismos nombres de las involucradas desde el destino al inicio, pero con el prefijo “FK_”, esto generalmente lo aplica el motor de base de datos al realizar la definición del campo como foráneo.	FK_Tabla1_Tabla2
Llave Primaria	C	Se utilizan los mismos nombres de las tablas, pero con el prefijo “PK_”, esto generalmente lo aplica el motor de base de datos al realizar la definición como llave primaria.	PK_Notarias
Índices	C	Se utilizan los mismos nombres de la tabla más el campo más una descripción auxiliar, pero con el prefijo “IDX_”, esto generalmente lo aplica el motor de base de datos al realizar la definición como índice; en algunos motores se define automáticamente el índice sobre un campo tipo llave primaria, o sobre otros tipos de campos.	IDX_<TABLA>_<CAMPO>_<DESCRIPCION> IDX_Notarias_Apellido IDX_Notarias_FechaCreacion_ParaConsulta

5.2 DOCUMENTACIÓN DE OBJETOS SOBRE LA BASE DE DATOS

Todo desarrollador debe garantizar que el código generado por él sea entendible y quede documentado de manera que cualquier otro desarrollador pueda comprender dicho código y sea capaz de realizar modificaciones sobre el mismo. Para esto, se recomienda al desarrollador de base de datos documentar los objetos que implemente incluyendo una breve descripción que explique el objetivo del campo u elemento creado. Se recomienda que esta documentación se incluya directamente en el motor de base de datos de manera que, posteriormente, se pueda generar el diccionario de datos de manera automática haciendo uso de alguna herramienta diseñada para tal fin. Ejemplos de esto es el uso del campo

Descripción (description para la versión en inglés) dentro de las propiedades de las columnas de una tabla.

Para el caso de los procedimientos almacenados se debe incluir un bloque de comentario previo a la definición del procedimiento donde se incluya información que permita identificar el objetivo y las modificaciones que se han realizado sobre el mismo. Por ejemplo:

```
/*****************************************************************************
  Nombre:     URNA_PKG_PROCESAR_XML
  Objetivo:   Recibe la solicitud del registro civil en formato XML
              y lo inserta/actualiza en las tablas del sistema interno del RUAF

  REVISIONES:
  Versión      Fecha          Autor
  ------------ -------------- ----------------------------------
  1.0          2012/02/01     Pedro Fernandez
*****************************************************************************/
```

Se sugiere además al desarrollador utilizar las mismas pautas mencionadas para legibilidad de código fuente en el desarrollo de procedimientos almacenados.

5.3 ASPECTOS DE SEGURIDAD DE LA BASE DE DATOS

Si la aplicación web o CGI se conecta a una base de datos es importante emplear un usuario con el mínimo privilegio posible.

Nunca establezca una contraseña nula o que sea igual al usuario. Utilice contraseñas de al menos ocho (8) caracteres de longitud que contenga números al menos una mayúscula y al menos un carácter especial.

6. PAUTAS PARA ESTÁNDARIZAR PERSISTENCIA EN MÓVILES

En el caso de aplicaciones móviles tenemos los siguientes tipos de almacenamiento.

Tabla 3. Tipo de almacenamiento en dispositivos móviles

TIPO DE ALMACENAMIENTO	DESCRIPCIÓN	LIMITACIÓN DE TAMAÑO POR APLICACIÓN	TIPO DE APLICACIÓN
Base de datos SQLite	Usa SQL	Todo lo que pueda la memoria del dispositivo	Nativa, híbridas
Local Storage	Almacenamiento en el navegador usando par de clave-valor	5MB	Híbridas, sitios web HTML5 para móviles
webSQL	Usa SQL	5MB	Híbridas, sitios web HTML5 para móviles
Preferencias de usuario local	Almacenamiento de datos primitivos usando par clave-valor	Todo lo que pueda la memoria del dispositivo	Nativa, híbridas
Preferencias de usuario en la nube	Almacenamiento de datos primitivos usando par clave-valor	1MB para iCloud (Apple), 1kB por mensaje en Android	Nativa, híbridas
Sistema de archivos interno	Almacena cualquier tipo de archivos	Todo lo que pueda la memoria del dispositivo	Nativa, híbridas
Sistema de archivos externo	Usando tarjetas externas SD, almacena cualquier tipo de archivos	Todo lo que pueda la tarjeta externa.	Sólo aplica para Android. Nativa, híbridas

Apple solo permite descargar aplicaciones que tenga como tamaño máximo 50MB para redes de datos, para redes WIFI el límite lo impone el espacio libre del dispositivo. Aunque hay sitios que informan que Apple puede contactar al desarrollador si la aplicación excede los 200MB en la práctica hay muchas aplicaciones que exceden ese tamaño.

Para Android se permiten aplicaciones (APK) de hasta 50MB pero puede adjuntar archivos hasta por 4GB. Para Android en redes WIFI sucede lo mismo que en iPhone.

Respecto a la sincronización de datos en general no hay limitaciones de tamaño diferentes a las expuestas en el punto anterior.

El tipo de almacenamiento particular para una aplicación móvil estará ligado a la arquitectura y diseño del proyecto particular. Sin embargo va a estar dentro de los tipos listados en esta sección. La sección “CONSIDERACIONES ESPECIALES DE LA SOLUCIÓN” tendrá los detalles adicionales que apliquen al tipo de solución móvil en particular.

7. PAUTAS PARA INTERFAZ VISUAL

El programador de la interfaz de usuario deberá velar por cumplir las siguientes características:

- Ortografía y gramática impecable.
- Uso de plantillas, archivos de estilos, temas, páginas maestras y en general cualquier componente que favorezca la estandarización y mantenibilidad del sitio a nivel gráfico.
- Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores
- Velar porque la aplicación sea accesible y correctamente visible desde la mayoría de navegadores web disponibles en el mercado.
- Velar porque la aplicación se visualice adecuadamente en diferentes resoluciones de pantalla.
- Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o incluso en pantallas en blanco y negro
- Donde sea posible haga uso de tecnologías como Ajax para mejorar la experiencia de usuario.
- En lo posible mantener un código limpio donde la lógica de negocio esté separada de la lógica de presentación y evitar componentes gráficos que le impliquen al usuario descargar componentes o plugins adicionales como en el caso de Flash, Silverlight, applets, ActiveX, etc.
- Se sugiere que el equipo de pruebas diseñe casos de pruebas independientes para garantizar las pautas acordadas a nivel de interfaz de usuario.

Criterios de Usabilidad:

- Navegación Global consistente: corresponde al menú principal del sitio web. Debe aparecer de la misma forma y en la misma ubicación a lo largo del sitio web. Permite evitar las páginas huérfanas.
- Navegación de Contexto: estándar de facto en donde el usuario espera encontrar un menú secundario. Sitios profundos se pueden ver muy beneficiados con una navegación contextual.

- Ruta de Migas: mecanismo de navegación auxiliar, se ubica generalmente en la parte superior del sitio y muestra la ruta que ha seguido el usuario hasta el lugar donde se encuentra.
- URL's Limpios: esquema que utiliza las siguientes características.
- Dirección que refleja en todo momento la jerarquía del sitio.
- No incluye caracteres especiales tales como $, &, ?, etc.
- TagLine: frase que subtitula un sitio web. Se ubica debajo del identificador del sitio. Responde a las preguntas tales como ¿ donde estoy ?, ¿ para que me sirve este sitio ?. Ayuda a los usuarios a decidir si permanecen o abandonan el sitio.
- Enlaces Bien formulados: usar títulos significativos sin ambigüedades. No usar palabras como haga clic. Deben ser cortos.
- Memoria de corto plazo (7+/- 2) Max 10 seg: utilizar diseños minimalistas con el contenido necesario. Enlaces que recuerden su visita. Típicamente se utilizan enlaces color purpura, no se recomienda utilizar otro tipo de convenciones

Diseño de interfaz de usuario:

- Ubicación del logotipo: por estándares de facto debe ubicarse a mano izquierda en la parte superior.
- Interfaces en Movimiento: evitar el uso de animaciones tipo Flash en elementos que interactúan con el usuario final.
- Evitar elementos tipo publicidad: se ha demostrado que elementos en texto plano (en información institucional) llama más la atención que elementos en movimiento tipo publicidad.
- Información transmitida a través del color (daltonismo 8% hombres y 1% en mujeres): evitar el uso de colores diferentes en el texto del sitio o redundar la información con algún elemento gráfico o informativo.
- Gris tipográfico: El color producido por la caja tipográfica no debe inclinarse al negro ni al blanco, debe ser un gris equilibrado. Es preferible evitar el uso de texto justificado.
- Ancho del cuerpo del texto: debe estar entre 60 y 80 cpl (caracteres por línea). Renglones cortos implican contantes saltos de línea, renglones largos de igual manera producen fatiga visual.
- Fuentes tipográficas comunes: utilizar fuentes tipográficas universales para evitar la pérdida de control en el diseño. Ver hojas de estilo de cascada CSS a través del explorador.
- Texto subrayado: para destacar textos utilizar etiquetas tipo (énfasis moderado) o (énfasis fuerte) y no subrayar.
- Desplazamiento horizontal: basados en una resolución estándar 1024 x 768 evitar el desplazamiento horizontal. Limitar el contenido a tamaños no superiores a 980 pixeles de ancho. Use Web Developer Toolbar de Firefox.

- Vínculo a la página de inicio: debe existir como un vínculo individual y asociado al logotipo del sitio en todas las páginas. No utilizar textos tales como home, principal, etc.
- Hojas de estilos para diferentes formatos: asignar como mínimo dos (2) estilos, uno para lectura en pantalla y otro para impresión en papel. Ver etiquetas <head> y </head>
- Independencia del navegador: IE, Chrome, Firefox y Safari.

Diseño de Interacción:

- Campos Obligatorios: diferenciar los campos requeridos de los opcionales. Ubicar a mano derecha del campo.
- Asociación de etiquetas a campos: utilizar un adecuado rotulado de campos. Ubicar etiquetas en la parte superior del control o a mano izquierda.
- Validación Local de Campos: validar a través de JavaScript evitando la validación a nivel de servidor. Validar de manera dinámica por cada campo ingresado y no al finalizar el ingreso de información.
- Error de página no encontrada (404): se produce por enlaces mal formulados, en donde se debe personalizar o atrapar el error, con el objeto de personalizar el mensaje. Proporcionar vínculo al mapa del sitio o a artículos de ayuda.
- Ventanas emergentes: evitar el uso de estos elementos sino son solicitados directamente por el usuario ya que es posible que el explorador las confunda con publicidad y las bloquee por defecto o el usuario las cierre.
- Botón atrás: verificar que el botón no deje de funcionar en la totalidad del sitio. No obligar al usuario a utilizar otros botones no naturales que cumplan con la misma función.
- Tiempo de carga en las páginas: reducir tiempos de carga.
- Ejemplos en los campos de formulario: proporcionar ejemplos de diligenciamiento en los campos de formulario que sean de difícil comprensión.

Para los proyectos de la fábrica de software de Gobierno en línea es muy importante considerar las pautas mencionadas en el Manual GEL 3.1. Se recomienda al equipo de desarrollo acordar con el cliente la herramienta que se utilizará en el proyecto como validador de usabilidad y el nivel que debe superarse (se recomienda el AA), para ver este documento se encuentra en: http://disico.com.co:8086/confluence/ en la carpeta Módulo Documentos/Administración/1.Iniciación/Documentos Referencia.

8. DEPURACIÓN Y VALIDACIÓN DE CÓDIGO FUENTE

El concepto de depuración y trazado de código fuente hace parte del concepto de instrumentación de una aplicación. Generalmente ésta involucra los ciclos de desarrollo, estabilización y producción.

Cuando se habla de instrumentación en la etapa de desarrollo, son comunes los términos de depuración, rastreo y seguimiento. Básicamente estos son aquellos artefactos y herramientas que permiten obtener metadatos del código binario que se está ejecutante en un momento determinado y hace parte de las reglas escritas por los desarrolladores.

8.1 COMPILACIÓN DE LA APLICACIÓN

- DEBUG: Generalmente los entornos y plataformas de desarrollo permiten la compilación en DEBUG, lo que significa que los componentes generados guardan dentro del archivo los símbolos de depuración (metadata de los documentos de código fuente), es decir, que si se detectan incidencias en tiempo de ejecución, la aplicación va a estar en capacidad de poder informar de manera de tallada la descripción y características del evento que produjo el problema o el fallo de ejecución. Esta información dependiendo de la plataforma y el sistema operativo será capturada en los visores de evento o en los logs de aplicaciones o procesos. Este modo debe ser usado en etapas de desarrollo y estabilización de una aplicación.

- RELEASE: Es la segunda opción de compilación, en este modo el compilador genera el código binario para la ejecución de la aplicación sin incluir información de depuración dentro de los componentes, lo que permite que la ejecución del software sea más eficiente, pero al mismo tiempo si se experimenta un fallo, no es posible obtener información técnica detallada de las incidencias. A diferencia del modo DEBUG, el modo RELEASE se usa para entornos de producción.

8.2 INSTRUMENTACIÓN DE CÓDIGO

Se recomienda como práctica para el desarrollo de la solución es uso de “Debug” y “Trace”. En este caso se espera que independientemente de la plataforma de desarrollo; se establezca de manera genérica un rastreador de eventos y un log en

todas las aplicaciones, que distinga cuando los mensajes de registro que se ejecutan expongan información diferente si es código compilado en DEBUG (se escribe con debug) y si es compilado en RELEASE (se escribe con trace) una vez se han capturado una excepción o un paso importante de código lógico.

Herramientas comunes de lo anterior son el System.Diagnostics de la plataforma .Net y el log4j para el caso de la plataforma Java.

A continuación un ejemplo de uso.

```
Trace.WriteLine("Inicio del proceso");
Stopwatch sw = Stopwatch.StartNew();
MiProceso();
sw.Stop();
Trace.WriteLine("Duración del proceso: " +
sw.Elapsed.ToString());
```

8.3 RECOMENDACIONES EN CALIDAD DE ESCRITURA DE CÓDIGO FUENTE

Al detectar incidencias sobre la aplicación a través de las pruebas unitarias, pruebas funcionales o pruebas no funcionales, se hace necesario revisar en detalle el comportamiento del código fuente para lo cual se plantean las siguientes pautas:

- Utilizar las herramientas de “debugging” provistas por el IDE sobre el cual se está desarrollando la aplicación.
- Hacer ejecución paso a paso
- Utilizar puntos de control “breakpoints” para analizar el estado actual (variables, condiciones).
- Utilizar las posibilidades que brindan algunos programas de depuración para hacer “bypass” de algunas condiciones y observar el comportamiento del sistema.
- Utilizar la impresión de mensajes que muestran en ejecución los valores de variables a analizar. Se recomienda el uso de herramientas de análisis de código de tal manera que se pueda suplir altos niveles de calidad durante la etapa de desarrollo y en el resto del ciclo de vida de la codificación. En la siguiente sección se listan herramientas para tal fin.

8.4 HERRAMIENTAS DE EVALUACIÓN DE CALIDAD DE CÓDIGO FUENTE

- Microsoft .Net:
  - FxCop: Herramienta que permite el análisis de código por medio de la introspección de ensamblados de código administrados. Ver: http://msdn.microsoft.com/en-us/library/bb429476.aspx
  - StyleCop: Herramienta que analiza el código en C# para reforzar un conjunto de reglas de consistencia y estilo. Ver: http://stylecop.codeplex.com/
- Java:
  - Sonar: Herramienta de verificación de calidad de código y diseño de componentes. Ver: http://www.sonarsource.org/
  - Understand: Herramienta para análisis de código y métricas. Ver: http://www.scitools.com/
- PHP:
  - Sonar: Herramienta de verificación de calidad de código y diseño de componentes. Ver: http://www.sonarsource.org/
  - Understand: Herramienta para análisis de código y métricas. Ver: http://www.scitools.com/

9. CONSIDERACIONES ESPECIALES DE LA SOLUCIÓN

Durante el desarrollo de la aplicación se debe seguir lo descrito en este documento; para los componentes web e implementación de los procesos se debe cumplir el estándar AA definido en www.w3c.org y GEL-XML.

Se recomienda al desarrollador dar nombre a los Servicios, a los Data Sources, a los datos de entrada-salida, a las sentencias de la Lógica y a las Operaciones, con nomenclatura camelCase; Además ninguna de los nombres anteriores debe superar los quince (15) caracteres.

Si la aplicación contempla desarrollo para móviles se debe seguir lo descrito en el numeral “6. PAUTAS PARA ESTÁNDARIZAR PERSISTENCIA EN MÓVILES” de lo contrario no.

10. TERMINOLOGÍA

BACKUP: Es una copia de los datos que se encuentran en nuestro disco duro, y que se preservan en otro medio de almacenamiento (discos duros / CD's / DVD's / cintas magnéticas, etc.) con el fin de conservarlos y/o protegerlos en caso de posible daño y/o destrucción de la fuente original. Dependiendo de su importancia, será decisión del usuario generar copias parciales («mis documentos», por ejemplo) o totales (particiones o discos duros completos). Para ello existe un sinnúmero de programas que permiten realizar esta labor de manera sencilla e intuitiva

BASE DE DATOS: Es un archivo compuesto por registros. Cada registro contiene uno o varios campos de datos significativos a los mismos. Con una base de datos se pueden realizar operaciones de búsquedas, ordenamientos, reordenamientos y otras funciones.

Por ejemplo, un colegio tendrá una base de datos de sus alumnos. Cada registro representará a un estudiante y en cada campo se indicará información sobre éste (apellidos, nombres, sexo, fecha de nacimiento, domicilio, etc.).

BAT:(BATCH) Un archivo de texto ASCII que contiene una secuencia de comandos del sistema operativo que al ejecutarse realiza alguna rutina para el computador.

BAUDIO: Término utilizado en comunicaciones para medir la velocidad de un dispositivo. Indica el número de intervalos por segundo que supone una señal

BETA: Versión nueva de un programa que está disponible para que los usuarios puedan ir probándolo en situaciones reales. Se caracteriza por traer la mayoría de las funciones que tendrá la versión final. Al ser una versión previa a la final, puede presentar inestabilidades por lo que solo se recomienda su utilización en entornos controlados, cuando no sea importante si se produce un error o por usuarios experimentados.

BINARIO: Que tiene dos componentes, alternativas o resultados. El sistema de numeración binario tiene como base 2, de modo que los valores se representan como combinaciones de dos dígitos cero (0) y uno (1).

BIOS: Acrónimo de Basic Input Output System / Sistema de Entrada y Salida.

Es un programa incorporado en un chip (memoria ROM) de la placa base que al prender la computadora se encarga de realizar las funciones básicas de manejo y configuración del computador.

BIT: Unidad mínima de información manejada por la PC. La presencia de una señal magnética que se representa para nosotros como uno (1) y la ausencia de la señal magnética como cero (0).

BOOKMARK: En los navegadores, es un vínculo a una página web u otro URI (Uniform Resource Identifier) que el usuario visita con regularidad. Éste se almacena en el equipo, permitiéndole un acceso sumamente rápido al sitio web. También conocido como “favoritos”.

BPS: Bits por segundo, unidad de transmisión de datos, empleada, principalmente, en referencia a módems o comunicaciones de red.

BUFFER: Memoria dedicada a almacenar temporalmente la información que debe procesar un dispositivo de hardware (disco duro o cd) para que lo pueda mantener el rendimiento de la transferencia. Un buffer de tamaño inadecuado da origen a la falla en grabar CDs.

BUS: Es el canal por el que circula información electrónica en forma de bits. El ancho de bus es el número de bits transmitidos simultáneamente por el bus.

BYTE: Unidad de información, compuesta de ocho (8) bits consecutivos. Cada byte puede representar, por ejemplo, una letra.

CODIGO: Hace referencia al texto escrito que representa los símbolos propios de un lenguaje de desarrollo de software, el conjunto de símbolos ordenados tienen una semántica que permite definir el comportamiento de un sistema particular.

DEBUG (Versión de Depuración): La versión de depuración del programa se compila sin optimizar y toda la información de depuración simbólica. La optimización complica la depuración, ya que la relación entre el código fuente y las instrucciones generadas es más compleja.

DEBUGGER: Depurador (por sus siglas en inglés) o herramienta de depuración, es una programa informático que es usado para probar y seguir otros programas (programa objetivo). El código a ser examinado puede alternativamente estar corriendo en un simulador de conjuntos de instrucciones, una técnica que permite gran poder en su habilidad de detenerse cuando se encuentran algunas condiciones específicas, pero esto será típicamente algo mas lento que ejecutar el código directamente en el procesador apropiado. Algunos depuradores ofrecen dos modos de operación - simulación full o parcial - para limitar el impacto.

EXCEPCIÓN: Representa los errores que se producen durante la ejecución de una aplicación.

IDE: Entorno de desarrollo integrado, por sus siglas en inglés, es un software que facilita el desarrollo de aplicaciones al ofrecer herramientas para escritura de código, compilación, depuración y ejecución.

RELEASE:22 (Versión de Liberación): La configuración Release del programa no contiene información de depuración simbólica y está totalmente optimizada. La información de depuración se puede generar en Archivos de base de datos de programa (C++), según las opciones del compilador utilizadas. Crear archivos PDB puede ser muy útil si luego necesita depurar la versión de lanzamiento.

.