TABLA DE CONTENIDO

1. INTRODUCCIÓN

Los sistemas de información[1] y aplicaciones de la Secretaría Distrital de Movilidad (SDM) apoyan la operatividad de los procesos y gestionan la información como insumo valioso.

La Entidad cuenta con sistemas de información que son desarrollados internamente, otros desarrollados por terceros o adquiridos a empresas desarrollo de software, por lo cual es necesario estandarizar y establecer los procesos y lineamientos de ingeniería de software.

Actualmente no existe una documentación o disposiciones explícitas para las actividades enmarcadas dentro de un proceso de desarrollo de software, lo que dificulta la unificación y estandarización de los artefactos y producto del desarrollo.

Es necesario identificar tipos de actores, participantes, roles, documentación, responsabilidades y modelos de operación para los proyectos de desarrollo de software y el ciclo de vida de las aplicaciones.

1.1. PROPÓSITO

Presentar de una forma explícita y consistente la gobernabilidad sobre el diseño y la implementación de los sistemas de información y aplicaciones de la Secretaría Distrital de Movilidad – SDM.

Este documento tiene como propósito, el establecimiento de lineamientos técnicos y estándares para el desarrollo interno o externo de sistemas de información para la SDM. Comprende la información necesaria que involucra el análisis, diseño, desarrollo e implementación de los sistemas de información bajo conceptos y tecnologías recientes.

1.2. ALCANCE

Este documento contiene las directrices y buenas prácticas para asegurar y gestionar el desarrollo y/o implementación la correcta operación y calidad de todos los sistemas de información y aplicaciones de la organización.

2. POLÍTICA DE DESARROLLO DE SOFTWARE

La política de desarrollo de software de la SDM contiene declaraciones, responsabilidades, lineamientos técnicos de arquitectura y desarrollo de software a los cuales se les debe dar cumplimiento.

La Secretaría Distrital de Movilidad a través de la Oficina de Tecnologías de la Información y Comunicaciones – OTIC, se compromete a verificar el cumplimiento y atributos de calidad de software como la modificabilidad, la escalabilidad, la seguridad de la información, entre otros contemplados en esté manual y en el Manual de Políticas Específicas de Seguridad de la Información PA04-M01, al implantar la presente política de desarrollo de software.

2.1 DECLARACIÓN DE LA POLÍTICA

- La verificación de la aplicación de la política de desarrollo de software está a cargo del equipo de Fábrica de Software perteneciente a la OTIC.
- Las personas involucradas y todos aquellos que tengan responsabilidad sobre el desarrollo de sistemas de información, deben optar los lineamientos contenidos en la presente política con la finalidad de cumplir con los pilares de la seguridad de la información del desarrollo de aplicaciones.
- La OTIC es la responsable de divulgar a las personas involucradas y vinculadas a la SDM para que cumplan con la política de desarrollo de software.
- El incumplimiento total o parcial de la presente política por parte del funcionariado, podrá constituirse como falta

disciplinaria y por lo tanto podrá dar lugar a la acción e imposición de la sanción correspondiente establecida en la ley penal, en el código disciplinario único y en las demás normas que se encuentren vigentes.

- El incumplimiento total o parcial de la presente política por parte de las y los contratistas, personal que labore en las instalaciones vinculado con proveedores de la SDM o personal externo (empresa o entidad externa), estará sujeto a las sanciones contractuales, civiles y/o penales a que haya lugar, por los daños y perjuicios generados a la Secretaría Distrital de Movilidad o a terceros.

2.2 RESPONSABILIDADES

- La política de desarrollo de software es de aplicación obligatoria para todo el personal de desarrollo de software que labore en la SDM y contratistas.
- La SDM y OTIC aprueban esta política y son responsables de la autorización de sus cambios y/o nuevas versiones.
- La persona oficial de seguridad de la información de la OTIC es responsable de impulsar la ejecución y cumplimiento del atributo de calidad: Seguridad de los sistemas de información, incluido en esta política de desarrollo de software.
- El equipo de fábrica de software de la OTIC es el responsable de impulsar la ejecución y cumplimiento de la política de desarrollo de software.
- Las personas responsables del desarrollo, adquisición y mantenimiento de los sistemas de información tienen la obligación de velar que dicho sistema considerado como un activo de información se mantenga disponible, íntegro y confidencial, mientras esté en el proceso de desarrollo, mantenimiento y puesta en marcha.
- Todos los procesos que requieran cualquier tipo de desarrollo de software deberán dar cumplimiento de la política de desarrollo de software.

2.3 REQUERIMIENTOS PARA DESARROLLO DE SOFTWARE

Todo requerimiento relacionado con Desarrollo de Software deberá ser tramitado por la Herramienta Aranda o la que haga de sus veces y soportado por el Formato de Solicitud de Requerimiento PA04-M02-F01, anexo al presente manual.

Adicionalmente el requerimiento de Desarrollo de Pruebas Sistemas de Información se deberá solicitar mediante el formato PA04-M02-F02 y para mayor claridad en el proceso del desarrollo interno está el formato PA04-M02-F03 de Especificaciones del Proceso de Desarrollo de Software y Flujograma anexo a este manual.

Todo requerimiento para desarrollo de software será revisado y evaluado por el equipo tecnológico que sea designado por la o el Jefe de la Oficina de Tecnologías y las Comunicaciones para validar la viabilidad y las capacidades tecnológicas del requerimiento.

2.4 ARQUITECTURAS DE REFERENCIA PARA EL DESARROLLO DE SOFTWARE.

Todo proyecto de desarrollo de software en ambiente Web que se desee adelantar en la entidad, debe adoptarse bajo el modelo de capas o N-Capas[2].

En las capas se puede desacoplar la infraestructura de la lógica del negocio o la implementación de un servicio de la lógica de negocio.

Para una correcta implementación de la arquitectura se debe tener en cuenta las siguientes separaciones:

- **Dominio**: Contendrá la definición del objeto de dominio, los value objec[3], las excepciones de dominio y la interfaz que va a interactuar con la infraestructura.
- **Aplicación**: Contendrá los casos de usos del dominio, en otras palabras, contendrá las acciones que podrá realizar ese objeto de dominio.
- **Infraestructura**: Contendrá la implementación de la interfaz definida en el dominio.

Se debe recordar que la capa superior podrá conocer las capas que se encuentren a un nivel inferior a ellas.

2.5 ATRIBUTOS DE CALIDAD

La construcción de una buena arquitectura de software impacta positivamente sobre su capacidad de satisfacer las necesidades de las partes interesadas.

Cada característica de está arquitectura (atributos de calidad) es una propiedad medible del sistema que permite evaluar aspectos de calidad tales como la disponibilidad, rendimiento, seguridad, usabilidad, escalabilidad, facilidad de pruebas, entre otros, como propuestos por la SDM para la construcción de todos sus productos de software.

De esta manera la entidad busca lograr de manera ordenada, estructurada, eficiente y segura, que sus aplicaciones y sus diferentes integraciones, minimicen los riesgos relacionados a la calidad, costos, tiempo y alcance; aumentando así la capacidad de aceptación del sistema por parte del usuario.

Disponibilidad: Antes de pasar un sistema a producción, cada proyecto de software deberá entregar un documento donde especifique los parámetros tenidos en cuenta para garantizar la disponibilidad del sistema.

Portabilidad: Toda aplicación desarrollada en la entidad debe tener la capacidad de cambiar de ambiente de producción de una manera ágil, sin afectar su funcionalidad inicial.

Rendimiento: Cada aplicación, hará un adecuado uso de los recursos ofrecidos para su entorno de producción de acuerdo con los que usa en condiciones específicas. También se requiere tener en cuenta el recurso utilizado en la interacción con otros productos de software para alguna funcionalidad en específico. De esta manera los tiempos de respuesta y procesamiento de la aplicación serán los apropiados.

Interoperabilidad: Todo desarrollo de software debe tener la capacidad de intercambiar datos bajo la tecnología REST[4] por medio del formato JSON[5], respetando los aspectos de seguridad establecidos en este documento.

Seguridad: Toda aplicación tendrá la capacidad de proteger la información y los datos de tal forma que los usuarios y/o sistemas no autorizados no puedan acceder a ellos. Para conceder al sistema los principios de integridad, autenticación y disponibilidad al sistema, se proponen los siguientes métodos:

Antes de realizar el paso a producción, todo proyecto de software debe contar con el visto bueno de la persona oficial de seguridad de la entidad o quien haga su vez.

- Es requisito para el paso a producción que los productos de software cuenten con un documento de análisis de riesgos como lo indica la guía de aseguramiento de aplicaciones.

- La persona oficial de seguridad deberá gestionar las vulnerabilidades en el desarrollo del software, revisando con una frecuencia mensual los logs generados por las aplicaciones o por el servidor de aplicaciones. Validar herramientas que permitan la automatización de este proceso.

- Se debe proteger la información sensible como (contraseñas, tokens, firmas digitales) en todos sus estados: tránsito, en reposo y en uso por medio de herramientas y técnicas criptográficas que garanticen la integridad y confidencialidad de la información.

Las demás que indique el Manual de Políticas Específicas en Seguridad de la Información de la SDM – PA04-M01.

Auditabilidad: Es la capacidad de que todos los sistemas deben llevar un control de Auditoría de acuerdo con las necesidades del negocio. Ejemplos del tipo de información que normalmente se audita son:

- Accesos al sistema
- Accesos y modificaciones a los recursos del sistema
- Transacciones

Cada registro de auditoria para modificación o eliminación debe tener como mínimo la siguiente información:

- Identificador de la transacción
- Identificador del usuario
- IP generadora del evento
- Fecha y hora
- Transacción: actualizar, eliminar, crear.
- Tabla
- Campo modificado
- Valor original
- Valor modificado

Usabilidad: Todos los sistemas y sus componentes desarrollados deben ser entendidos y aprendidos por el usuario de manera sencilla y práctica.

- Todo producto de software deberá seguir parámetros de usabilidad como simplicidad, organización de los elementos y adaptación a diferentes dispositivos, navegación intuitiva y accesibilidad sin importar su sistema operativo y/o navegador preferido.

- Cada componente desarrollado debe incluir documentación guía para el usuario, como lo son por ejemplo los video tutoriales, diagramas de proceso y manuales.

Escalabilidad: Cada aplicación debe poder manejar una carga constante y creciente de datos y trabajo, en simultaneo con el crecimiento de la Entidad. A fin de configurar el entorno de la aplicación para la atención de peticiones de manera conjunta y distribuir su carga de trabajo se establece:

- Se debe implementar un sistema orquestador de contenedores por medio de Docker y Kubernetes[6] para garantizar el balanceo de cargas y el auto mantenimiento ante fallos por peticiones o no disponibilidad del sistema, contemplando el uso de firewall web o físico, lo anterior para garantizar la recuperabilidad del sistema y escalabilidad horizontal.

Facilidad de pruebas: Toda aplicación permitirá realizar pruebas a los componentes desarrollados o modificados para así poder identificar si provoca efectos secundarios o adversos durante su ejecución.

A continuación, se expone cada uno de los tipos de pruebas propuestos para verificar y validar la aplicación:

Se deben realizar pruebas unitarias de código con la finalidad de garantizar el funcionamiento esperado de los métodos usados en el desarrollo.

Se aplicarán pruebas de integración sobre cada uno de los módulos y otros sistemas con los que se comunica la aplicación. De esta manera se podrá validar el comportamiento de los componentes presentes en el sistema y garantizar su funcionalidad.

Se deben realizar pruebas funcionales de sistema que validen su funcionamiento en condiciones específicas: sistemas operativos, navegadores, servidores, bases de datos o tipo de dispositivo en el que lo usen.

Posterior a una modificación y/o mejora de alguna funcionalidad en la aplicación, se realizarán pruebas de regresión para asegurar que estos cambios o adiciones no alteraron ni eliminaron funcionalidades existentes.

En caso requerirse ejecución de pruebas no funcionales, estas deben diseñarse durante el ejercicio de arquitectura especificando cada uno de los atributos de calidad. el diseño corresponde a los escenarios de calidad que debe cumplir el software para lo cual fue diseñado.

2.6 PRINCIPIOS Y PATRONES DE DISEÑO

- Principios de Programación Orientada a Objetos

El paradigma de programación orientada a objetos contiene los siguientes principios básicos:

- Abstracción
- Polimorfismo
- Herencia
- Encapsulamiento

- Principios SOLID

Los 5 principios SOLID de diseño de software son:

S – Single Responsibility Principle o principio de responsabilidad simple.

O– Open/Closed Principle o principio de abierto / cerrado.

L – Liskov Substitution Principle o principio de sustitución de Liskov.

I – Interface Segregation Principle o principio de segregación de interfaces

D – Dependency Inversion Principle o principio de inversión de dependencias.

- Todo desarrollo de software debe seguir los principios de programación orientada a objetos que se alinean con los principios SOLID de forma obligatoria.
- Todo desarrollo de software debe implementar la arquitectura de software de referencia mencionada en esta política.
- Cada desarrollador de software debe implementar patrones de diseño en los proyectos en los que participe teniendo su alcance y propósito. Dado esto, se debe utilizar de acuerdo su clasificación: patrones creacionales, patrones estructurales y patrones de comportamiento.
- Cada proyecto de desarrollo debe incluir un ORM[7] para el mapeo de base de datos, validando que dicha herramienta no posea vulnerabilidades de inyección. Dado el caso en que no se pueda implementar el ORM en módulos específicos del proyecto, se deben realizar una justificación y utilizar sentencias preparadas para prevenir ataques de SQL[8] Injection.

2.7 METODOLOGÍA DE DESARROLLO

Durante el proceso de desarrollo de software se deben tener buenas prácticas y un proceso claro para trabajar colaborativamente en equipo. A la hora de poner en marcha un proyecto, toda empresa debe asegurar que el equipo

implicado conoce las tareas y plazos de tiempo de entrega.

Scrum es una metodología de trabajo que nos ayuda a conseguirlo y que, además, permite agilizar la entrega de valor al cliente en iteraciones cortas de tiempo.

Esta etapa requiere generar los siguientes entregables, dando instrucción al procedimiento realizado para la puesta en producción de la aplicación:

Product Backlog (Artefacto): El Product Backlog es una lista emergente y ordenada de lo que se necesita para mejorar el producto. Es la única fuente del trabajo realizado por el Scrum Team.

Sprint Backlog (Artefacto): El Sprint Backlog se compone del Objetivo del Sprint (por qué), el conjunto de elementos del Product Backlog seleccionados para el Sprint (que), así como un plan de acción para entregar el Increment (como).

Se debe establecer para cada proyecto, módulo o formulario la metodología de gestión SCRUM en la cual se oriente el ciclo de vida de desarrollo en los proyectos de software de la SDM.

2.8 DOCUMENTACIÓN TÉCNICA

La OTIC debe disponer de un espacio en disco o plataforma (en este caso, se define como plataforma principal Azure DevOps) para que los equipos de desarrollo almacenen la información de documentación requerida.

Todo desarrollo que se realice en la SDM debe generar los siguientes entregables:

- Plan de trabajo o Product Backlog
- Historias de usuario.
- Diagramas de clases, modelo de datos.
- Solicitud de cambios o ajustes a software (Documento Control de Cambios)
- Documento Plan de Pruebas.
- Guía de usuario.
- Códigos fuentes documentados.
- Paso a producción de software (Documento Control de Cambios)
- Diccionario De Modelo De Datos.
- Diagramas de arquitectura de software (componentes y despliegue)

2.9 LINEAMIENTOS Y ESTÁNDARES PARA EL DESARROLLO DE SOFTWARE

- Para el manejo de nomenclatura se deben adoptar estándares que aseguran un código legible, fácil de entender y mantener.
- Todo desarrollo de software debe especificar las tecnologías con las cuales se desarrollará los aplicativos, módulos o sistemas, y deben ser un estándar para los equipos de trabajos que realicen dichos desarrollos.
- Para garantizar seguridad de la información se deben establecer para cada proyecto la separación de ambientes de construcción, testing y producción cada uno representado en una rama del repositorio del proyecto, con el objetivo de controlar de mejor manera los cambios al sistema de información y lo demás que indique el Manual de Políticas específicas de Seguridad de la Información PA04- M01.
- Todo desarrollo de software debe contar con 3 ambientes: desarrollo, pruebas y producción. Cada uno con diferente base de datos.

- Todo desarrollo de software debe tener por lo menos 2 ramas en el repositorio de código, una para producción y una para desarrollo. Se recomienda tener una rama independiente para el ambiente de pruebas.

  El desarrollo debe cumplir las siguientes métricas de calidad para la validación del código, Se sugiere SonarQube como la herramienta para este tipo de mediciones:

- Cobertura de código >70%: Esta métrica indica el porcentaje de código que está cubierto por pruebas unitarias. Una cobertura más alta sugiere que más partes del código están siendo probadas, lo que generalmente se considera un indicador de mayor calidad.

- Duplicación de código <5%: Mide el porcentaje de código duplicado en el proyecto. La duplicación excesiva puede conducir a una mayor complejidad y dificultad para el mantenimiento del código.

- Complejidad ciclomática: Esta métrica evalúa la complejidad del código midiendo la cantidad de caminos de ejecución posibles a través de un método o función. Un valor más alto indica una mayor complejidad, lo que puede hacer que el código sea más difícil de entender y mantener.

- Reglas de codificación: SonarQube incluye un conjunto de reglas de codificación estándar que ayudan a identificar posibles problemas y malas prácticas en el código. Estas reglas cubren aspectos como convenciones de nomenclatura, manejo de excepciones, uso de API obsoletas, entre otros.

- Cumplimiento de estándares de seguridad: SonarQube también evalúa el código en busca de posibles

vulnerabilidades de seguridad, como vulnerabilidades conocidas, uso de funciones criptográficas débiles o configuraciones inseguras.

- Cantidad de código: Además de las métricas de calidad específicas, SonarQube también proporciona información sobre el tamaño del código, como el número de líneas de código, la cantidad de clases y métodos, etc.

Estas métricas proporcionan una visión integral de la calidad del código y ayudan a los equipos de desarrollo a identificar áreas de mejora y mantener estándares de calidad en los proyectos.

Es importante tener en cuenta que estas métricas se personalizan y ajustan según las necesidades del lenguaje de programación a usar y se establecen de igual forma para la incorporación de nuevo código.

- Se deben implementar las pruebas de integración que sean necesarias a los sistemas con la finalidad de asegurar que en las aplicaciones desarrolladas se ha implementado los requisitos de seguridad definidos antes de comenzar el desarrollo. Su evidencia es la implementación en la herramienta que se ese usando para el proyecto, de la práctica de integración y despliegue continuo CI/CD que se haya planeado para cada uno de los requerimientos o sistemas de información.

- Se deben realizar pruebas funcionales a las aplicaciones desarrolladas en conjunto con el/la dueño del sistema de información, haciendo uso del formato de casos de prueba, realizando la respectiva documentación a las pruebas realizadas y aprobando los pasos a producción.
- Realizar el mergen de la rama de desarrollo a pruebas una vez finalizado el proceso de desarrollo y realizar el merge de la rama de pruebas a producción una vez validado y aprobado el proceso de pruebas integrales.

- Las herramientas de desarrollo de software utilizadas deben estar licenciadas por la Secretaría Distrital de Movilidad o el contratista según corresponda y deberá estar en custodia de la OTIC únicamente.

- Se debe hacer uso de un software de control de versiones el cual salvaguarde la integridad del código fuente desarrollado, la administración de los repositorios debe estar a cargo de una persona funcionaria y/o cuenta de la persona oficial de la SDM designada por la OTIC.

- Se debe llevar un control de la versión del producto software, el cual se identifica con cuatro dígitos W.X.Y.Z iniciando en 0.0.0.0.

  El digito Z estará asociado a liberación de integración de producto a cargo del Líder Técnico en el ambiente de desarrollo, cualquier cambio del producto después de sesión de integración implica incremento en el dígito Z.

  El digito Y estará asociado a liberación para Pruebas Funcionales del equipo interno o del proveedor. Todo cambio por ajustes de incidencias a nivel interno y que sea para pruebas funcionales, implica un incremento en el dígito Y.

  Los dígitos W.X estarán asociados a los ambientes UAT y Producción. Cualquier cambio a un producto entregado para pruebas de usuario implica un incremento en el dígito X.

  Si es una entrega final a ambiente de producción, implica un incremento en el dígito W.

  Cada vez que se incremente cualquier digito se deben reiniciar los dígitos a la derecha, es decir, iniciar desde cero nuevamente.

  El responsable de asignar la versión del producto liberado es del Líder Técnico del proyecto.

  La versión del producto se visualizará en la página de autenticación del sitio.

- Mantener actualizada las versiones de lenguaje de programación, frameworks[9] y librerías usadas en los proyectos de acuerdo con la tabla de versiones seguras, lo anterior para solventar los problemas de seguridad del lenguaje de las versiones anteriores.

- Se debe identificar frameworks y/o librerías y de fuentes reconocidos con mantenimiento, actualización y desarrollo activo por parte de la comunidad y que sean ampliamente usados en la industria. Los frameworks y/o librerías de terceros reconocidos que incluyen elementos de seguridad son primordiales para minimizar errores de implementación en componentes en los que el desarrollador no tenga el conocimiento y dominio necesario. Es de gran importancia realizar un inventario de dichos frameworks y librerías para tener un control de las actualizaciones y prevenir vulnerabilidad.

2.10 LINEAMIENTOS PARA EL DESARROLLO SEGURO

La política de desarrollo seguro de la SDM comprende las reglas para el desarrollo de software y sistemas dentro de la organización. Para esto, se establecen los siguientes lineamientos:

- Se deberán utilizar técnicas de programación seguras tanto para los desarrollos nuevos como en las situaciones de reutilización de códigos donde es posible que no se conozcan las normas que se aplican al desarrollo o donde

no sean coherentes con las buenas prácticas actuales. Lo anterior, tanto para el desarrollo interno como externo.

- Se deben estandarizar, los criterios de seguridad y calidad, que serán considerados, durante cada fase del proceso de desarrollo de sistemas de información.

- En cuanto al desarrollo de sistemas por terceros, se deben celebrar contratos, con las empresas proveedoras, que contengan cláusulas, que resguarden la propiedad intelectual para la SDM, y, asimismo, aseguren, los niveles de confidencialidad de la información, en el proyecto respectivo y lo demás que indique el Manual de Políticas específicas de Seguridad de la Información PA04-M01.

- Se debe diferenciar, entre la persona encargada de celebrar y autorizar los contratos con terceros, de los que deben validar y verificar su cumplimiento.

Para el atributo de seguridad, la SDM establece características específicas no funcionales como lineamientos a cumplir especificado en el documento anexo PA04-M02-L01 Lineamientos de Código Seguro

En caso de que se requiera por razones de seguridad de datos, se debe aplicar la política de enmascaramiento de datos definida en el numeral 5.46 “Política de Enmascaramiento de Datos” del PA04-M01 Manual de Políticas Específicas de Seguridad de la Información.

2.11 ESPECIFICACIONES PROCESO DEL DESARROLLO

Las especificaciones del proceso del desarrollo de software se encuentran definidos en el anexo PA04-M02-F03 Especificaciones del Proceso de Desarrollo de Software, por medio del cual se describe las fases de requerimientos, planeación, desarrollo y pruebas, revisión y controles; y despliegue.

3. ESTÁNDARES DE PROGRAMACIÓN

3.1 DISEÑO DE BASE DE DATOS

En la construcción de un software, no sólo se debe configurar las rutinas del programa para máximo desempeño, sino se debe tener en cuenta el diseño físico y lógico del almacenamiento de los datos.

- Para un buen diseño de base de datos tener en cuenta:

- Proveer un mínimo de tiempo de búsqueda para localizar registros específicos
- Almacenar los datos en la manera más eficiente posible
- Hacer las actualizaciones lo más fácil posible.
- Construcción la suficientemente flexible para la inclusión de nuevas funciones requeridas por el programa.

- Objetivos del Diseño

- Eliminar datos redundantes
- Poder localizar registros individuales rápidamente
- Realizar mejoras a la base de datos fáciles de implementar
- Fácil mantenimiento de la base de datos.

- Actividades en el Diseño de la Base de Datos

La creación de un buen diseño de la base de datos involucra las siguientes actividades:

- Modelamiento de la aplicación
- Determinar los datos requeridos por la aplicación
- Organizar los datos en tablas
- Establecer relaciones entre las tablas
- Usar índices y validaciones para los datos
- Crear y almacene cualquier búsqueda necesaria para la aplicación
- Revisar el diseño de la base de datos.

- Organización de los Datos

Uno de los aspectos importantes del diseño, es determinar cómo serán organizados los datos en la base de datos. Se deben organizar de manera que la información sea fácil de recuperar y realizar mantenimiento. Dentro de la base de datos, los datos son almacenados en una o más tablas. Se puede mantener un manejo eficiente de los datos almacenándolos en múltiples tablas y estableciendo relaciones entre ellas.

La normalización de datos es el proceso de eliminar datos redundantes dentro de una base de datos. Tomando la normalización en su concepto extremo, cada pieza de información en la base de datos debería aparecer una única

vez, aunque esto no es siempre practico.

Una manera de manejar la normalización es lo que se conoce como tablas hijas. Las tablas hijas son tablas en las que todas las entradas comparten información común que es almacenada en otras tablas. Las tablas que contienen la información común se denominan tablas padres.

Se debe utilizar estos lineamientos:

- Determinar un tópico para cada tabla, y asegurar que todos los datos en la tabla se relacionan con el tópico
- Si un número de registros en la tabla contiene campos intencionalmente en blanco, dividir la tabla en dos tablas similares
- Si la información se repite en un número considerable de registros, mover la información a otra tabla y establecer relaciones entre las tablas
- Campos repetidos indican la necesidad de tablas hijas.
- Usar vistas para reducir el volumen de datos e incrementar la veracidad de los datos.
- No almacenar información en tablas si esta puede ser calculada de otras tablas.

- Uso de Índices

Cuando la información es ingresada a una tabla, los registros son almacenados en el orden en que fueron adicionados. Este es el orden físico de los datos. Sin embargo, se requiere un orden diferente al orden de entrada de los datos, este se define como orden lógico.

Un índice provee un método para mostrar la información de una tabla en un orden específico. Un índice es una tabla especial que contiene una llave (usualmente derivada de los valores de uno o más campos) para la búsqueda de datos en la tabla. El índice contiene un puntero que le dice al motor de la base de datos donde se encuentra un registro específico. Este tipo de índice es similar al índice de un libro.

La estructura de un índice permite localizar rápidamente los datos para recuperarlos. Si se tiene una tabla indexada alfabéticamente por el nombre, se puede encontrar un registro por un nombre específico rápidamente utilizando el índice.

Una tabla puede tener diferentes índices asociados dependiendo de la organización de la tabla. El tipo más común de índices son los índices de una sola llave, los cuales se basan en el valor de un solo campo de la tabla. Por ejemplo, la cédula, el NIT. Si se necesita más detalle de acuerdo a más campos de la tabla, se usan los índices de múltiple llave, que se basan en los valores de dos a más campos de la tabla.

- Tablas

Las tablas son objetos de la base de datos que contienen todos sus datos. Una definición de tabla consiste en una colección de columnas de la misma manera en que una base de datos es una colección de tablas. Para poder almacenar datos en una base de datos, primero es necesario comprender cómo se crean, modifican y mantienen las tablas de la base de datos. Esto incluye tareas como definir claves y agregar o eliminar columnas de tablas.

- Índices

Son elementos que se utilizan para mejorar el rendimiento de la aplicación de bases de datos, aumentan la velocidad de las consultas. Los índices se colocan sobre un conjunto de columnas en una determinada tabla, con el fin de realizar consultas ágiles.

Existen varios tipos de índices, dependiendo su finalidad:

- Únicos – Sin Duplicados
- De llave Primaria
- Con Duplicados.

- Llaves Primarias

Son restricciones y/o índices que identifican de manera única un registro en una tabla. La llave primaria está conformada por una o más columnas de una tabla. El ejemplo clásico de una llave primaria es el número de identificación en una tabla de clientes.

- Llaves Foráneas – Relaciones

Las diferentes llaves primarias e índices de una tabla pueden relacionar o estar relacionada en otra tabla, es decir que existen restricciones que hacen posible que dos tablas se puedan cruzar con efectividad, obtener datos con integridad al utilizar estas restricciones e índices.

- Desencadenadores

Objetos de la base de datos que se activan cuando una tabla los contiene, dependiendo de la acción que se ejecute, es decir existen varios tipos de desencadenadores:

- De inserción. Se activan cuando se ingresa un nuevo registro en la base de datos.
- De Borrado. Ejecuta el código específico cuando se eliminan datos de la base de datos.
- De Actualización. Este tipo de desencadenador puede apuntar a la actualización de cada una de las columnas de la tabla o a una columna específica para ser activado.

- Procedimientos Almacenados

Conjunto de instrucciones que pueden modificar o generar un resultado sobre los objetos propietarios de la base de datos. Los procedimientos almacenados (SP) pueden contener diferentes tipos de parámetros, cada uno con diferentes características:

Parámetros de Entrada: Variables que son valoradas desde otro procedimiento o aplicativo externo, pero que no son modificables en el mismo procedimiento.

Parámetros de Salida: Variables definidas en el encabezado del procedimiento, toman valor únicamente en el mismo procedimiento, para ser utilizadas en otros procedimientos o aplicativos externos.

Parámetros de Entrada – Salida: Son valoradas por otros procedimientos o aplicativos y a la vez pueden ser modificadas y externalizadas por el mismo procedimiento que las contiene.

- Vistas

Son consultas que se guardan en la base de datos y que son utilizadas por otros objetos. Su utilización en la generación de datos es similar a la de una tabla en la base de datos.

- Funciones

Tiene la misma funcionalidad de un procedimiento almacenado, con la salvedad que puede ser utilizado en una vista, pues devuelve un tipo de dato específico.

3.2 ESTÁNDARES PARA BASES DE DATOS

TABLAS	Las tablas se nombrarán de acuerdo con la función que desempeñe en la base de datos. La estructura es la siguiente: *XXX_NOMBRE_TABLA *XX: Nombre del Proyecto *X: Tipo de Tabla *Nombre: Finalidad de la tabla. La restricción de longitud del nombre debe ser máximo de 50 caracteres, sobra decir que debe ser muy diciente el nombre de la tabla, y en lo posible en las herramientas que contiene comentarios utilizarlos. Los tipos de tabla son los siguientes: *B: Tablas Básicas *H: Tablas Históricas *I: Tablas de Interfaces *T: Talas temporales Ej: MCH_CLIENTE Los comentarios en las tablas y en los campos son obligatorios. Los motores de bases de datos tienen opciones o comandos para comentar los objetos de la base de datos.

INDICES	Se rigen de acuerdo a la siguiente estructura: IDX_NOMBRE_INDICE El nombre del índice debe contener el nombre de la tabla a la cual hace referencia, y utilizar los comentarios asociados, si los tiene la aplicación para describir su funcionamiento. Si existe más de un índice asociado, enumerarlo en orden secuencial al último creado Ej: IDX_MCB_PARAMETRO.
LLAVES PRIMARIAS	Deben tener la siguiente estructura PK_NOMBRE_TABLA La llave primaria debe contener el nombre de la tabla dentro de su estructura, la combinación pk determina que la restricción o el índice cumplen con el condicionamiento de unicidad. Ej: PK_MCB_PARAMETRO
LLAVES FORÁNEAS	Las relaciones de integridad deben denominarse de la siguiente forma: FK_TABLAREFERIDA_TABLAREFERENCIA El nombre de la referencia debe contener los nombres de las dos tablas que la conforman. Se hace referencia a la tabla referida como aquella que tiene la restricción, y la tabla de referencia como aquella de donde se toman los datos que van a ser comparados. Ej: FK_MCH_DETALLE_MCH_CLIENTE01. El consecutivo es para diferenciar la coexistencia de más de una relación entre las dos tablas.
STORED PROCEDURE o PROCEDIMIENTOS ALMACENADOS	Su estructura es la siguiente: XXSP_NOMBRE_PROCEDIMIENTO XX: Iniciales del proyecto El nombre del procedimiento debe ser acorde con la función que desempeñe. Ej: MCSP_CALCULARFORMULA

VISTAS	La denominación de la vista desde ser: XXV_NOMBRE_VISTA XX: Iniciales del proyecto. NombreVista: Debe ser claro, describiendo la función que realiza. Ej: MCV_CLIENTE_ACTIVO La nomenclatura también aplica para el objeto VISTA MATERIALIZADA de ORACLE
FUNCIONES	Están compuestas por: XXSF_NOMBRE_FUNCIO N XX: Iniciales del Proyecto Nombre Funcion: Debe ser claro, describiendo las operaciones que ejecuta. Ej: MCSF_NOMBRE_CLIENTE.
RESTRICCIONES DE CHEQUEO	En caso que se requieran incluir restricciones de chequeo, estas deben contener el nombre de la tabla, seguido de “CHK” seguido de al menos el nombre de una de las columnas que lo componen. Ej: CHK_MCH_CLIENTE_TIPO_ENTIDAD

PAQUETES	Para el caso de las bases de datos que utilicen paquetes como ORACLE, se debe utilizar el siguiente estándar XXPQ_NOMBREPAQUETE XX: Perefijo de la base de datos PQ: Identifica que se trata de un paquete Ejemplo: MCPQ_GENERAR_CONSULTA Los procedimientos y/o funciones dentro de los paquetes se nombran con el prefijo SP o SF según sea el caso. Ejemplo: MCPQ_RETENCION.SP_CALCULAR
COMENTARIOS	Después del encabezado del procedimiento o función se debe registrar la información correspondiente al objetivo del procedimiento, fecha de creación, parámetros de entrada y salida, así: -- DESCRIPCION: Este procedimiento calcula los totales de las notas credito activas y las registra en TCAR_CLIENTES -- PARAMETROS: Descripción clara de los parametros de entrada y salida. -- MODIFICACIONES: JUAN PEREZ FECHA CREACION: 01/01/2023 FINALIDAD DE LA MODIFICACION Se deben incluir todos los comentarios y explicaciones que se consideren para hacer el código claro de entender, estos de deben realizar empleando doble guión antes del comentario.

TRIGGERS	Debe estructurarse de la siguiente forma: TRG_NOMBRE_TABLA TRG: Referido a un desencadenador NombreTabla: Es el nombre sobre la tabla en que actúa el desencadenador. Ej: TRG_MCH_FACTURA01 El consecutivo es para diferenciar la coexistencia de más de un desencadenador del mismo tipo en la misma tabla. En el cuerpo del trigger debe incluirse un área para los comentarios o descripción de la finalidad del mismo.

SECUENCIAS	Las secuencias aplican para base de datos ORACLE, y se usan como generadores de numeros secuenciales. La estructura es XXSQ_NOMBRE_SECUENCIA Ejm: MCSQ_CONSECUTIVO_DETALLE
VARIABLES DENTRO DE PROCEDIMIENTOS Y FUNCIONES	Independientemente del motor de base de datos, las variables utilizadas dentro de la funcionalidad de los procedimientos de nombran de la siguiente manera. Para los nombres de las variables utilizadas dentro del cuerpo del procedimiento. V_NOMBRE_VARIABLE Ejms: V_CEDULA_CLIENTE NVARCHAR (15) SQLSERVER V_CEDULA_CLIENTE VARCHAR2(15) ORACLE Para los nombres de los parámetros que son recibidos en los procedimientos o funciones P_NOMBRE_PARAMETRO Ejms: P_CODIGO_CUIDAD NUMERIC (5) SQLSERVER P_CODIGO_CIUDAD NUMBER (5) ORACLE Para la definición de los nombres de los cursores. C_NOMBRE_CURSOR Ejms: DECLARE C_NORMAS CURSOR FOR SELECT …. SQLSERVER
RECOMENDACIONES	Utilizar la directriz %ROWTYPE, para definir las variables del mismo tipo del campo definido en la tabla. Ejm: V_NOMBRE_CLIENTE%ROWTYPE (EXCLUSIVO DE ORACLE) Utilizar objetos de tipo DOMINIO, que identifican tipos de datos definidos por el usuario. Aplicable para SQLSERVER

3.3. ESTÁNDARES DE PROGRAMACIÓN

El documento pretende continuar con estándares internacionalmente aceptados y desarrollados en los diferentes entornos de desarrollo (IDE's)

Se tendrán en cuenta las recomendaciones emitidas por los fabricantes, instituciones especializadas o las comunidades de desarrollo, enunciadas a continuación.

Java y JEE.

https://wiki.sei.cmu.edu/confluence/display/java/SEI+CERT+Oracle+Coding+Standard+for+Java

https://wiki.sei.cmu.edu/confluence/display/java/Java+Coding+Guidelines
https://wiki.sei.cmu.edu/confluence/display/java/Java+Rules

.Net.

https://msdn.microsoft.com/es-es/library/ms184412(v=vs.110).aspx

4. PLATAFORMA TECNOLÓGICA

Los sistemas deberán estar desarrollados sobre plataforma WEB utilizando los servicios tecnológicos, tecnologías, herramientas y software base disponibles y soportados en la infraestructura tecnológica de la Secretaría Distrital de Movilidad.

5. ESTRUCTURACIÓN DE SOLUCIONES Y PROYECTOS

Las siguientes son las reglas aplicables a los desarrollos nuevos y mantenimientos sobre componentes de software:

- Nombres de los proyectos

Para definir los nombres de los proyectos que hacen parte de una solución se debe tener en cuenta los siguientes ítems:

Se denomina como “Solución principal”, a un conjunto ordenado de proyectos que son integrados y que funcionan para entregar un sistema de información; la solución puede contener múltiples proyectos, servicios, bibliotecas, componentes, páginas WEB, archivos de configuración, sitios WEB, entre otros.

La Solución principal debe permitir que cada uno de los componentes que hacen parte de él tenga independencia funcional de los demás componentes del sistema, por lo que para facilitar el desarrollo independiente se crea una solución que contiene los proyectos que conformen el componente, el nombramiento de estas soluciones debe ser de la siguiente forma: XXX.Componente.

XXX: Corresponde a la abreviatura del sistema de información que se está desarrollado, por ejemplo: SAP.Seguridad.

Para los casos de capas de desarrollo o componentes de nivel jerárquico más alto, por ejemplo: capa de presentación, capa de servicios, capa de negocio, entre otras, el nombramiento considera el nombre lógico de la capa a utilizar, de la siguiente forma: XXX.Capa.Componente

Para el sistema SAP, capa de presentación WEB y componente de seguridad, quedará de la siguiente forma: SAP.Web.Seguridad

- Estructura

Los proyectos que se construyen en la solución principal, deben cumplir con la siguiente estructura, dependiendo de la funcionalidad que cumple y a que capa del sistema de información está afectando: XXX.Componente.CapaOFuncionalidad

Para el sistema SAP, componente de seguridad y capa de datos es: SAP.Seguridad.Datos

En los casos que dentro de una misma solución deban existir más de un proyecto que haga referencia a una capa del sistema, como la capa de presentación y entidades por hacer referencia a modelos de base de datos diferente, la estructura de estos proyectos debe incluir el nombre del modelo que se está afectando. XXX.Componente.CapaOFuncionalidad.Modelo

Para el sistema SAP, componente de WorkFlow, capa de datos, en la entidad de datos Factura, quedará de la siguiente forma: SAP.Workflow.Datos.Factura, o SAP.Workflow.Entidades. Factura

Para cada solución se recomienda la utilización de directorios para cada tipo de proyecto. Un ejemplo de esto es que se pueden tener varios proyectos en una solución con tecnologías diferentes por ejemplo proyectos web ASP.NET, proyectos de WinForms así como WPF, WCF, Web Services etc. De esta manera se pueden tener varios proyectos y varios folders para cada proyecto enmarcado en la tecnología que utiliza.

- /Codigo
- /WinForms
- /Web
- /WCF
- /LibreriadeClases1
- /LibreriadeClases2
- SolucionPrincipal.sln

---

[1] *Conjunto de agentes, códigos y procesos que interactúan coordinadamente entre sí.*

[2] *Estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo.*

[3] *Utilizados para presentar valores que no cambian con el tiempo y que no tienen un identificador único*

[4] *Interfaz entre sistemas que use HTTP para obtener datos o generar operaciones*

[5] *(JavaScript Object Notation) es un formato ligero de intercambio de datos*

[6] *Docker Desktop se utiliza para ejecutar, editar y administrar el desarrollo de contenedores. Kubernetes se utiliza para ejecutar aplicaciones de producción a gran escala.*

[7] *Modelo de programación que permite mapear las estructuras de una base de datos relacional*

[8] *Lenguaje gestor para el manejo de la información en las bases de datos relacionales*

[9] *Marco o esquema de trabajo generalmente utilizado por programadores*

ELABORÓ DEL PROCESO	REVISÓ DEL PROCESO	REVISÓ OFICINA ASESORA DE PLANEACIÓN INSTITUCIONAL	APROBÓ
Adriana Gutierrez Arismendi Profesional Especializado(a) 222-19	Roger Alfonso Gonzalez Herrera Contratista	Lady Carolina Cardenas Perez Contratista	Edgar Eduardo Romero Bohorquez Jefe(a) de Oficina

Lady Carolina Cardenas Perez @ 2024-11-19, 14:36:46