CONTENIDO

1. OBJETIVO

El objetivo principal del Manual es proporcionar un marco claro y consistente para el desarrollo de software seguro y de alta calidad en la Secretaría Distrital de Seguridad, Convivencia y Justicia. Al seguir estas directrices, se busca minimizar los riesgos y vulnerabilidades, asegurar la integridad de los sistemas de información y fomentar una cultura de mejora continua y colaboración en el equipo de desarrollo.

2. ALCANCE

Este Manual establece los estándares y prácticas para el desarrollo seguro de software en la Secretaría Distrital de Seguridad, Convivencia y Justicia, con el fin de mantener la calidad, seguridad y eficiencia en el desarrollo de aplicaciones.

3. AMBITO DE APLICACIÓN

El manual es aplicable a todos los proyectos de desarrollo de software de la Secretaría, independientemente del lenguaje de programación o las herramientas utilizadas.

4. NORMATIVIDAD ASOCIADA

La seguridad en el desarrollo de software es un aspecto fundamental para garantizar la seguridad de los datos y la integridad de las aplicaciones. Además, requiere la implementación de la Política de Seguridad y Privacidad de la Información1 y del Manual de Seguridad y Privacidad de la Información MA-GT-1 de la Secretaría Distrital de Seguridad, Justicia y Convivencia.

5. DOCUMENTOS ASOCIADOS

Manual de Seguridad y Privacidad de la Información MA-GT-1.

6. GLOSARIO

Autenticación: Proceso de verificar la identidad de un usuario o sistema.

Autorización: Proceso de determinar si un usuario tiene permiso para realizar una acción específica.

[1] https://scj.gov.co/sites/default/files/marco-legal/Resoluci%C3%B3n%200025%20del%2029%20de%20enero%20de%202021-Nueva%20Politica%20de%20Seguridad%20de%20la%20Informaci%C3%B3n%20DTSI-firmada.pdf

CamelCase: Convención de escritura en la que cada palabra dentro de una frase comienza con una letra mayúscula y no se utilizan espacios ni guiones bajos (por ejemplo, MiVariableEjemplo).
CI/CD (Integración Continua y Despliegue Continuo): Prácticas de desarrollo de software donde las modificaciones se integran y prueban automáticamente y las aplicaciones se despliegan de manera automatizada.
Contenedor: Unidad estandarizada de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable en diferentes entornos.
DevSecOps: Práctica de integrar la seguridad en todas las fases del ciclo de vida del desarrollo de software, desde el diseño hasta la entrega y operación.
Docker: Plataforma que permite desarrollar, enviar y ejecutar aplicaciones dentro de contenedores.
GitHub/GitLab: Plataformas de alojamiento de código fuente que permiten el control de versiones, la gestión de proyectos y la colaboración en el desarrollo de software.
HTTP (HyperText Transfer Protocol): Protocolo de comunicación utilizado en la web para la transferencia de documentos de hipertexto.
IDE (Entorno de Desarrollo Integrado): Aplicación que proporciona un conjunto de herramientas completas para el desarrollo de software, incluyendo un editor de código, un depurador y un compilador.
Inyección de Código: Vulnerabilidad de seguridad que permite a un atacante introducir código malicioso en un programa.
Kubernetes: Plataforma de orquestación de contenedores que automatiza el despliegue, la escala y el manejo de aplicaciones contenedorizadas.
Logging: Proceso de registrar información sobre eventos y actividades en un sistema para su análisis posterior.
MFA (Autenticación Multifactor): Método de autenticación que requiere dos o más formas de verificación para otorgar acceso a un recurso.
OWASP (Open Web Application Security Project): Fundación que se dedica a mejorar la seguridad del software mediante la creación de estándares, herramientas y metodologías abiertas.

Pipeline: Conjunto de procesos automatizados para la compilación, prueba y despliegue de aplicaciones de software.

RBAC (Control de Acceso Basado en Roles): Método de restringir el acceso a recursos del sistema basado en los roles de los usuarios.

Repositorio: Almacenamiento centralizado para gestionar y controlar las versiones del código fuente de un proyecto de software.

REST (Representational State Transfer): Estilo arquitectónico para la comunicación entre sistemas en la web.

SSL/TLS (Secure Sockets Layer/Transport Layer Security): Protocolos criptográficos diseñados para proporcionar comunicaciones seguras a través de una red.

Token: Pequeño fragmento de datos utilizado para autenticar y autorizar acciones en un sistema.

XSS (Cross-Site Scripting): Tipo de vulnerabilidad de seguridad en aplicaciones web que permite a un atacante inyectar scripts maliciosos en el contenido que se envía a un usuario.

7. DESARROLLO

7.1 ESTÁNDARES DE CODIFICACIÓN

Esta sección detalla las convenciones y buenas prácticas que deben seguirse al escribir código en el ciclo de vida de desarrollo de software. El objetivo es garantizar la calidad, consistencia y el mantenimiento del código fuente en el repositorio privado de código fuente distribuido Git de la Entidad[2], el cual requiere de una VPN y los permisos de acceso para que el desarrollador realice las actualizaciones en los sistemas de información asignados.

Con su implementación se obtendrían proyectos de software de alta calidad y se facilitará la colaboración y reutilización de componentes entre los desarrolladores.

Nomenclatura

[2] http://172.21.20.84/users/sign_in

- Paquetes:
  - Los paquetes en Java deben seguir la estructura jerárquica: co.gov.scj.[sistema].[clasificador].
  - Ejemplo: co.gov.scj.sidijus.util
- Nombres de Clases:
  - Sustantivos con la primera letra en mayúsculas (CamelCase).
  - Ejemplo: public class Ciudadano {…}
- Nombres de Interfaces:
  - Prefijo "I" para diferenciar de las clases que implementan.
  - Ejemplo: public interface IAgendaService {…}
- Métodos:
  - Verbos escritos en minúsculas, con CamelCase para métodos compuestos.
  - Ejemplo: public void eliminaAgenda(Agenda agenda) {…}
- Variables:
  - Minúsculas, con CamelCase para nombres compuestos.
  - Ejemplo: private Unidad unidad;
- Constantes:
  - En mayúsculas, separadas por guion bajo.
  - Ejemplo: public static final int LONGITUD_MAXIMA.
- Archivos:
  - Nombre claro y en UpperCamelCase.
  - Ejemplo: NombreValido.java

Organización de Archivos

- Comentario inicial sobre el archivo.
- Definición del paquete y clases importadas.
- Estructura de la clase o interfaz, documentada adecuadamente.

Convenciones de Codificación

- Sangrado: 4 espacios por unidad.
- Comentarios: Claros y descriptivos.
- Declaraciones: Evitar nombres ofuscados y declarar una variable por línea.

- Sentencias comunes y buenas prácticas: Uso de llaves {} para sentencias compuestas, manejo adecuado de espacios en blanco y paréntesis.

7.2 SEGURIDAD EN EL DESARROLLO

La seguridad en el desarrollo de software es un aspecto fundamental para garantizar la seguridad de los datos y la integridad de las aplicaciones. Además, requiere la implementación de la Política de Seguridad y Privacidad de la Información[3] y del Manual de Seguridad y Privacidad de la Información MA-GT-1 V4 [4] de la Secretaría Distrital de Seguridad, Justicia y Convivencia.

Controles a Nivel de Aplicación

Los controles a nivel de aplicación son medidas específicas que se implementan directamente en el código para prevenir vulnerabilidades y proteger la aplicación de ataques, minimizando los riesgos de vulnerabilidades.

- Control y Seguridad de Contraseñas: Utilizar cifrado, controles de autenticación, y notificaciones de cambios.
- Manejo de Sesiones de Usuarios: Identificador de sesión del lado del servidor, funcionalidad de terminar sesión, y tiempo límite de sesiones.
- Autorización de Usuarios: Políticas de autenticación, limitación de intentos fallidos, y autenticación de dos factores.
- Manejo de Errores: Mensajes de error legibles y registros detallados en el sistema.
- Logging: Información completa y segura en los logs, sin datos sensibles.
- Auditoría: Registrar información importante y proteger el acceso a los logs.
- Manejo de Archivos: Permisos mínimos, protocolos seguros, y validación de nombres de archivos.

[3] https://scj.gov.co/sites/default/files/marco-legal/Resoluci%C3%B3n%200025%20del%2029%20de%20enero%20de%202021-Nueva%20Politica%20de%20Seguridad%20de%20la%20Informaci%C3%B3n%20DTSI-firmada.pdf

[4]

MA-GT-1	Manual de Seguridad y Privacidad de la Información

- Comunicación Segura: Cifrado para transmisión de datos sensibles y uso de certificados digitales.

Controles a Nivel del Proceso de Desarrollo de Software

El Procedimiento Ciclo de vida de desarrollo de Software PD-GT-17 de la Entidad permite definir un flujo técnico para el diseño, análisis, desarrollo, pruebas y puesta en funcionamiento mediante el procedimiento de gestión de cambios. Permitiendo la definición y el control de medidas de seguridad en el análisis de requerimientos y en las mesas técnicas, para su documentación de análisis de negocio y de propuesta de la solución. Así cómo en las reglas de validación de las historias de usuario.

- Repositorios de Código: Control centralizado y revisión de código fuente.
- Revisión de Código Fuente: Por desarrolladores senior o líderes técnicos.
- Cadena de Custodia: Verificación de artefactos antes del despliegue.
- Políticas de Desarrollo de Software Seguro: Basadas en OWASP.

Recomendaciones Técnicas

- Seguridad por defecto: Todas las aplicaciones deben diseñarse con seguridad como prioridad desde el inicio.
- Mínimo privilegio: Los componentes de la aplicación solo deben tener los permisos necesarios para realizar sus tareas.
- Validación de entradas: Toda entrada de usuario debe ser validada y controlada para prevenir inyecciones (SQL, XSS, etc.). Se deben definir validaciones, limitaciones y restricciones en las entradas de los sistemas de información:
- Validación del tipo de dato: Asegurarse de que los datos de entrada sean del tipo esperado.
- Limitar la longitud de las entradas: Establecer límites mínimos y máximos.
- Escapar caracteres especiales: Utilizar funciones de escape para neutralizar caracteres especiales que puedan ser utilizados para inyecciones.
- Validación de patrones: Emplear expresiones regulares para validar formatos de datos.

- Gestión de errores segura: Los errores deben manejarse de forma segura, evitando la divulgación de información sensible.
- Cifrado: La información sensible debe ser cifrada tanto en tránsito como en reposo.

Protección contra Ataques Comunes

- Protección contra inyección: Se debe realizar una validación exhaustiva de todas las entradas e implementar medidas para prevenir inyecciones SQL, XSS, etc.
- Protección contra CSRF: Implementar tokens CSRF para prevenir ataques de solicitud cruzada.
- Protección contra clickjacking: Implementar medidas para prevenir el secuestro de clics.
- Protección contra ataques de fuerza bruta: Implementar mecanismos de bloqueo de cuentas y detección de anomalías. El uso de un Recapcha en los inicios de sesión previene ataques de fuerza bruta y evita ataques de degeneración de Servicio.

7.3 ACTUALIZACIÓN DE LAS TECNOLOGÍAS Y SISTEMAS DE INFORMACIÓN

Introducción

La actualización de las tecnologías y los sistemas de información es un proceso continuo y esencial para mantener la seguridad de las aplicaciones y de los sistemas de información.

En esta sección se definen las mejores prácticas, recomendaciones y consideraciones para actualizar las tecnologías de los sistemas de información desarrollados, minimizando los riesgos y garantizando una transición controlada de las actualizaciones de los sistemas de información mediante tecnologías modernas y de vanguardia.

Análisis Técnico

El desarrollo de los sistemas de información de la DTSI ha estado diseñado para su funcionamiento y despliegue en el aprovisionamiento y arquitectura de la infraestructura bajo tecnología Oracle definida con un servidor de aplicaciones monolítico Weblogic 12c y JAVA 7-8 con sus tres ambientes en segmentos de red independientes.

Se ha realizado un escalamiento vertical de la infraestructura y servicios de bases de datos con su capacidad y procesamiento, pero se requiere un escalamiento horizontal, que permita

garantizar el funcionamiento y la actualización independiente de los sistemas de información, optimizando su administración y el uso de recursos de cómputo y procesamiento.

Se recomienda para nuevos desarrollos propios utilizar las últimas versiones de las tecnologías de desarrollo. Así mismo, se recomienda la actualización del servidor de aplicaciones Weblogic 12c definido en la Arquitectura de la Infraestructura monolítica actual, y realizar un proceso de migración a un servidor de aplicaciones distribuido que permita desacoplar los sistemas de información en tecnologías modernas como contenedores; los cuales permiten realizar migraciones y despliegues independientes de los sistemas de información, garantizando el funcionamiento y minimizando los riesgos a fallos generalizados en los sistemas de información misionales al compartir el mismo servidor.

Estado Actual:

Sistemas de Información	**Tecnología JAVA**		**Frameworks Primefaces, Bootfaces**		**Arquitectura de Software**		**Arquitectura de Infraestructura**		**Servidor de Aplicaciones**	
	Versi ónAct ual	Versión Actualiz ada	Versión Actual	Versión Actualiza da	Actual	Rec om end ada	Actual	Recom endada	Versió n Actual	Versión Actualizad a
Misionales (LICO, COPE, SIDIJUS, PROGRESS US, SIRPA, DELIVERY, APELACIO NES)	JDK 8	> JDK 17 JDK 21	Primefac es 8 Bootsfac es	Primefac es 13 Bootsfac es 1.5 2.0	MVC Micro-servici os Front-BackE nd	N/A	Servido r de aplicaci ones Mónolíti co	Servido r Distribu ido Conten edores	Weblo gic 12 c JDK 7	Servidor de Aplicacion es Distribuid o >JDK17 JDK 21

Objetivos de la Actualización

- Mitigar vulnerabilidades: Aplicar actualizaciones a las últimas versiones y aplicar parches de seguridad para corregir errores conocidos y prevenir ataques.

- Mejorar el rendimiento: Optimizar el código para aumentar la velocidad y capacidad de respuesta.
- Incorporar nuevas funcionalidades: Agregar características solicitadas por los funcionarios requeridos por las áreas funcionales.
- Automatizar: Utilizar herramientas de automatización para agilizar el proceso de actualización y reducir el riesgo de errores manuales.
- Pruebas continuas: Integrar pruebas automatizadas en el ciclo de desarrollo para garantizar la calidad del software.

Proceso de Actualización

- Evaluación de riesgos: Identificar los posibles impactos de la actualización, como interrupciones del servicio, pérdida de datos o degradación del rendimiento.
- Definición del alcance: Establecer claramente qué componentes del sistema se actualizarán y cuáles se mantendrán sin cambios.
- Establecimiento de un cronograma: Crear un plan detallado con fechas de inicio y fin para cada tarea.

7.4 CONTENEDORES Y ORQUESTACIÓN

Introducción

Esta sección se abordará la transición de una arquitectura monolítica a una arquitectura distribuida basada en contenedores. Se detallarán los beneficios, desafíos y pasos clave involucrados en este proceso, así como las tecnologías y herramientas más adecuadas para llevar a cabo esta transformación y actualización de los sistemas de información.

Beneficios de la Arquitectura Distribuida con Contenedores

- Escalabilidad: Permite escalar de forma independiente los componentes de los sistemas de información, optimizando el uso de recursos.
- Agilidad: Facilita el desarrollo, despliegue y actualización de los servicios de manera más rápida.

- Flexibilidad: Permite utilizar diferentes tecnologías y lenguajes de programación en distintos componentes.
- Resiliencia: Aislamiento de fallos y mayor tolerancia a errores.
- Reutilización: Fomenta la reutilización de componentes a través de contenedores.

Tecnologías y Herramientas

- Docker: Implementación y Buenas Prácticas
  - Creación de Dockerfiles: Instrucciones claras y mantenibilidad.
  - Buenas Prácticas: Mantener imágenes ligeras, minimizar capas, y escaneo de vulnerabilidades.
- Orquestación con Kubernetes
  - Componentes Principales: Pods, servicios, volúmenes, ConfigMaps, y Secrets.
  - Buenas Prácticas: Uso de namespaces, controladores de admisión, monitoreo y logging.
- Seguridad en Contenedores y Kubernetes
  - Seguridad en Contenedores: Mínimos privilegios y escaneo de imágenes.
  - Seguridad en Kubernetes: Configuración de RBAC, políticas de red, y uso de contenedores firmados.

7.5 AUTOMATIZACIÓN Y DESPLIEGUE

Integración y Despliegue Continuo (CI/CD)

- Pipelines de CI/CD: Automatización de construcción, prueba y despliegue.

Pruebas Automatizadas

- Tipos de Pruebas: Unitarias, de integración y de seguridad.

Integración de DevSecOps

- Prácticas de Seguridad: Integración de seguridad en todas las etapas del desarrollo.

7.6 REVISIÓN Y ACTUALIZACIÓN DE ESTÁNDARES

Procedimiento de Revisión

- Ciclo de Revisión: Cada 6 meses.
- Comité de Revisión: Responsabilidades y composición.

7.7 GESTIÓN DE PROYECTOS Y COLABORACIÓN

Herramientas de Gestión de Proyectos

- JIRA, Trello: Planificación y seguimiento.

Integración de Herramientas de Colaboración

- GitHub, GitLab: Colaboración en el desarrollo y control de versiones.

7.8 CAPACITACIÓN Y ACTUALIZACIÓN CONTINUA

Programas de Formación y Certificaciones

- Capacitación Continua: Programas y certificaciones para el equipo.

Talleres y Sesiones de Actualización

- Actualización de Conocimientos: Nuevas tecnologías y mejores prácticas.

Elaboró: Nelcy Patricia Casas Rodríguez – Contratista Dirección de Tecnologías y Sistemas de la Información
Jorge Andres Serrano Jaimes - Contratista Dirección de Tecnologías y Sistemas de la Información

Revisó: Armando Alfonso Leyton González – Contratista Dirección de Tecnologías y Sistemas de la Información
Diego Mauricio Usme - Contratista Dirección de Tecnologías y Sistemas de la Información

La información de aprobación de este documento podrá ser consultada en el sistema “Portal MIPG” - https://portalmipg.scj.gov.co