SERVIÇO PÚBLICO FEDERAL
MJSP
COORDENAÇÃO DE SOLUÇÕES DE TECNOLOGIA DA INFORMAÇÃO - CST/CGTEC/STI/SE/MJSP

ROTEIRO DE CODIFICAÇÃO

ANEXO IV DO TERMO DE REFERÊNCIA

Processo Administrativo nº 08006.000703/2025-56

Termo de Referência nº 11/2026 - MJSP - Instrumento de governança do processo de desenvolvimento de software (item 4.6.1, alínea "c", do Anexo I da Portaria SGD/MGI nº 750/2023, incluído pela Portaria SGD/MGI nº 4.777, de 16 de junho de 2026)

Histórico de Revisões

Data	Versão	Descrição	Autor
29/07/2026	1.0	Criação do documento, a partir do Template - Roteiro de Codificação v1.0 (SISP) e do Anexo XIV do TR 65/2023	Guilherme Borges
05/08/2026	1.1	Inclusão do item 3.7 (Codificação Sustentável) e do item 1.3, em atendimento ao item 19.14 do Parecer SIRT nº 3051/2026/MGI e aos itens 4.4.5 e 8.2.1, alínea “f”, do Anexo I da Portaria SGD/MGI nº 750/2023, na redação da Portaria SGD/MGI nº 4.777/2026; atualização das citações normativas e da referência ao TR nº 11/2026; nota sobre os dois patamares de tamanho mínimo de chave criptográfica reproduzidos do Template SISP (itens 4.10.4 e 4.10.7); renumeração de Anexo VII para Anexo IV, com ajuste da remissão ao RMQPC, que passou a Anexo V; formatação no padrão dos anexos do Termo de Referência	Guilherme Borges

1. INTRODUÇÃO

1.1. O presente Roteiro dispõe sobre as melhores práticas de codificação com vistas a assegurar uma codificação enxuta, limpa, clara e eficiente, observando as diretrizes de codificação segura publicadas pela Secretaria de Governo Digital.

1.2. Este Roteiro é instrumento de governança do processo de desenvolvimento de software do MJSP, nos termos do item 4.6.1, alínea "c", do Anexo I da Portaria SGD/MGI nº 750/2023, incluído pela Portaria SGD/MGI nº 4.777, de 16 de junho de 2026, e deve ser observado em conjunto com os requisitos mínimos de qualidade e padronização dos aspectos técnicos da codificação (Anexo V - RMQPC) e com o processo de desenvolvimento de software do órgão (Anexo II - Metodologia de Desenvolvimento de Software).

1.3. Nos termos do item 4.4.5 do Anexo I da Portaria SGD/MGI nº 750/2023, com a redação dada pela Portaria SGD/MGI nº 4.777, de 16 de junho de 2026, as diretrizes deste Roteiro contemplam requisitos mínimos de qualidade, segurança, sustentabilidade ambiental e padronização dos aspectos técnicos da codificação.

2. DIRETRIZES GERAIS

2.1. Além das diretrizes constantes deste Roteiro, deve-se adotar, no que couber, as orientações constantes dos seguintes documentos publicados pela Secretaria de Governo Digital:

a) Guia de Requisitos e Obrigações quanto a Privacidade e à Segurança da Informação, disponível em: https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/ppsi/guia_requisitos_obrigacoes.pdf

b) Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicações Web, disponível em: https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/ppsi/guia_requisitos_minimos_web.pdf

c) Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para APIs, disponível em: https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/ppsi/guia_requisitos_minimos_apis.pdf

d) Guia de Requisitos Mínimos de Privacidade e Segurança da Informação para Aplicativos Móveis, disponível em: https://www.gov.br/governodigital/pt-br/privacidade-e-seguranca/ppsi/guia_requisitos_minimos_apps.pdf

3. ESCRITA DE CÓDIGO

3.1 Nomes Significativos

3.1.1. Os nomes utilizados para variáveis, funções, classes e métodos devem ser autoexplicativos, ou seja, devem responder às seguintes perguntas: "Por que ele existe?", "o que ele faz?" e "como é usado?".

3.1.2. Não se deve incluir no código informações inverídicas ou falsas para mascarar o código ou dificultar sua manutenção.

3.1.3. Os nomes devem ser pronunciáveis e ter significado adequado ao contexto em língua portuguesa.

3.1.4. Evite codificações ou encriptações em nomes de variáveis, classes, funções e métodos.

3.1.5. Os nomes devem ser de fácil entendimento associado ao contexto do domínio do negócio.

3.2 Funções e Classes

3.2.1. As funções e classes devem ter o mínimo de linhas possível, mantendo-se a clareza e implementando o código necessário para traduzir o objetivo esperado da função ou da classe.

3.2.2. As funções e classes devem ter objetivos específicos, possuindo um único nível de abstração (devem ter apenas uma responsabilidade ou objetivo).

3.2.3. A escrita do código deve seguir uma narrativa top-down.

3.2.4. As funções e classes não devem possuir efeitos escondidos ou colaterais, ou seja, não devem produzir efeitos distintos daqueles que se propõem a realizar.

3.2.5. Deve-se evitar níveis de herança exagerados, superior a 4 (quatro).

3.3 Comentários

3.3.1. O código útil deve ser autoexplicativo, limpo e eficiente, devendo possuir o mínimo de comentários possível.

3.3.2. Evite comentários redundantes ou óbvios, que descrevem o que o código faz sem acrescentar nenhuma informação útil.

3.4 Estruturas

3.4.1. Não se deve criar estruturas de código redundantes ou sem efetiva geração de valor à finalidade pretendida.

3.4.2. Não se deve replicar linhas de código ou estruturas tornando-as redundantes.

3.4.3. Deve-se evitar duplicar ou replicar funções ou classes que possuam a mesma responsabilidade ou objetivo.

3.5 Identação

3.5.1. Deve-se adotar a identação adequada para melhor entendimento e clareza do código, evitando-se identações exageradas ou nenhuma identação.

3.6 Complexidade Ciclomática

3.6.1. Deve-se adotar a menor complexidade ciclomática possível por algoritmo.

3.6.2. Deve-se evitar o aninhamento excessivo de instruções de condição ("if" dentro de "if").

3.6.3. Deve-se evitar o exagero de condições ("case") dentro de instruções de escolha ("switch").

3.6.4. Deve-se remover construções desnecessárias, isto é, aquelas que podem ser removidas sem alterar a semântica e a finalidade do código.

3.6.5. A complexidade ciclomática do código deve ser igual ou inferior a 10 por algoritmo, salvo se aprovado pela CONTRATANTE.

Nota: o limite de 10 acima é uma diretriz geral de desenho por algoritmo. Os limites de aferição operacional por arquivo, classe e função, utilizados no cálculo do IQC via ferramenta de análise estática, são os definidos no Anexo V - RMQPC, que prevalecem para fins de glosa.

3.7 Codificação Sustentável

3.7.1. A codificação deve observar a eficiência no uso de recursos computacionais, de modo a reduzir processamento, armazenamento, tráfego de dados e consumo energético desnecessários, nos termos dos itens 4.4.5 e 8.2.1, alínea “f”, do Anexo I da Portaria SGD/MGI nº 750/2023, com a redação dada pela Portaria SGD/MGI nº 4.777, de 16 de junho de 2026.

3.7.2. Devem-se otimizar algoritmos, consultas e estruturas de dados, evitando processamento redundante, laços ineficientes e o padrão de consultas N+1 a bancos de dados.

3.7.3. Devem-se utilizar racionalmente memória, conexões, threads e demais recursos, com liberação adequada após o uso e emprego de cache e paginação quando aplicável.

3.7.4. Deve-se minimizar o volume de dados trafegados, mediante compressão, respostas parciais e filtragem no servidor, e o armazenamento desnecessário, mediante expurgo e compactação de logs e arquivos temporários.

3.7.5. Devem-se dimensionar de forma eficiente as imagens de contêiner e os artefatos de implantação, removendo dependências não utilizadas.

3.7.6. O atendimento a este item integra a Definição de Pronto da sprint (Tabela 3.2 do Termo de Referência) e é aferido, sempre que possível, por ferramenta automatizada de análise de código.

4. DIRETRIZES DE SEGURANÇA DA INFORMAÇÃO

4.1 Armazenamento de Dados

4.1.1. Nos casos de armazenamento de dados abertos (dados públicos), não se deve utilizar meio de armazenamento que não possua acesso para escrita restrito por senha.

4.1.2. Nos casos de armazenamento de dados fechados (acesso restrito a grupo específico de pessoas), não se deve utilizar meio de armazenamento que não possua acesso para leitura e escrita restrito por senha.

4.1.3. Não se deve disponibilizar a aplicações acesso a banco de dados utilizando login de usuário com permissões de root.

4.1.4. Não se deve utilizar armazenamento de senhas em código-fonte.

4.1.5. Não se deve utilizar as mesmas senhas para ambientes de desenvolvimento ou homologação e produção.

4.1.6. Deve-se armazenar de forma segura os dados de usuários e os sistemas que utilizam cada senha fornecida.

4.2 Gerenciamento de Acessos e Permissões de Usuários

4.2.1. Não se deve armazenar senhas em texto plano sem utilizar um algoritmo de hash seguro e salt.

4.2.2. Deve-se utilizar controle de usuário e senha nominais para determinar a identidade do usuário.

4.2.3. Deve-se utilizar HTTPS para controle de autenticação ao menos nas telas de login do sistema web.

4.3 Comunicação

4.3.1. Deve-se empregar canal de comunicação com controle de duplicação e perda de informações/mensagens.

4.3.2. Deve-se empregar canal de comunicação com controle de autenticação (ex.: HTTPS, certificados digitais de autoridades confiáveis, VPNs).

4.3.3. Deve-se empregar canal de comunicação que garanta confidencialidade dos dados transmitidos (ex.: HTTPS, VPNs).

4.4 Medidas contra Ataques

4.4.1. Deve-se prevenir ataques de injeção de SQL (SQL Injection), com medidas mínimas como: (a) não concatenar parâmetros textuais de origem não segura em SQLs; (b) restringir permissões de acesso ao banco de dados para o usuário da aplicação; (c) utilizar prepared statements sempre que possível; (d) tratar com escape ou codificação hexadecimal consultas que não possam ser parametrizadas.

4.4.2. Deve-se prevenir ataques de injeção de HTML e Javascript.

4.4.3. Deve-se prevenir ataques do tipo cross-site scripting (XSS).

4.4.4. Deve-se prevenir ataques de quebra de autenticação e gerenciamento de sessão (Broken Authentication and Session Management).

4.5 Auditoria, Rastreamento e Logs

4.5.1. Devem ser registrados eventos como: login/logout; acesso a telas ou seções restritas; acesso a informações sigilosas (ex.: processos em segredo de justiça, dados pessoais ou bancários); inclusão, alteração ou exclusão de registros; alteração de perfil de acesso; execução de jobs e tarefas automatizadas.

4.5.2. Devem ser auditáveis, para cada evento monitorado: data e hora; usuário; IP; identificador de sessão; tela de origem; identificador de instância (sistemas em cluster); tipo de operação, tabela, ID do registro e valores antes/depois (para inclusão/alteração/exclusão); parâmetros informados pelo usuário (sem registrar dados sensíveis como senhas); tempo de resposta; resultado de jobs automatizados.

4.5.3. A captura pode ocorrer via triggers no banco, via aplicação, ou via filtro de interceptação de requisições (ex.: em Java).

4.5.4. O documento de especificação de requisitos do sistema deve definir quais informações serão registradas e onde serão armazenados os dados de auditoria.

4.6 Cópias de Segurança (Backups)

4.6.1. O plano de projeto deve prever a necessidade e a responsabilidade pela realização de backups do banco de dados e do código-fonte, bem como as políticas de acesso a esse backup.

4.6.2. Deve-se definir procedimento estruturado para restauração de backups.

4.6.3. Devem-se definir e capacitar responsáveis pela recuperação dos backups.

4.7 Testes

4.7.1. Devem-se realizar testes manuais de segurança antes de cada versão que modifique a estrutura do software (login, serviços não autenticados, novos formulários, etc.).

4.7.2. Deve-se garantir, por testes automatizados, que serviços e dados sigilosos estão protegidos e disponíveis apenas aos usuários autorizados.

4.7.3. Deve-se elaborar política de testes (automatizados ou não) quanto à não vulnerabilidade aos principais ataques conhecidos.

4.7.4. Devem-se definir cenários de teste para os requisitos não funcionais, preferencialmente por equipe de testes distinta da de desenvolvimento.

4.7.5. Devem-se definir cenários de teste, principalmente de segurança, para atualizações de arquitetura (servidores, banco de dados, browsers, sistema operacional, etc.).

4.8 Medidas contra Falhas de Segurança

4.8.1. Deve haver procedimento planejado para indisponibilização imediata do sistema e manutenção corretiva.

4.8.2. Deve-se definir política de acompanhamento pós-correção de falhas de segurança.

4.9 Instalação, Configuração e Gerenciamento de Ambientes

4.9.1. Deve-se utilizar controle de versão distribuído (ex.: Git).

4.9.2. Devem-se utilizar bancos de dados distintos por ambiente (desenvolvimento, produção).

4.9.3. Devem-se utilizar servidores de aplicação/web distintos por ambiente.

4.9.4. O acesso a ambientes de desenvolvimento/testes/homologação deve ser restrito à equipe do projeto e interessados (stakeholders).

4.9.5. Devem-se realizar testes periódicos de segurança desses ambientes.

4.10 Proteção de Dados Sensíveis

4.10.1. O método criptográfico deve seguir o princípio de Kerckhoffs (método e parâmetros públicos, só a chave em sigilo).

4.10.2. Não se deve utilizar cifrador cuja quebra seja mais fácil que força bruta por tentativa e erro.

4.10.3. Não se deve utilizar o modo de cifrador ECB (electronic codebook) ou menos seguro.

4.10.4. Chave mínima de 128 bits (simétrico) ou 1024 bits (assimétrico) - patamar mínimo absoluto.

4.10.5. Não se deve utilizar função de hash sem salt.

4.10.6. São vedados algoritmos obsoletos: MD5, SHA1, DES/3DES, RC2, RC4, MD4.

4.10.7. Chave mínima de 192 bits (simétrico) ou 2048 bits (assimétrico).

4.10.8. Não se devem distribuir chaves criptográficas sem infraestrutura de chave pública (cifrador assimétrico).

Nota: os itens 4.10.4 e 4.10.7 reproduzem o Template SISP v1.0, que enuncia dois patamares mínimos de tamanho de chave. Para fins contratuais prevalece o mais restritivo, do item 4.10.7: 192 bits para cifrador simétrico e 2.048 bits para cifrador assimétrico.

4.11 Geração de Senhas

4.11.1. Mínimo de 8 caracteres.

4.11.2. Vedado usar apenas um tipo de caractere (letras, dígitos, símbolos).

4.11.3. Vedado usar palavras comuns (nome de animal de estimação, familiar, data de aniversário, feriado, time esportivo, "senha"/"password").

4.11.4. Periodicidade de troca não superior a 1 ano.

4.11.5. Vedado armazenar senha em claro.

4.11.6. Deve-se armazenar ao menos o hash com salt.

4.11.7. Vedado transmitir a senha em canal sem proteção.

4.11.8. Método de conferência deve ser ao menos tão seguro quanto desafios baseados em hash.

4.11.9. Máximo de 5 tentativas de validação de senha por minuto.

4.11.10. A senha não deve ser exibida diretamente ao ser digitada; deve haver opção de mostrar/ocultar.

4.12 Segurança nas Fases de Especificação, Projeto e Design

4.12.1. O modelo de projeto deve contemplar, no mínimo: modelagem de ameaças; definição clara dos riscos de segurança; nível de severidade do comprometimento de dados sensíveis.

4.12.2. Não se deve omitir a definição de responsabilidades pela segurança de dados e como ela será verificada.

4.12.3. O cronograma do projeto deve prever pontos de verificação de segurança ao longo da construção.

4.13 Atualização de Softwares

4.13.1. Não se deve habilitar atualização automática de software ou componentes, sob pena de introdução indevida de falhas de segurança.

4.13.2. Deve-se utilizar apenas código de terceiros confiável.

Elaborado a partir do Template - Roteiro de Codificação, versão 1.0 (SISP/Governo Digital) e do conteúdo técnico já praticado pelo MJSP (herdado do Anexo XIV do TR 65/2023), sem divergências de mérito entre as duas fontes.