PLATAFORMA INOVAÇÃO PARA INDÚSTRIA

GUIA ORIENTATIVO PARA TECNOLOGIA DA INFORMAÇÃO

PREMISSAS TECNOLÓGICAS NORTEADORAS

CATEGORIA SAÚDE CONECTADA SESI

CHAMADA TÉCNICA REGIONAL:

LINHAS DE CUIDADOS PARA DOENÇAS CRÔNICAS NÃO TRANSMISSÍVEIS (DCNT)

Versão	Data	Itens Revisados
Inicial	02/09/2024	-

1. Objetivo

Orientar os Departamentos Regionais do SESI e/ou Centros de novação do SESI, interessados na submissão de projetos junto a Plataforma Inovação para a Indústria/Categoria Saúde Conectada SESI/Linhas de Cuidados para Doenças Crônicas Não Transmissíveis (DCNT).

2. Aplicação

Este documento aplica-se aos Gestores e Equipe Técnica dos Departamentos Regionais do SESI e/ou Centros de Inovação do SESI.

3. Especificações

3.1. Ambientes de Operação e Cloud:

- As soluções devem ser desenvolvidas utilizando linguagens de programação, bancos de dados e camadas de serviços que permitam a operação em ambientes de cloud computing, preferencialmente utilizando tecnologias de containerização.
- A linguagem de programação, frameworks e banco de dados utilizados devem estar em suas versões mais recentes com suporte ativo (LTS - Long Term Support), garantindo a estabilidade e a longevidade da solução.

3.2. Arquitetura Agnóstica:

- A solução deve ser projetada com uma arquitetura "agnóstica", ou seja, que se adapte a diferentes tipos de plataformas e hospedagem, evitando o "lock-in" e/ou dependência com provedores ou terceiros.
- É importante que a solução funcione por completo em uma única instância instalada, tanto da aplicação quanto da base de dados, mesmo que tenha requisitos funcionais de operação em ambientes multirregionais, multiempresa ou multifiliais.

3.3. Integração e Exposição de Dados:

- Deve-se utilizar web services baseados em protocolo REST para a exposição e integração de dados, garantindo compatibilidade com sistemas legados e futuros.
- A solução deve permitir a exportação de dados em múltiplos formatos, incluindo .CSV, .XML, .JSON, .XLSX e .PDF.
- A solução deve aderir ao padrão FHIR estabelecido pela RNDS, garantindo interoperabilidade e conformidade com as normas de saúde pública.

3.4. Segurança e Conformidade:

- Quando tratarmos de desenvolvimento de software, seguir rigorosamente as regras descritas nos documentos "Cláusulas de Segurança da Informação", "Manual de Desenvolvimento Seguro", "Guia de Utilização GIT", e "Padrões de Versionamento e Qualidade de Código", conforme disponíveis em seus respectivos anexos.

- A proteção dos dados pessoais deve ser priorizada, em conformidade com a Lei Geral de Proteção de Dados (LGPD), garantindo a privacidade e a segurança em todas as etapas do projeto.

3.5. Desenvolvimento e Sustentação:

- A solução deve ser desenvolvida utilizando práticas de DevOps, Test-Driven Development (TDD) e metodologias ágeis, como SCRUM, para garantir a eficiência e a qualidade do código.
- Durante o período de vigência da chamada, devem ser contempladas as seguintes atividades de responsabilidade do DR/CIS:
  - Desenvolvimento
  - Sustentação (Manutenção Corretiva e Evolutiva)
  - Suporte de primeiro e segundo nível à equipe da área de negócio do contratante, seus clientes e à equipe de infraestrutura de TI, durante todo o processo, incluindo os ambientes de desenvolvimento, homologação e produção.
  - Utilização do fluxo DevOps a ser indicado pelo SESI
  - Internalização do sistema na infraestrutura do SESI
  - Operação assistida com repasse de conhecimento para o SESI após a internalização do código e aplicação na infraestrutura do SESI

3.6. Documentação e Transferência de Conhecimento:

- A documentação técnica deve ser completa, incluindo detalhes sobre a implantação, arquitetura da solução, estudo de capacity, e plano de repasse tecnológico para o SESI.
- Deve-se garantir a elaboração de documentação técnica indicada pelo SESI, incluindo:
  - Documentação técnica de implantação da solução no ambiente SESI
  - Documentação de arquitetura da Solução
  - Estudo de capacity com indicação do dimensionamento do ambiente SESI para uso da solução a nível nacional
- Elaboração e validação junto ao SESI de Projeto de repasse tecnológico e Internalização da solução na infraestrutura do SESI a ser concluído antes do encerramento da chamada/projeto.

3.7. Requisitos Funcionais e Não Funcionais:

- A solução deve ser compatível com os principais navegadores, como Microsoft Edge, Mozilla Firefox, e Google Chrome (últimas três versões).
- Deve permitir acesso ilimitado de usuários com perfis configuráveis, disponibilizar manuais, tutoriais e assistentes virtuais, além de oferecer uma interface amigável e prática para o usuário.
- Os projetos devem ser desenvolvidos conforme os níveis hierárquicos do SESI e possibilitar a segmentação dos dados conforme a estrutura de atuação: Departamento Nacional (DN) → Departamento Regional (27 DRs) → Unidades Operacionais do SESI (UO) → Empresa cliente → Estabelecimento cliente (CNPJ ou CEI) → pessoas físicas.

3.8. Fluxo de Validação:

- Os testes da solução deverão ser realizados em empresas piloto e Departamentos Regionais do SESI, com o objetivo de validar funcionalidades e processos definidos no plano de projeto.
- O plano de projeto deve prever a validação da documentação técnica pela equipe do SESI DN antes do desenvolvimento.

4.Anexos

- Anexo 1: Cláusulas de Segurança da Informação;
- Anexo 2: Manual de Desenvolvimento Seguro;
- Anexo 3: Guia de Utilização GIT;
- Anexo 4: Padrões de Versionamento;
- Anexo 5: Qualidade de Código.

Anexo 1: Cláusulas de Segurança da Informação

1. Cláusulas de Segurança:

1.1. Visando a SEGURANÇA DA INFORMAÇÃO, caberá às PARTES:

1.1.1. Cada PARTE será única responsável pela seleção, implementação, e manutenção de procedimentos e políticas de segurança que sejam suficientes para garantir que: (i) o uso da conexão de rede por tal parte (e o uso do patrimônio de informações dos CONTRATANTES, pela CONTRATADA) seja seguro e utilizado somente para fins autorizados, e (ii) os registros e dados de negócios de tal parte estejam protegidos contra acesso ou uso indevidos, alteração, perda, ou destruição.

1.1.2. Qualquer terceira parte que conecte um dispositivo a uma rede dos CONTRATANTES deverá ser autorizada pelo Encarregado de Segurança de TI local. O Fornecedor deverá prover proteção contra o uso externo não autorizado, modificação, divulgação ou destruição, acidental ou intencional, do patrimônio dos CONTRATANTES. A terceira parte será responsável pela segurança do dispositivo, através de antivírus e patches, para proteger o patrimônio de informações dos CONTRATANTES. Qualquer dispositivo pertencente a uma terceira parte, e que armazene ativos de informações dos CONTRATANTES, deverá ser criptografado.

1.1.3. Somente usuários autorizados pelos CONTRATANTES poderão acessar as informações e dados contidos na infraestrutura dos CONTRATANTES ou na infraestrutura de terceiros contratados ou utilizados pelos CONTRATANTES. O CONTRATANTE deverá deter o poder de decisão final, a respeito de quem estará autorizado a acessar os dados e informações, incluindo o próprio pessoal dos CONTRATANTES, o pessoal da CONTRATADA, todo o pessoal subcontratado, e qualquer outra terceira parte. Todos os acessos deverão atender às Normas e políticas de tecnologia da informação dos CONTRATANTES.

1.1.4. Serão concedidas somente autorizações de acessos individuais (login e senha). Contas genéricas ou compartilhadas são absolutamente proibidas.

1.1.5. Dados ou informações sobre os CONTRATANTES, ou contidos na infraestrutura dos CONTRATANTES, quer possuídos, ou cedidos, ou hospedados nas instalações de uma terceira parte, ou na infraestrutura de um Fornecedor de Serviços de Aplicativos, não deverão ser divulgadas a quaisquer terceiras partes, sem a prévia aprovação por escrito dos CONTRATANTES.

1.2. Visando a SEGURANÇA DA INFORMAÇÃO, caberá à CONTRATADA:

1.2.1. Obedecer às normas, procedimentos e Políticas de Tecnologia da Informação dos CONTRATANTES, com todos os regulamentos e todas as atualizações correspondentes deste enquadramento, relativas aos países a partir dos quais a CONTRATADA preste os serviços ou hospede aplicativos ou dados.

1.2.2. Preservar a disponibilidade e precisão das informações dos processos suportados e proteger as informações confidenciais e proprietárias, bem como os direitos de propriedade intelectual dos CONTRATANTES.

1.2.3. Assegurar que o acordo de confidencialidade e as obrigações deste CONTRATO sejam atendidos por seus diretores, funcionários, representantes, agentes, e quaisquer outros subcontratados que irão executar tarefas descritas neste CONTRATO, antes da CONTRATADA direcioná-los à prestação dos serviços objeto deste CONTRATO.

1.2.4. As alterações relativas aos aplicativos, arquitetura, procedimentos operacionais, procedimentos de segurança e sua respectiva avaliação de riscos, deverão ser comunicadas antecipadamente.

1.2.5. Oferecer a sua máxima cooperação, para permitir que os CONTRATANTES realizem a verificação de conformidade dos produtos fornecidos e/ou serviços prestados.

1.2.6. Atender aos requisitos de governança de tecnologia da informação dos CONTRATANTES, o que implica em fornecer informações úteis sobre: (i) infraestrutura / arquitetura de tecnologia da informação e de segurança atualizadas, (ii) organização de tecnologia da informação responsável pelo serviço; e (iii) alterações relativas à arquitetura e procedimentos de segurança, e sua correspondente avaliação de riscos.

1.2.7. Responsabilizar-se exclusivamente perante os CONTRATANTES sobre todas as considerações de segurança de TI. Sob nenhuma circunstância, os CONTRATANTES assumirão ou aceitarão o compartilhamento de responsabilidade entre o Fornecedor e qualquer outra terceira parte, incluindo parceiros de negócios e subcontratados do Fornecedor.

1.2.8. Assegurar que todas as atividades do serviço mantenham adequadamente os registros de negócios apropriados, de acordo com as melhores práticas da indústria e em conformidade com as leis e regulamentações aplicáveis, e assegurar também que os documentos estejam protegidos contra acesso ou uso indevidos, perda, alteração ou destruição. A CONTRATADA deverá realizar revisões regulares para detectar potenciais problemas de segurança. A CONTRATADA poderá ser obrigada a revelar as configurações, o processo de revisão, e os resultados.

1.2.9. Responsabilizar-se única e exclusivamente por todas as medidas de segurança para garantir a proteção de seus sistemas e redes internas, isentando os CONTRATANTES de qualquer responsabilidade.

1.2.10. Juntamente com a definição de requisitos de negócios para novos sistemas ou serviços de informação, ou melhorias em sistemas ou serviços de informação existentes, a CONTRATADA deverá assegurar que a avaliação de segurança seja realizada e que controles de segurança apropriado sejam identificados e incorporados aos requisitos. A avaliação e os controles de segurança a serem incluídos deverão ser analisados e aprovados pelos CONTRATANTES.

1.2.11. Não deverá estabelecer qualquer tipo de link de comunicação de dados entre as suas redes e sistemas e os dos CONTRATANTES, incluindo, mas não se limitando a atividades de supervisão, intervenção remota, e qualquer tipo de transferência de dados, por qualquer meio, incluindo, mas não se limitando a links diretos permanentes, Internet, VPN [rede privada virtual], ou dial-up [acesso por linha discada], sem a aprovação dos CONTRATANTES.

1.2.12. Garantir que o ambiente de Tecnologia da Informação, em seu lado, assegure a integridade, disponibilidade e confidencialidade do patrimônio de Tecnologia da Informação dos CONTRATANTES, incluindo dados, informações e direitos de propriedade intelectual.

1.2.13. Somente usuários autenticados e autorizados, com uma data de expiração de usuário pré-definida, deverão ter permissão para acessar sistemas dos CONTRATANTES e o prazo de acesso deverá ser restrito.

1.2.14. Deverão ser mantidos logs de auditoria completos, nos níveis de conexão, sistema operacional, e aplicativos. Os CONTRATANTE se reserva o direito de supervisionar em tempo real, gravar e utilizar os dados armazenados das conexões.

1.2.15. Todas as conexões com a CONTRATADA deverão ser criptografadas e serão estabelecidas somente através de firewalls dos CONTRATANTES. Somente serão permitidos hosts e protocolos previamente autorizados.

1.2.16. Notificar imediatamente os CONTRATANTES, sempre que qualquer funcionário autorizado da CONTRATADA deixe a empresa, ou já não necessite de acesso, ou tenham sido modificadas as responsabilidades para acessar qualquer patrimônio de informações dos CONTRATANTES.

1.2.17. Assim que termine a prestação do serviço pela CONTRATADA, por qualquer razão, normal ou anormal, todos os acessos deverão ser imediatamente revogados. A CONTRATADA não será mais autorizada a acessar o patrimônio de informações dos CONTRATANTES.

1.2.18. Todas as informações de propriedade dos CONTRATANTES deverão ser enviadas os CONTRATANTES, em duas cópias, sendo em seguida removidas com segurança de todos os dispositivos.

1.2.19. Rever a qualquer tempo o acesso de seus empregados ao patrimônio de informações dos CONTRATANTES, e prontamente corrigir qualquer discrepância em potencial. Mediante solicitação dos CONTRATANTES, a CONTRATADA deverá revelar os resultados da última análise de acessos, bem como as respectivas ações corretivas.

1.2.20. Não utilizar qualquer criptografia desenvolvida internamente, para proteger as informações, os aplicativos ou a infraestrutura dos CONTRATANTES. Quaisquer algoritmos simétricos, assimétricos, ou de hashing, utilizado na infraestrutura dos CONTRATANTES, deverá utilizar algoritmos e ferramentas bem conhecidos, publicados, apresentados à comunidade global de criptografia, e adotados como padrão da indústria.

1.2.21. A resistência dos algoritmos de criptografia deverá ser a mais alta possível, aprovada pelo CSO [Chief Security Officer / Chefe de Segurança] local dos CONTRATANTES, e em conformidade com os requisitos de negócios dos CONTRATANTES e com a legislação aplicável.

1.2.22. Conexões que utilizem a Internet deverão ser protegidas utilizando tecnologias de criptografia, adotados como padrão de mercado.

1.2.23. Deverão ser criptografados todos os dispositivos móveis e portáteis utilizados para prover o serviço os CONTRATANTES e que contenham dados confidenciais.

1.2.24. Prover os CONTRATANTES uma lista com os nomes e funções dos funcionários que terão acesso aos Locais dos CONTRATANTES. Pessoas incluídas na lista acima referida deverão apresentar-se na recepção do Local dos CONTRATANTES, onde receberão um cartão de visitante ou crachá, que deverão portar de forma visível, durante todo o tempo de permanência nas instalações dos CONTRATANTES.

1.2.25. Se, por qualquer motivo, uma pessoa não incluída na lista precise acessar o site, ele ou ela deverá ser registrado na recepção, mediante apresentação de sua identificação com foto, ou passaporte. A referida pessoa deverá ser acompanhada por um colaborador dos CONTRATANTES, em todos os momentos.

1.2.26. Caso algum membro do pessoal da CONTRATADA, quer previamente autorizado através da lista ou não, necessite acessar áreas sensíveis (tais como salas de servidores, datacenters, armários de rede, etc.), este deverá ser acompanhado por um colaborador dos CONTRATANTES, em todos os momentos.

1.2.27. Se a CONTRATADA necessitar acessar os Locais, fora dos horários de trabalho, os CONTRATANTES deverão informar à CONTRATADA as regras específicas a serem respeitadas.

1.2.28. Designar uma pessoa como responsável pela proteção e descarte seguro dos meios de armazenamento. Caso qualquer mídia de armazenamento precise ser descartada, a CONTRATADA deverá desativá-la de maneira segura, de tal forma que os dados não possam ser recuperados a partir da mesma. A mídia deverá ser descartada utilizando uma terceira parte certificada para realizar descarte seguro, ou destruída fisicamente (após os dados haverem sido apagados) tornando-a completamente inoperante. A desativação / destruição de mídia requer a aprovação prévia dos CONTRATANTES e a posterior notificação por esta à CONTRATADA.

1.2.29. Quando houver a ocorrência de qualquer incidente envolvendo os serviços contratados, execução do CONTRATO e os dados e/ou informações de clientes do CONTRATANTE (e/ou

suas próprias informações), a CONTRATADA obriga-se a comunicar imediatamente o CONTRATANTE tomando de imediato todas as medidas que possam minimizar eventuais perdas e danos causados em razão do incidente, além de aplicar os melhores esforços para cessar o incidente com a maior brevidade possível.

1.3. Visando a SEGURANÇA DA INFORMAÇÃO, caberá os CONTRATANTES:

1.3.1. Verificar e controlar a qualquer tempo que os produtos fornecidos e/ou serviços prestados pela CONTRATADA estejam em conformidade com o CONTRATO.

1.3.2. Comunicar à CONTRATADA as regras de segurança estabelecidas em conformidade com a legislação aplicável, bem como as regras de acesso.

1.3.3. Autorizar antecipadamente, conforme sua conveniência, o uso de qualquer tipo de mídia de armazenamento, com exceção dos discos rígidos internos e fitas de backup. O uso de mídias de armazenamento removíveis e portáteis, tais como CDs, DVDs, drives USB, Flashdrives, etc., é estritamente proibido.

1.3.4. Autorizar antecipadamente, conforme sua conveniência, todos os ativos de tecnologia da informação dos CONTRATANTES que devam deixar as instalações da CONTRATADA (por exemplo, para reparação). Todas as informações armazenadas no dispositivo deverão ser previamente removidas de forma segura. Se a remoção não for prática ou possível, a CONTRATADA será o responsável principal por proteger os ativos contra uso não autorizado, modificação, divulgação, ou destruição, acidental ou intencional.

1.3.5. Previamente à remoção dos ativos das instalações dos CONTRATANTES, os controles de segurança a estabelecer deverão ser apresentados pelo CONTRATADA e aprovados pelos CONTRATANTES.

1.3.6. Se os ativos tiverem de ser desativados ou descartados, as informações deverão ser destruídas de forma segura, e o processo de descarte deverá estar em conformidade com as leis e regulamentos locais.

1.4. Visando a SEGURANÇA DA INFORMAÇÃO, disposições gerais:

1.4.1. As auditorias poderão verificar, especialmente mas não limitando, a: (i) desempenho e qualidade dos serviços prestados; (ii) o cumprimento das obrigações deste CONTRATO, inclusive os níveis de serviços dos mesmos; (iii) teste internos e externos para assegurar o cumprimento das obrigações de acordo com a legislação e o CONTRATO firmado entre as PARTES; e (iv) aplicação de testes internos de penetração na infraestrutura ou aplicativos hospedados.

1.4.2. Independentemente das tarefas contidas diretamente na atividade de serviço de segurança, ou se o serviço incluir atividades de segurança, a CONTRATADA deverá assegurar que todas as atividades do Serviço atendam às Políticas de Segurança da Informação dos CONTRATANTES e às normas e diretrizes de segurança e arquitetura, tanto aquelas existentes no momento da assinatura, quanto àquelas que venham a ser lançadas no futuro.

1.4.3. As atividades de segurança deverão proporcionar proteção contra o uso não autorizado, modificação, divulgação ou destruição, acidental ou intencional, para o patrimônio de dados corporativos, oferecendo trilhas de auditoria adequada, e em conformidade com as leis aplicáveis.

1.4.4. O escopo das atividades de segurança deverá incluir (não se limitando a) os seguintes processos: antivírus, patches [correções] e atualizações, criptografia, configuração de novos usuários, redefinições de senhas, e exclusão ou modificação de contas de usuários.

1.4.5. Qualquer mudança na configuração do ambiente de produção deverá obedecer ao procedimento de autorização. Quando a CONTRATADA for responsável pela configuração e/ou

gestão de equipamentos, deverá informar os CONTRATANTES a configuração atual, as alterações propostas, o protocolo de testes, e os resultados dos testes. As alterações de configuração só poderão ser aplicadas após a notificação os CONTRATANTES e sua aprovação.

1.4.6. Todo o desenvolvimento de software deverá incluir um teste de segurança, com a finalidade de evitar a injeção de código malicioso. O processo e os resultados dos testes de segurança da CONTRATADA deverão ser revelados os CONTRATANTES (por exemplo, testes das funções de autenticação, autorização e contabilização, revisões de código-fonte, e quaisquer outras atividades destinadas a validar a arquitetura de segurança)

1.4.7. Desativar imediatamente a totalidade ou parte da funcionalidade do serviço, caso seja identificado um problema de segurança.

1.4.8. A CONTRATADA deverá apresentar um documento da arquitetura proposta, que inclua um completo diagrama da rede do Ambiente dos CONTRATANTES, ilustrando a relação entre o Ambiente dos CONTRATANTES e quaisquer outras redes relevantes, com um fluxograma de dados completo, detalhando onde residem os dados dos CONTRATANTES, os aplicativos que os manipulam, e a segurança dos mesmos. Quaisquer alterações ou atualizações no modelo de comunicação de dados entre CONTRATADA e CONTRATANTES, mudanças significativas e atualizações nas configurações de segurança, deverão ser previamente comunicadas e aprovadas pelos CONTRATANTES.

1.4.9. A rede que hospeda o aplicativo deverá ser fisicamente isolada [air-gapped] de qualquer outra rede ou cliente que a CONTRATADA possa ter.

1.4.10. As instalações de desenvolvimento, testes e operacionais deverão ser isolados uma das outras. A CONTRATADA deverá cumprir as políticas dos CONTRATANTES para a transferência de software do ambiente de desenvolvimento para o ambiente de produção. Software em desenvolvimento e software operacional deverão ser executados em sistemas ou processadores de computador diferentes, e em domínios, diretórios e segmentos de rede diferentes. Compiladores, editores e outras ferramentas de desenvolvimento ou utilitários de sistema não deverão ser acessíveis a partir de ambientes produção.

1.4.11. Qualquer administrador de sistemas com acesso ao ambiente de desenvolvimento não deverá ter acesso ao ambiente de produção, sem autorização dos CONTRATANTES

1.4.12. A nenhum desenvolvedor deverá ser concedido acesso ao ambiente de produção. Qualquer exceção ocasionada por uma alteração de emergência deverá ser aprovada pelos CONTRATANTES, notificada e devidamente documentada.

1.4.13. O ambiente de testes deverá emular o ambiente operacional do sistema, tanto quanto possível.

1.4.14. Os dados de testes deverão ser protegidos e controlados em conformidade com as políticas dos CONTRATANTES. O Fornecedor não está autorizado a utilizar dados reais ou dados de produção, para testes. Caso se tornem necessárias quaisquer cópias dos dados de produção, cada cópia deverá ser previamente autorizada pelos CONTRATANTES, registrada, e devidamente sanitizada [depurada] / codificada. O processo a ser usado para sanitização e/ou codificação dos dados deverá ser previamente aprovado pelos CONTRATANTES. Logo que os testes sejam concluídos, os dados deverão ser removidos com segurança, do ambiente de testes.

1.4.15. Para todos os ambientes, a CONTRATADA deverá revelar como, e em que medida, os hosts e servidores que compõem a infraestrutura dos CONTRATANTES foram reforçados contra-ataques. A CONTRATADA deverá fornecer uma lista de patches atuais nos hosts, incluindo patches do Sistema Operacional host, de servidores web, de bancos de dados, e de qualquer outro aplicativo material, bem como informações sobre como e quando os patches de segurança serão aplicados. Em caso de dispositivos de rede, laptops e desktops, os CONTRATANTES poderão exigir estas ou quaisquer outras informações. A CONTRATADA

deverá revelar os seus processos para monitoramento da integridade e disponibilidade desses hosts.

1.4.16. A CONTRATADA deverá atender à política de senhas dos CONTRATANTES, para a infraestrutura dos CONTRATANTES (incluindo Sistema Operacional, Bancos de dados, e dispositivos de rede que suportem o serviço), incluindo comprimento mínimo de senhas, diretrizes de geração de senhas, e idade de senhas. Contas de usuário e/ou senhas compartilhadas são proibidas. A CONTRATADA deverá revelar o método utilizado para autenticar usuários.

1.4.17. Deverá ser definido um procedimento de acesso de emergência, para o acesso à infraestrutura dos CONTRATANTES. Este processo precisa ser aprovado pelos CONTRATANTES.

1.4.18. A CONTRATADA deverá fornecer informações sobre a geração, manutenção e processo de encerramento de contas, para "contas especiais" (de administração, de serviço, de monitoramento e de manutenção), bem como para contas de usuários. Esta informação deverá abranger a criação de contas, informação de contas e senhas para usuários finais, e revogação de contas.

1.4.19. A CONTRATADA deverá fornecer informações sobre a sua política de logs, incluindo as configurações de logs implementadas, e procedimento de revisão.

1.4.20. Assim que o serviço prestado termine (vencimento do contrato), a CONTRATADA deverá entregar duas cópias das informações dos CONTRATANTES e realizar de forma segura a destruição das informações armazenadas. O acesso deverá ser imediatamente revogado, e a CONTRATADA não será mais autorizado a acessar o patrimônio de informações dos CONTRATANTES.

1.4.21. A CONTRATADA deverá revelar a forma pela qual irá autenticar os usuários (por exemplo, LDAP, Netegrity, Certificados de Cliente).

1.4.22. A CONTRATADA deverá fornecer informações sobre a sua política de senhas para a infraestrutura de aplicativos dos CONTRATANTES, incluindo comprimento mínimo de senhas, diretrizes de geração de senhas, e a frequência com que as senhas serão alteradas.

1.4.23. A CONTRATADA deverá revelar a forma pela qual protege as senhas, quando armazenadas e transmitidas dentro da infraestrutura de aplicativos dos CONTRATANTES, e a forma pela qual destrói as informações, quando não tenham mais utilidade.

1.4.24. A CONTRATADA poderá ser obrigada a revelar os arquivos de configuração específicos, para qualquer sistema associado às funções de suporte (tais como mecanismos de busca ou bases de dados) e detalhes técnicos (linguagens de programação, etc.) sobre o(s) aplicativo(s).

1.4.25. Os equipamentos que hospedem patrimônio de Tecnologia da Informação dos CONTRATANTES deverão estar localizados em uma instalação fisicamente segura. Será requerido, no mínimo, o controle por crachás, para acesso à instalação. O patrimônio de Tecnologia da Informação dos CONTRATANTES deverá estar localizado em um ambiente do tipo sala cofre.

1.4.26. O CONTRATANTE deverá deter o poder de decisão final, a respeito de quem estará autorizado a entrar em qualquer ambiente físico fechado e acessar o patrimônio de Tecnologia da Informação dos CONTRATANTES. A CONTRATADA deverá revelar a lista do pessoal autorizado a acessar o ambiente de hospedagem do patrimônio de Tecnologia da Informação dos CONTRATANTES. Todos os controles de segurança física deverão ser verificados regularmente.

1.4.27. O procedimento de autorização para o processo de colocação em produção deverá respeitar as políticas e procedimentos de tecnologia da informação dos CONTRATANTES. Os

Testes de Aceitação do Usuário (que serão responsabilidade e prerrogativa dos CONTRATANTES, independentemente do suporte necessário para realizar estas tarefas) deverão ser realizados antes de qualquer mudança na produção.

1.4.28. O acesso ao código-fonte de programas e itens associados (tais como projetos, especificações, planos de verificação e planos de validação) deverá ser estritamente controlado. Bibliotecas de programas-fonte não deverão ser mantidas em sistemas em operação, e qualquer exceção a esta regra deverá ser aprovada pelos CONTRATANTES. Todas as atualizações de bibliotecas de programas-fonte e itens associados, bem como a emissão de fontes de programas para os programadores, deverão ser executadas somente após a devida autorização haver sido recebida.

1.4.29. Um Acordo de Custódia deverá ser firmado entre CONTRATANTES e Fornecedor com a finalidade de proteger os investimentos de tecnologia dos CONTRATANTES, através da garantia da manutenção do software, ou do acesso ao código-fonte. O Fornecedor deverá propor, discutir e celebrar um Acordo apropriado com os CONTRATANTES definindo explicitamente como: o depósito do código-fonte do software em uma conta mantida por uma terceira parte, como agente depositário; as condições detalhadas, sob as quais o código-fonte será liberado, permitindo os CONTRATANTES acessá-lo caso o licenciador se declare insolvente, ou deixe de manter e atualizar o software, conforme prometido no contrato de licença de software.

1.4.30. A CONTRATADA deverá gerenciar todas as medidas necessárias para assegurar a continuidade do serviço, em conformidade com o SLA [Service Level Agreement / Acordo de Níveis de Serviço] definido, incluindo a aplicação das atualizações e "patches" [correções] necessárias para hardware e software, conforme recomendação de cada Fornecedor de Tecnologia.

1.4.31. A CONTRATADA deverá oferecer regularmente as recomendações necessárias para garantir a compatibilidade com as últimas versões de software, hardware, e software distribuído comercialmente, à medida que tais versões sejam autorizadas no âmbito dos CONTRATANTES e executar as atividades recomendadas, tão logo seja possível, após obtida a aprovação dos CONTRATANTES.

1.4.32. O serviço incluirá gestão de backup e restauração: designadamente, um serviço de planejamento, implantação e testes dos procedimentos de recuperação (incluindo login pelos usuários finais) necessários para restabelecer a funcionalidade do Serviço, Sistema Operacional do sistema, bancos de dados, aplicativos, e dados dos usuários, em caso de uma falha.

1.4.33. Gerenciar todas as medidas necessárias para assegurar a continuidade do serviço objeto deste CONTRATO.

1.4.34. A CONTRATADA deverá informar regularmente as recomendações necessárias para garantir a compatibilidade com as últimas versões de software, hardware, e software distribuído comercialmente, à medida que tais atualizações das versões sejam autorizadas por escrito pelos CONTRATANTES, a CONTRATADA deverá executar as atividades recomendadas, passando os documentos trocados pela PARTES, em razão do disposto neste item serão parte integrante do CONTRATO.

1.4.35. Ao firmar o presente instrumento, a CONTRATADA compromete-se a observar, durante a execução dos serviços de desenvolvimento/manutenção nos Sistemas Aplicativos dos CONTRATANTES, o Anexo IA - Ciclo de Vida de Desenvolvimento de Software - SDLC .

1.4.36. O Guia de Desenvolvimento Seguro poderá ser, a critério dos CONTRATANTES, atualizado de tempos em tempos. Havendo atualizações, no momento da solicitação de proposta comercial, os CONTRATANTES disponibilizarão à CONTRATADA a última versão do mencionado guia.

1.4.37. Eventuais manutenções, aperfeiçoamentos, atualizações e/ou modificações cuja execução seja necessária para garantir que os serviços de desenvolvimento/manutenção

estejam em conformidade com o Guia de Desenvolvimento Seguro e, ainda, com a legislação aplicável e/ou às demais normas expedidas por qualquer autoridade governamental competente, serão realizados pela CONTRATADA, sem qualquer ônus os CONTRATANTES. A responsabilidade da CONTRATADA pelo cumprimento das normas bem como da legislação aplicável vigorará por tempo indeterminado.

1.4.38. A CONTRATADA concorda que: (a) deverá entregar os CONTRATANTES, além do Código-Fonte, a Documentação referente a cada desenvolvimento.

1.4.39. A CONTRATADA concorda que: (b) Os Sistemas Aplicativos, incluindo seus programas, aperfeiçoamentos, adaptações, correções e demais funcionalidades desenvolvidas pela CONTRATADA, sob encomenda dos CONTRATANTES, em decorrência do Contrato, assim como a respectiva Documentação são de propriedade intelectual total e definitiva dos CONTRATANTES.

1.4.40. A CONTRATADA concorda que: (c) Os CONTRATANTES é a única legitimada a promover qualquer registro relacionado aos dados, informações, relatórios ou quaisquer outros documentos criados pela CONTRATADA no cumprimento do objeto do Contrato, podendo realizar tais registros da maneira que julgar conveniente, para resguardar ou salvaguardar seus direitos de exclusiva proprietária;

1.4.41. A CONTRATADA concorda que: (d) A CONTRATADA, desde já, compromete-se a fornecer para os CONTRATANTES toda e qualquer informação ou documento necessário para a realização dos registros mencionados no item (b) acima;

1.4.42. A CONTRATADA concorda que: (e) A CONTRATADA somente poderá fazer uso de qualquer Material Pré-Existente para a prestação dos serviços, se obtiver prévia autorização por escrito dos CONTRATANTES. Caso os CONTRATANTES expressamente autorize a utilização de Material Pré-Existente para o desenvolvimento de um determinado Sistema Aplicativo, a CONTRATADA deverá ceder os CONTRATANTES, o direito perpétuo de utilização de referido Material Pré-Existente que não possa ser obtido/licenciado diretamente pelos CONTRATANTES e que seja necessário à absorção da tecnologia e à compilação dos Sistemas Aplicativos desenvolvidos sob o Contrato;

Anexo 2: Manual de Desenvolvimento Seguro_2022

Procedimento para Desenvolvimento Seguro de Aplicações Web

Código: **PR_27**
Versão: **1.0**

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

Controle de Versão

Histórico (para uso da Gerência de Governança):

Versão	Data Publicação	Elaborado	Motivo da Versão
1.0	01/02/2017	Gerência de Infraestrutura	Tornar as aplicações web mais seguras
2.0	22/07/2022	Rogério Ferreira	Atualização e recomendações de práticas de segurança da informação com vistas a mitigar riscos de vulnerabilidades em aplicações web.

Aprovação da Versão Atual:

Nome	Cargo
Fábio Leandro Bernardes Duarte	Gerente de Tecnologia do Negócio

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

Sumário

1. Justificativa 4
2. Objetivo 4
3. Escopo 4
4. Referências 4
5. Considerações Gerais 4
6. Descrição do Procedimento 5
   - 6.1 Requisito 1: Validação dos dados de entrada 5
   - 6.2 Requisito 2: Codificação de dados de saída 6
   - 6.3 Requisito 3: Autenticação e gerenciamento de credenciais 7
   - 6.4 Requisito 4: Gerenciamento de sessões 9
   - 6.5 Requisito 5: Controle de acesso 10
   - 6.6 Requisito 6: Criptografia 12
   - 6.7 Requisito 7: Tratamento de erros e logs 13
   - 6.8 Requisito 8: Proteção de dados 14
   - 6.9 Requisito 9: Segurança nas comunicações 15
   - 6.10 Requisito 10: Configuração do sistema 15
   - 6.11 Requisito 11: Segurança em Banco de Dados 16
   - 6.12 Requisito 12: Gerenciamento de Arquivos 17
   - 6.13 Requisito 13: Gerenciamento de memória 18
   - 6.14 Requisito 14: Práticas Gerais de Codificação 19
7. Exceções 20
8. Utilização de Dados Pessoais (LGPD) 20
9. Anexos 20

Título:	
Procedimento para Desenvolvimento Seguro de Aplicações Web	
Diretoria:	Gerência Responsável:
DSC/STI	**Gerência de Tecnologia do Negócio**
Dono do Processo:	Elaboradores:
Fábio Leandro Bernardes Duarte	**Rogério Oliveira Ferreira** **Darley Fernandes da Silva**

1. Justificativa

O desenvolvimento seguro de aplicações web é primordial para que as organizações não venham a sofrer impactos importantes provenientes de ataques cibernéticos pela exploração de vulnerabilidade em aplicação web. Segurança cibernética tem sido uma preocupação constante nas organizações devido ao alto índice de ameaças que exploram vulnerabilidades em seus ambientes tecnológicos. A busca por melhores práticas em matéria de segurança em aplicações web tem levado a implementação de algumas práticas baseadas em frameworks como Open Web Application Security Project (OWASP) e do Center for Internet Security (CIS).

2. Objetivo

O objetivo deste procedimento é auxiliar os profissionais de desenvolvimento e manutenção de sistemas da Superintendência de Tecnologia da Informação - STI a adotarem práticas recomendadas de segurança da informação com vistas a mitigar riscos de vulnerabilidades em aplicações web.

3. Escopo

O escopo deste documento abrange todas as aplicações web a serem desenvolvidas, seja por profissionais da STI ou fornecedores contratados.

4. Referências

OWASP - Open Web Application Security Project
CIS - Center for Internet Security

5. Considerações Gerais

A equipe da STI de infraestrutura e desenvolvimento é responsável por validar todos os requisitos constantes neste documento para que aplicações web não entrem em ambiente de produção com vulnerabilidades que de alguma forma possam incidir nos casos de ataques apresentados abaixo, mas não se limitando a estes:

Casos de abuso	Riscos
Injeção de código	Ocorrem quando dados não confiáveis são enviados para um interpretador como parte de um comando ou consulta. Os dados manipulados pelo atacante podem iludir o interpretador para que este execute comandos indesejados ou permita o acesso a dados não autorizados.

Casos de abuso	Riscos
Quebra de Autenticação e Gerenciamento de Sessão	Ocorre quando as funções da aplicação relacionadas à autenticação e gerenciamento de sessão são implementadas de forma incorreta, permitindo que os atacantes comprometam senhas, chaves e tokens de sessão ou explorem outra falha da implantação para assumir a identidade de outros usuários.
Cross-Site Scripting (XSS)	Falhas XSS permitem aos atacantes executarem *scripts* no navegador da vítima.
Referência Insegura e Direta a Objetos	Ocorre quando um programador expõe uma referência à implantação interna de um objeto, como um arquivo, diretório, ou registro da base de dados.
Exposição de Dados Sensíveis	Ocorre quando as aplicações não protegem devidamente os dados sensíveis, tais como cartões de crédito, IDs fiscais e credenciais de autenticação.
Falta de Função para Controle do Nível de Acesso	Ocorre quando as aplicações não verificam os direitos de acesso, em nível de função, antes de tornar essa funcionalidade visível na interface do usuário.
Cross-Site Request Forgery (CSRF)	Ocorre quando o navegador da vítima é forçado a executar uma ação maliciosa em favor do atacante.
Utilização de Componentes Vulneráveis Conhecidos	Componentes, tais como bibliotecas, *frameworks*, e outros módulos de *software* quase sempre são executados com privilégios elevados.
Redirecionamentos e Encaminhamentos Inválidos	Aplicações frequentemente redirecionam e encaminham usuários para outras páginas ou sites, e usam dados não confiáveis para determinar as páginas de destino.

6. Descrição do Procedimento

Nesta sessão do documento estão descritos os requisitos para desenvolvimento seguro bem como o detalhamento dos controles de segurança críticos que devem ser observados para o desenvolvimento seguro:

6.1 Requisito 1: Validação dos dados de entrada

As vulnerabilidades baseadas nos dados de entrada podem surgir em qualquer funcionalidade de uma aplicação, e podem estar presentes em praticamente toda tecnologia de uso comum existente. Uma grande variedade de ataques a aplicações web se origina ou envolve o envio de dados de entrada

inesperados ou especialmente construídos para causar comportamentos inesperados na aplicação. Sendo assim, toda entrada de dados em um sistema deve ser considerada, a princípio, não confiável.

Subcontroles de Segurança Críticos - Validação dos dados de entrada
1.1 Efetuar toda a validação dos dados em um sistema confiável – por exemplo, centralizar todo o processo no servidor.
1.2 Identificar todas as fontes de dados e classificá-las como sendo confiáveis ou não. Em seguida, validar os dados provenientes de fontes nas quais não se possa confiar (ex: base de dados, stream de arquivos etc.).
1.3 A rotina de validação de dados de entrada deve ser centralizada na aplicação.
1.4 Especificar conjunto de caracteres apropriado, como UTF-8, para todas as fontes de entrada de dados.
1.5 Codificar os dados para um conjunto de caracteres comuns antes da validação (Canonicalize).
1.6 Quando há falha de validação, a aplicação deve rejeitar os dados fornecidos.
1.7 Determinar se o sistema suporta conjuntos de caracteres estendidos UTF-8 e, em caso afirmativo, validar após efetuar a descodificação UTF-8.
1.8 Validar todos os dados provenientes dos clientes antes do processamento, incluindo todos os parâmetros, campos de formulário, conteúdo das URLs e cabeçalhos HTTP, como, por exemplo, os nomes e os valores dos Cookies. Certificar-se, também, de incluir mecanismos automáticos de postback nos blocos de código JavaScript, Flash ou qualquer outro código embutido.
1.9 Verificar se os valores de cabeçalho, tanto das requisições, como das respostas, contêm apenas caracteres ASCII.
1.10 Validar dados provenientes de redirecionamentos. Os atacantes podem incluir conteúdo malicioso diretamente para o alvo do mecanismo de redirecionamento, podendo assim contornar a lógica da aplicação e qualquer validação executada antes do redirecionamento.
1.11 Validar tipos de dados esperados.
1.12 Validar intervalo de dados.
1.13 Validar o tamanho dos dados.
1.14 Validar, sempre que possível, todos os dados de entrada através de um método baseado em “listas de permissões" (whitelists) que utilizem uma lista de caracteres ou expressões regulares para definirem os caracteres permitidos.
1.15 Se qualquer caractere potencialmente perigoso precisa ser permitido na entrada de dados da aplicação, certificar-se de que foram implementados controles adicionais como codificação dos dados de saída, APIs específicas que fornecem tarefas seguras e trilhas de auditoria no uso dos dados pela aplicação. A seguir, como exemplo de caracteres potencialmente perigosos”, temos: <, >, ", ', %, (,), &, +, \, \', \".
1.16 Se a rotina de validação padrão não abordar as seguintes entradas, então elas devem ser verificadas:
a) Verificar bytes nulos (%00)
b) Verificar se há caracteres de nova linha (%0d, %0a, \r, \n)
c) Verificar se há caracteres “ponto-ponto barra” (../ ou ..\) que alteram caminhos. Nos casos de conjunto de caracteres que usem a extensão UTF-8, o sistema deve utilizar casos de conjunto de caracteres que usem a extensão UTF-8, o sistema deve utilizar representações alternativas como: %c0%ae%c0%ae/. A canonicalização deve ser utilizada para resolver problemas de codificação dupla (double encoding) ou outras formas de ataques por ofuscação

6.2 Requisito 2: Codificação de dados de saída

Com a diversidade de arquiteturas modernas de aplicações web, a realização da codificação de saída é muito importante. Pode ser difícil fornecer validação de entrada robusta em certos cenários. Portanto, o uso de APIs mais seguras com consultas parametrizadas, estruturas de modelagem com escape

automático ou codificação de saída cuidadosamente escolhida é fundamental para a segurança da aplicação.

O propósito da codificação de saída é converter a entrada não confiável em uma forma segura, onde a entrada é exibida como dados para o usuário, sem executar uma codificação no navegador. A tabela a seguir detalha uma lista de subcontroles de codificação de saída.

Subcontroles de Segurança Críticos - Codificação de dados de saída
2.1 Efetuar toda a codificação dos dados em um sistema confiável - por exemplo, centralizar todo o processo no servidor.
2.2 Utilizar uma rotina padrão e testada para cada tipo de codificação de saída.
2.3 Realizar a codificação, baseada em contexto, de todos os dados enviados para o cliente que têm origem em um ambiente fora dos limites de confiança da aplicação. A codificação das entidades HTML é um exemplo, mas nem sempre funciona para todos os casos.
2.4 Codificar todos os caracteres, a menos que sejam conhecidos por serem seguros para o interpretador de destino.
2.5 Realizar o tratamento (sanitização), baseado em contexto, de todos os dados provenientes de fontes não confiáveis usados para construir consultas SQL, XML e LDAP.
2.6 Tratar todos os dados provenientes de fontes que não sejam confiáveis que gerem comandos para o sistema operacional.

6.3 Requisito 3: Autenticação e gerenciamento de credenciais

Um requisito necessário em praticamente toda aplicação é o acesso do usuário aos seus dados e às funcionalidades. Esse acesso é gerenciado normalmente por três mecanismos interrelacionados:

- Autenticação
- Gerenciamento de sessões (requisito 4)
- Controle de Acesso (requisito 5)

A autenticação é o processo que busca verificar a identidade digital de uma entidade de um sistema quando tal entidade requisita acesso a esse sistema. O processo é realizado por meio de regras preestabelecidas, geralmente pela comparação das credenciais apresentadas pela entidade com outras já pré-definidas no sistema, reconhecendo como verdadeiras ou legítimas as partes envolvidas em um processo. Vide abaixo uma lista de subcontroles a serem considerados nessa hipótese:

Subcontroles de Segurança Críticos - Autenticação e gerenciamento de credenciais
3.1 Requerer autenticação para todas as páginas e recursos, exceto para aqueles que são intencionalmente públicos.
3.2 Os controles de autenticação devem ser executados em um sistema confiável - por exemplo, centralizar todo o processo no servidor.
3.3 Sempre que possível, estabelecer e utilizar serviços de autenticação padronizados e testados.
3.4Utilizar uma implementação centralizada para realizar os procedimentos de autenticação, disponibilizando bibliotecas que invoquem os serviços externos de autenticação.
3.5 Separar a lógica de autenticação do recurso que está a ser requisitado e usar redirecionadores dos controladores de autenticação centralizados.

Subcontroles de Segurança Críticos - Autenticação e gerenciamento de credenciais
3.6 Quando ocorrerem situações excepcionais nos controles de autenticação, executar procedimentos em caso de falha com o propósito de manter o sistema seguro.
3.7 Todas as funções administrativas e de gerenciamento de contas devem ser tão seguras quanto o mecanismo de autenticação principal.
3.8 Se a aplicação gerenciar um repositório de credenciais, esta deverá garantir que as senhas são armazenadas na base de dados somente sob a forma de resumo/hash da senha na forma de one-way salted hashes e que a tabela/arquivo que armazena as senhas e as próprias chaves são manipuladas apenas pela aplicação. Não utilizar algoritmos de hash reconhecidamente inseguros.
3.9 A geração dos resumos (hash) das senhas deve ser executada em um sistema confiável - por exemplo, centralizar o controle no servidor.
3.10 Validar os dados de autenticação somente no final de todas as entradas de dados, especialmente para as implementações de autenticação sequencial.
3.11 As mensagens de falha na autenticação não devem indicar qual parte dos dados de autenticação está incorreta. Por exemplo, em vez de exibir mensagens como “Nome de usuário incorreto” ou “Senha incorreta”, utilize apenas mensagens como: “Usuário e/ou senha inválidos”, para ambos os casos de erro. As respostas de erro devem ser idênticas nos dois casos.
3.12 Utilizar autenticação para conexão a sistemas externos que envolvam tráfego de informação sensível ou acesso às funções.
3.13 As credenciais de autenticação para acessar serviços externos à aplicação devem ser cifradas e armazenadas em um local protegido de um sistema confiável - por exemplo, no servidor da aplicação. Obs.: o código-fonte não é considerado um local seguro.
3.14 Utilizar apenas requisições POST para transmitir credenciais de autenticação.
3.15 Somente trafegar senhas (não temporárias) através de uma conexão protegida (SSL/TLS) ou no formato de dado cifrado, como no caso de envio de e-mail cifrado. Senhas temporárias enviadas por e-mail podem ser um caso de exceção aceitável.
3.16 Exigir que os requisitos de complexidade de senha estabelecidos pela política ou regulamento sejam cumpridos. As credenciais de autenticação devem resistir a ataques que, tipicamente, ameaçam o ambiente de produção. Um exemplo pode ser a exigência do uso simultâneo de caracteres alfabéticos, numéricos e/ou caracteres especiais.
3.17 Exigir que os requisitos de comprimento de senha estabelecidos pela política ou pelo regulamento sejam cumpridos. O uso de oito caracteres é o mais comum, porém usar 8 é mais recomendado. Considere, ainda, o uso de senhas que contenham várias palavras (uma frase).
3.18 A entrada da senha deve ser ocultada na tela do usuário. Em HTML, utilizar o campo do tipo "password".
3.19 Desativar a conta após um número pré-definido de tentativas inválidas de autenticação (e.g. cinco tentativas é o mais comum). A conta deve ser desativada por um período suficientemente longo para desencorajar a dedução das credenciais pelo método de força bruta, mas não tão longo ao ponto de permitir um ataque de negação de serviço.
3.20 Os processos de redefinição de senhas e operações de mudanças devem exigir os mesmos níveis de controle previstos para a criação de contas e autenticação
3.21 Esquemas de pergunta/resposta (pré-definidos) usados para a redefinição da senha devem evitar ataques que lancem respostas aleatórias. Por exemplo, “livro favorito” é uma questão fraca, pois “A Bíblia” é uma resposta muito comum.
3.22 Se optar por usar redefinição de senha baseada em e-mail, envie um e-mail somente para o endereço pré-definido contendo um link ou uma senha de acesso temporário que permitam ao usuário redefinir a senha.

Subcontroles de Segurança Críticos - Autenticação e gerenciamento de credenciais
3.23 O tempo de validade das senhas e dos links temporários deve ser curto.
3.24 Exigir a mudança de senhas temporárias na próxima vez que o usuário realizar a autenticação no sistema
3.25 Notificar o usuário quando a senha for reiniciada (reset).
3.26 Prevenir a reutilização de senhas.
3.27 As senhas devem ter, pelo menos, um dia de duração antes de poderem ser alteradas, a fim de evitar ataques de reutilização de senhas.
3.28 Garantir que a troca de senhas está em conformidade com os requisitos estabelecidos na política ou regulamento. Sistemas críticos podem exigir alterações mais frequentes nas credenciais de segurança. O tempo entre as trocas de senhas deve ser controlado administrativamente.
3.29 Desativar a funcionalidade de lembrar a senha nos campos de senha do navegador.
3.30 A data e a hora da última utilização (bem ou malsucedida) de uma conta de usuário devem ser comunicadas no próximo acesso ao sistema.
3.31 Realizar monitoramento para identificar ataques contra várias contas de usuários, utilizando a mesma senha. Esse padrão de ataque é utilizado para explorar o uso de senhas padrão.
3.32 Modificar todas as senhas que, por padrão, são definidas pelos fornecedores, bem como os identificadores de usuários (IDs), ou desativar as contas associadas.
3.33 Exigir nova autenticação dos usuários antes da realização de operações críticas.
3.34 Utilizar autenticação de múltiplos fatores (utilizando simultaneamente token, senha, biometria, etc.) para contas altamente sensíveis ou de alto valor transacional
3.35 Caso utilize código de terceiros para realizar a autenticação, inspecione-o cuidadosamente para garantir que ele não é afetado por qualquer código malicioso.

6.4 Requisito 4: Gerenciamento de sessões

Praticamente toda sessão de usuário é implementada por meio de um token que identifica a sessão e que é concedido após o usuário se autenticar. A sessão, em si, é um conjunto de estruturas armazenadas no servidor, que mantém controle das interações do usuário com a aplicação.

A grande maioria dos ataques contra o gerenciamento de sessões de uma aplicação busca comprometer o token concedido a outros usuários. Se bem-sucedido, o autor do ataque pode se passar pelo usuário (vítima) como se fosse o usuário autenticado.

Os principais problemas surgem na forma como o token é criado – o que pode permitir a um atacante adivinhar o token de outros usuários - e na forma como os tokens são gerenciados - permitindo que um autor de ataque obtenha o token de outro usuário e, com base no envio de tal token, realize um ataque de personificação. Vide abaixo uma lista de subcontroles a serem considerados nessa hipótese:

Subcontroles de Segurança Críticos - Gerenciamento de sessões
4.1 Utilizar controles de gerenciamento de sessão baseados no servidor ou em framework. A aplicação deve reconhecer apenas esses identificadores de sessão como válidos.
4.2 A criação dos identificadores de sessão deve ser sempre realizada em um sistema confiável - por exemplo, centralizado no servidor.
4.3 O controle de gestão de sessão deve usar algoritmos conhecidos, padronizados e bem testados que garantam a aleatoriedade dos identificadores de sessão.

Subcontroles de Segurança Críticos - Gerenciamento de sessões
4.4 Definir o domínio e o caminho para os cookies que contenham identificadores de sessão autenticados, para um valor devidamente restrito ao site.
4.5 A funcionalidade de saída (logout) deve encerrar completamente a sessão ou conexão associada.
4.6 A funcionalidade de saída (logout) deve estar disponível em todas as páginas que requerem autenticação.
4.7 Estabelecer um tempo de expiração da sessão que seja o mais curto possível, baseado no balanceamento dos riscos e requisitos funcionais do negócio. Na maioria dos casos, não deve ser maior que algumas horas.
4.8 Não permitir logins persistentes (sem prazo de expiração), e realizar o encerramento da sessão periodicamente, mesmo quando ela estiver ativa. Isso deve ser feito, especialmente, em aplicações que suportam várias conexões de rede ou que se conectam a sistemas críticos. O tempo de encerramento deve estar em sintonia com os requisitos do negócio e o usuário deve receber notificações suficientes para atenuar os impactos negativos dessa medida.
4.9 Se uma sessão estava estabelecida antes do login, ela deve ser encerrada para que uma nova seja estabelecida após o login.
4.10 Gerar um novo identificador de sessão quando houver uma nova autenticação.
4.11 Não permitir conexões simultâneas com o mesmo identificador de usuário.
4.12 Não expor os identificadores de sessão em URLs, mensagens de erro ou logs. Os identificadores de sessão devem apenas ser encontrados no cabeçalho do cookie HTTP. Por exemplo, não trafegar os identificadores de sessão sob a forma de parâmetros GET.
4.13 Proteger os dados de sessão do lado servidor contra acessos não autorizados por outros usuários do servidor, através da implementação de controles de acesso apropriados no servidor.
4.14 Gerar um novo identificador de sessão e desativar o antigo periodicamente. Isso pode mitigar certos cenários de ataques de sequestro de sessão (session hijacking), quando o identificador de sessão original for comprometido.
4.15 Gerar um novo identificador de sessão caso a segurança da conexão mude de HTTP para HTTPS, como pode ocorrer durante a autenticação. Internamente à aplicação, é recomendável utilizar HTTPS de forma constante em vez de alternar entre HTTP e HTTPS.
4.16 Utilizar mecanismos complementares ao mecanismo padrão de gerenciamento de sessões para operações sensíveis do lado servidor, como no caso de operações de gerenciamento de contas, através da utilização de tokens aleatórios ou parâmetros associados à sessão. Esse método pode ser usado para prevenir ataques do tipo Cross Site Request Forgery (CSRF).
4.17 Utilizar mecanismos complementares ao gerenciamento de sessões para operações altamente sensíveis ou críticas, utilizando tokens aleatórios ou parâmetros em cada requisição em vez de basear-se apenas na sessão.
4.18 Configurar o atributo "secure" para cookies transmitidos através de uma conexão TLS.
4.19 Configurar os cookies com o atributo “HttpOnly”, a menos que seja explicitamente necessário ler ou definir os valores deles através de scripts do lado cliente da aplicação.

6.5 Requisito 5: Controle de acesso

Uma das principais etapas tratamento de acesso do usuário é averiguar se cada solicitação individual de acesso deve ser permitida ou negada. Se os mecanismos de validação de identidade estão funcionando corretamente, a aplicação reconhece se as requisições de acesso são válidas.

Uma aplicação pode suportar inúmeras funções de usuários, cada uma envolvendo diferentes combinações de privilégios específicos. Funções específicas podem implementar limites de transação e outras verificações, todas baseadas na identidade do usuário.

Devido à natureza complexa dos requisitos típicos de controle de acesso, tal função é uma fonte frequente de vulnerabilidades de segurança que permitem a um invasor obter acesso não autorizado a dados e funcionalidades.

Os desenvolvedores costumam fazer suposições erradas sobre como os usuários irão interagir com o aplicativo e frequentemente cometem erros ao dispensar verificações de controle de acesso de algum aplicativo. Analisar essas vulnerabilidades costuma ser trabalhoso, porque essencialmente as mesmas verificações precisam ser repetidas para cada funcionalidade da aplicação.

Devido à prevalência de falhas de controle de acesso, no entanto, esse esforço é uma atividade que pode fornecer benefícios indevidos a um autor de ataque a uma aplicação da web.

Vide abaixo uma lista de subcontroles a serem considerados nessa hipótese.

Subcontroles de Segurança Críticos - Controle de acesso
5.1 Utilizar apenas objetos do sistema que sejam confiáveis, como ocorre com os objetos de sessão do servidor, para realizar a tomada de decisão sobre a autorização de acesso.
5.2 Utilizar um único componente em toda a aplicação Web para realizar o processo de verificação de autorização de acesso. Isto inclui bibliotecas que invocam os serviços externos de autorização.
5.3 Quando ocorrer alguma falha no controle de acesso, ela deve ocorrer de modo seguro, sem que sejam repassados detalhes que possam facilitar ataques direcionados.
5.4 Negar todos os acessos, caso a aplicação não consiga ter acesso às informações contidas na configuração de segurança.
5.5 Garantir o controle de autorização em todas as requisições, inclusive em scripts do lado servidor, "includes" e requisições provenientes de tecnologias do lado cliente.
5.6 Isolar do código da aplicação os trechos de código que contêm lógica privilegiada.
5.7 Restringir o acesso aos arquivos e outros recursos, incluindo aqueles que estão fora do controle direto da aplicação, somente aos usuários autorizados.
5.8 Restringir o acesso às URLs protegidas somente aos usuários autorizados.
5.9 Restringir o acesso às funções protegidas somente aos usuários autorizados.
5.10 Restringir o acesso às referências diretas aos objetos somente aos usuários autorizados.
5.11 Restringir o acesso aos serviços somente aos usuários autorizados.
5.12 Restringir o acesso aos dados da aplicação somente aos usuários autorizados.
5.13 Restringir o acesso aos atributos e dados dos usuários, bem como informações das políticas usadas pelos mecanismos de controle de acesso.
5.14 Restringir o acesso às configurações de segurança relevantes apenas aos usuários autorizados.
5.15 As regras de controle de acesso representadas pela camada de apresentação devem coincidir com as regras presentes no lado servidor.
5.16 Se o estado dos dados deve ser armazenado no lado cliente, utilizar mecanismos de criptografia e verificação de integridade no lado servidor para detectar possíveis adulterações.
5.17 Garantir que os fluxos lógicos da aplicação obedecem às regras de negócio.
5.18 Limitar o número de transações que um único usuário ou dispositivo pode executar em determinado período. As transações por período devem estar acima das necessidades reais do negócio, mas abaixo o suficiente para impedir ataques automatizados.
5.19 Utilizar o campo “referer” do cabeçalho somente como forma de verificação suplementar. Ele não deve ser usado sozinho como forma de validação de autorização, pois pode ter o valor adulterado.

Subcontroles de Segurança Críticos - Controle de acesso
5.20 Se for permitida a existência de sessões autenticadas por longos períodos, fazer a revalidação periódica da autorização do usuário para garantir que os privilégios não foram modificados e, caso tenham sido, realizar o registro em log do usuário e exigir nova autenticação.
5.21 Implementar a auditoria das contas de usuário e assegurar a desativação de contas não utilizadas. Por exemplo, a conta deve ser desativada não mais do que 30 dias após a expiração da senha.
5.22 A aplicação deve dar suporte à desativação de contas e ao encerramento das sessões quando terminar a autorização do usuário - por exemplo, quando ocorrer alguma alteração dos dados do usuário, situação profissional, processos de negócio etc.
5.23 As contas de serviço ou contas de suporte a conexões provenientes ou destinadas a serviços externos devem possuir o menor privilégio possível.
5.24 Criar uma Política de Controle de Acesso para documentar as regras de negócio da aplicação, tipos de dados e critérios ou processos de autorização, para que os acessos possam ser devidamente concedidos e controlados. Isso inclui identificar requisitos de acessos - tanto para os dados, como para os recursos do sistema.
5.25 Configurar os cabeçalhos HTTP para o uso do C.O.R.S (Cross-Origin Resource Sharing), utilizado para permitir ou bloquear o acesso a conteúdo como AJAX, fontes em outros sites, de acordo com as regras de negócio da aplicação.

6.6 Requisito 6: Criptografia

As aplicações necessitam ser desenhadas com uma arquitetura de criptografia forte para proteger seus dados de acordo com sua classificação. Criptografar todas as informações pode ser inviável, mas não criptografar nenhum dado é um grande risco assumido. Um equilíbrio deve ser buscado, normalmente durante a fase de desenho e projeto da aplicação. Desenhar e desenvolver a arquitetura de criptografia durante o desenvolvimento da aplicação, ou após a aplicação estar pronta, inevitavelmente irá custar muito mais que simplesmente construí-la de forma segura desde o início do desenvolvimento.

Alguns requisitos de alto nível devem ser observados:

- Os módulos de criptografia devem apresentar erros de maneira segura e ser tratados corretamente.
- Um gerador de número aleatórios seguro deve ser utilizado.
- O acesso às chaves de criptografia deve ser gerenciado de forma segura.

Ainda sobre criptografia, deve ser observado que o item 6.5, listado na tabela abaixo, deve ser implementado de forma atenta e crítica pelos gestores da área de tecnologia da informação. Isso porque o Federal Information Processing Standard (FIPS) 140-3 foi aprovado em 22 de março de 2019, tendo entrado em vigor em 22 de setembro de 2019 , em substituição ao FIPS 140-2. Embora os respectivos certificados de validação do FIPS 104-3 ainda não tenham sido emitidos, há a expectativa de que isso ocorra em breve.

Subcontroles de Segurança Críticos - Criptografia
6.1 Todas as funções de criptografia utilizadas para proteger dados sensíveis dos usuários da aplicação devem ser implantadas em um sistema confiável - neste caso, o servidor.
6.2 A senha mestra deve ser protegida contra acessos não autorizados.
6.3 Quando ocorrer alguma falha nos módulos de criptografia, permitir que ela ocorra de modo seguro.

Subcontroles de Segurança Críticos - Criptografia
6.4 Todos os números, nomes de arquivos, GUIDs e strings aleatórias devem ser gerados usando um módulo criptográfico com gerador de números aleatórios, somente se os valores aleatórios gerados forem impossíveis de serem deduzidos.
6.5 Os módulos de criptografia usados pela aplicação devem ser compatíveis com a FIPS 140-2 ou com um padrão equivalente (http://csrc.nist.gov/groups/STM/cmvp/validation.html).
6.6 Estabelecer e utilizar uma política e um processo que defina como é realizado o gerenciamento das chaves criptográficas.

6.7 Requisito 7: Tratamento de erros e logs

O principal objetivo do tratamento de erros e logs é fornecer informação útil para usuários, administradores e times de resposta a incidentes. Devem-se buscar logs de alta qualidade, com mais sinal do que ruído, de forma a evitar a criação de uma quantidade massiva de logs.

Logs com informação de qualidade normalmente possuem dados sensíveis e devem ser protegidos conforme a Norma Complementar nº 21 IN01/DSIC/GSIPR, que trata de diretrizes para o Registro de Eventos, Coleta e Preservação de Evidências de Incidentes de Segurança. Por geralmente possuírem informações bastante sensíveis, os logs também podem ser um alvo bem atraente para os atacantes.

É importante também garantir que a aplicação apresente erros de forma segura, e que esses erros não vazem informações desnecessariamente.

Subcontroles de Segurança Críticos - Tratamento de erros e logs
7.1 Não expor informações sensíveis nas repostas de erros, inclusive detalhes de sistema, identificadores de sessão ou informação da conta do usuário.
7.2 Usar mecanismos de tratamento de erros que não mostrem informações de depuração (debug) ou informações da pilha de exceção.
7.3 Usar mensagens de erro genéricas e páginas de erro personalizadas.
7.4 A aplicação deve tratar os erros sem se basear nas configurações do servidor.
7.5 A memória alocada deve ser liberada de modo apropriado quando ocorrerem condições de erro.
7.6 O tratamento de erros lógicos associados com os controles de segurança deve, por padrão, negar o acesso.
7.7 Todos os controles de log devem ser implementados em um sistema confiável, por exemplo, centralizar todo o processo no servidor.
7.8 Os controles de log devem dar suporte tanto para os casos de sucesso como os de falha relacionados com os eventos de segurança.
7.9 Garantir que os logs armazenam eventos importantes.
7.10 Garantir que as entradas de log que incluam dados nos quais não se confia não sejam executadas como código-fonte na interface de visualização de logs.
7.11 Restringir o acesso aos logs apenas para pessoal autorizado.
7.12 Utilizar uma rotina centralizada para realizar todas as operações de log.
7.13 Não armazenar informações sensíveis nos registros de logs, como detalhes desnecessários do sistema, identificadores de sessão e senhas
7.15 Garantir o uso de algum mecanismo que conduza (ou facilite) o processo de análise de logs.

Subcontroles de Segurança Críticos - Tratamento de erros e logs
7.16 Colete logs do provedor de serviços, onde houver suporte. Exemplos de implementações incluem coleta de eventos de autenticação e autorização, eventos de criação e de descarte de dados e eventos de gestão de usuários.

6.8 Requisito 8: Proteção de dados

É necessário que a aplicação proteja os dados tratados por ela, de forma que o acesso às suas informações se restrinja ao mínimo necessário. Além disso, é importante adotar controles de segurança ao armazenar as informações para garantir que os dados necessários sejam criptografados e que informações temporárias ou registradas em cache sejam eliminadas quando não forem mais utilizadas.
É preciso também evitar que informações sensíveis sejam transportadas de forma insegura, e evitar que informações sensíveis sobre a aplicação estejam visíveis aos usuários finais.

Subcontroles de Segurança Críticos - Proteção de dados
8.1 Implementar uma política de privilégio mínimo, restringindo aos usuários apenas às funcionalidades, dados e informações do sistema que são necessárias para executarem suas tarefas.
8.2 Proteger contra acesso não autorizado todas as cópias temporárias ou registradas em cache que contenham dados sensíveis e estejam armazenadas no servidor; excluir esses arquivos logo que não forem mais necessários.
8.3 Criptografar informações altamente sensíveis quando armazenadas – como dados de verificação de autenticação – mesmo que estejam no lado servidor, usando sempre algoritmos conhecidos, padronizados e bem testados. Consulte a seção que trata sobre “Práticas de Criptografia” para orientações adicionais.
8.4 Proteger o código-fonte presente no servidor para que não seja acessado por usuários não autorizados.
8.5 Não armazenar senhas, strings de conexão ou outras informações confidenciais em texto claro/legível ou em qualquer forma criptograficamente insegura no lado cliente. Isso é válido também quando há utilização de formatos inseguros, como MS viewstate ou código compilado que é executado no lado cliente.
8.6 Remover comentários do código de produção que podem ser acessados pelos usuários e podem revelar detalhes internos do sistema ou outras informações sensíveis.
8.7 Remover aplicações desnecessárias e documentação do sistema que possam revelar informações importantes para os autores de ataques.
8.8 Não incluir informações sensíveis nos parâmetros de requisição HTTP GET.
8.9 Desativar a funcionalidade de autocompletar nos formulários que contenham informações sensíveis, inclusive no formulário de autenticação.
8.10 Desativar a cache realizada no lado cliente das páginas que contenham informações sensíveis. O parâmetro "Cache-Control: no-store" pode ser usado em conjunto com o controle definido no cabeçalho HTTP “Pragma: no-cache”, que é menos efetivo, porém compatível com HTTP/1.0 .
8.11 A aplicação deve dar suporte à remoção de dados sensíveis quando estes não forem mais necessários - como, por exemplo, informação pessoal ou dados financeiros.
8.12 Implementar mecanismos de controle de acesso apropriados para dados sensíveis armazenados no servidor. Isto inclui dados em cache, arquivos temporários e dados que devem ser acessíveis somente por usuários específicos do sistema.
8.13 Prover mecanismos que garantam a anonimização dos dados pessoais e dados pessoais sensíveis, conforme a Lei Geral de Proteção de Dados Pessoais brasileira.

6.9 Requisito 9: Segurança nas comunicações

Para transmissão de dados e informações, é ideal que se utilizem canais de comunicação seguros, o que pode ser implementado utilizando o protocolo TLS ou outra cifra forte. Devem-se utilizar de recomendações mais recentes de boas práticas de configuração para habilitar e ordenar os algoritmos e cifras preferenciais.

Algoritmos e cifras fracos, ou perto de serem considerados obsoletos, devem ser considerados somente em último caso. Algoritmos e cifras conhecidamente inseguros ou obsoletos não devem ser utilizados.

Subcontroles de Segurança Críticos - Segurança nas comunicações
9.1 Utilizar criptografia na transmissão de todas as informações sensíveis. Isto deve incluir TLS para proteger a conexão e deve ser complementado com criptografia de arquivos que contém dados sensíveis ou conexões que não usam o protocolo HTTP.
9.2 Os certificados TLS devem ser válidos, possuir o nome de domínio correto, não estar expirados e ser instalados com certificados intermediários, quando necessário.
9.3 Quando ocorrer alguma falha nas conexões TLS, o sistema não deve fornecer uma conexão insegura.
9.4 Utilizar conexões TLS para todo o conteúdo que requerer acesso autenticado ou que contenha informação sensível.
9.5 Utilizar TLS para conexões com sistemas externos que envolvam funções ou informações sensíveis.
9.6 Utilizar um padrão único de implementação TLS configurado de modo apropriado.
9.7 Especificar a codificação dos caracteres para todas as conexões.
9.8 Filtrar os parâmetros que contenham informações sensíveis, provenientes do “HTTP
referer”, nos links para sites externos.

6.10 Requisito 10: Configuração do sistema

A configuração de uma aplicação recém-lançada deve ser segura o suficiente para estar publicada na Internet, o que significa uma configuração segura por padrão. É importante certificar que a aplicação possua:

- Um ambiente de construção seguro, repetível e automatizado;
- Biblioteca de terceiros reforçada, gerenciamento de dependência e configuração de forma que componentes desatualizados ou inseguros não sejam incluídos na aplicação;
- Uma configuração baseada no conceito Security by Default – a qual possui, por padrão, configurações seguras, de forma que a escolha por configurações menos rígidas de segurança seja uma escolha de administradores e usuários.

Subcontroles de Segurança Críticos - Configuração do sistema
10.1 Garantir que os servidores, frameworks e componentes do sistema estão executando a última versão aprovada.
10.2 Garantir que os servidores, frameworks e componentes do sistema possuem as atualizações mais recentes e seguras aplicadas para a versão em uso.
10.3 Desativar a listagem de diretórios.
10.4 Restringir, para o mínimo possível, os privilégios do servidor Web, dos processos e das contas de serviços.
10.5 Quando ocorrerem exceções no sistema, garantir que as falhas ocorram de modo seguro.

Subcontroles de Segurança Críticos - Configuração do sistema
10.6 Remover todas as funcionalidades e arquivos desnecessários.
10.7 Remover código de teste ou qualquer funcionalidade desnecessária para o ambiente de produção, antes de instalar o sistema no servidor de produção.
10.8 Prevenir a divulgação da estrutura de diretórios, impedindo que robôs de busca façam indexação de arquivos sensíveis, através da configuração do arquivo "robots.txt". Os diretórios que não devem ser acessados por estes indexadores devem ser colocados em um diretório isolado. Assim, apenas é necessário negar o acesso ao diretório pai definido no arquivo “robots.txt”, evitando ter que negar o acesso a cada diretório individualmente.
10.9 Definir quais métodos HTTP, GET ou POST a aplicação irá suportar, e se serão tratados de modo diferenciado nas diversas páginas da aplicação.
10.10 Desativar as extensões HTTP desnecessárias como, por exemplo, o WebDAV. Caso seja necessário o uso de alguma extensão HTTP com o propósito de suportar a manipulação de arquivos, utilize um mecanismo de autenticação conhecido, padronizado e bem testado.
10.11 Se o servidor processa tanto requisições HTTP 1.0 como HTTP 1.1, certificar-se de que ambos são configurados de modo semelhante ou assegure que qualquer diferença existente seja compreendida - como, por exemplo, o manuseio de métodos HTTP estendidos.
10.12 Remover informações desnecessárias presentes nos cabeçalhos de resposta HTTP e que podem estar relacionadas com o sistema operacional, versão do servidor web e frameworks de aplicação.
10.13 O armazenamento da configuração de segurança para a aplicação deve ser capaz de ser produzida de forma legível para dar suporte à auditoria.
10.14 Implementar um sistema de gestão de ativos para manter o registro dos componentes e programas.
10.15 Isolar o ambiente de desenvolvimento da rede de produção e conceder acesso somente para grupos de desenvolvimento e testes. É comum os ambientes de desenvolvimento serem configurados de modo menos seguro do que os ambientes de produção. Deste modo, os autores de ataques podem usar essa diferença para descobrir vulnerabilidades comuns ou encontrar formas de exploração.
10.16 Implementar um sistema de controle de mudanças para gerenciar e registrar as alterações no código, tanto de desenvolvimento, como dos sistemas em produção.
10.17 Rejeitar requisições que utilizem métodos (ou verbos) HTTP além do GET ou POST que não são utilizados pela aplicação (Verb tempering attacks), como, por exemplo, os verbos TRACE e OPTIONS e demais métodos.

6.11 Requisito 11: Segurança em Banco de Dados

A coleta e o tratamento de dados são tarefas fundamentais para que as organizações consigam, através de análises específicas, traçar metas e alcançar o objetivo final de atendimento aos usuários.
Desta forma, é fundamental que as organizações consigam garantir a proteção dos dados armazenados em banco de dados contra acessos indevidos, ações de hackers e incidentes técnicos.

Subcontroles de Segurança Críticos - Segurança em Banco de Dados
11.1 Usar consultas parametrizadas fortemente tipadas.
11.2 Utilizar validação de entrada e codificação de saída e assegurar a abordagem de meta caracteres. Se houver falha, o comando não deverá ser executado no banco de dados.
11.3 Certificar-se de que as variáveis são fortemente tipadas.

Subcontroles de Segurança Críticos - Segurança em Banco de Dados
11.4 Realizar a codificação (escaping) de meta caracteres em instruções SQL.
11.5 A aplicação deve usar o menor nível possível de privilégios ao acessar o banco de dados.
11.6 Usar credenciais seguras para acessar o banco de dados.
11.7 Não incluir strings de conexão na aplicação. As strings de conexão devem estar em um arquivo de configuração separado, armazenado em um sistema confiável e as informações devem ser criptografadas.
11.8 Usar procedimentos armazenados (stored procedures) para abstrair o acesso aos dados e permitir a remoção de permissões das tabelas no banco de dados.
11.9 Encerrar a conexão assim que possível.
11.10 Remover ou modificar senhas-padrão de contas administrativas. Utilizar senhas robustas (pouco comuns ou difíceis de deduzir) ou implementar autenticação de múltiplos fatores. Desativar qualquer funcionalidade desnecessária no banco de dados, como “stored procedures” ou serviços não utilizados. Instalar o conjunto mínimo de componentes ou de opções necessárias (redução da superfície de ataque).
11.11 Eliminar o conteúdo desnecessário incluído por padrão pelo fornecedor como esquemas e bancos de dados de exemplo.
11.12 Desativar todas as contas criadas por padrão e que não sejam necessárias para suportar os requisitos de negócio.
11.13 A aplicação deve conectar-se ao banco de dados com diferentes credenciais de segurança para cada tipo de necessidade - como, por exemplo, usuário, somente leitura, convidado ou administrador.
11.14 O Banco de Dados deve ser hospedado em um servidor diferente (virtualizados ou não) dos demais serviços.
11.15 Instalar, tão logo quanto possível, patches e hotfixes de versões mais atuais validadas e homologadas.
11.16 Encriptar os Backups de modo a proteger o sigilo das informações em caso de perda ou extravio.

6.12 Requisito 12: Gerenciamento de Arquivos

É comum que as aplicações recebam arquivos do usuário. Porém, os arquivos fornecidos pelos usuários também são um vetor de ataque bastante utilizados por atacantes. Por meio do envio de arquivos maliciosos, por exemplo, é possível obter novas formas de interagir com o servidor e até executar códigos de forma remota.

Assim, é importante adotar um gerenciamento de arquivos que trate, manipule e armazene de forma segura os arquivos que possam ser manipulados pelos usuários.

Subcontroles de Segurança Críticos - Gerenciamento de Arquivos
12.1 Não repassar dados fornecidos pelos usuários diretamente a uma função de inclusão dinâmica.
12.2 Solicitar autenticação antes de permitir que seja feito o carregamento de arquivos.
12.3 Limitar os tipos de arquivos que podem ser enviados para aceitar somente os necessários ao propósito do negócio.
12.4 Validar se os arquivos enviados são do tipo esperado, através da validação dos cabeçalhos, pois realizar a verificação apenas pela extensão é insuficiente.
12.5 Não salvar arquivos no mesmo diretório de contexto da aplicação Web. Os arquivos devem ser armazenados no servidor de conteúdos ou no banco de dados.
12.6 Prevenir ou restringir o carregamento de qualquer arquivo que possa ser interpretado ou executado pelo servidor Web.

Subcontroles de Segurança Críticos - Gerenciamento de Arquivos
12.7 Desativar privilégios de execução nos diretórios de armazenamento de arquivos.
12.8 Implantar o carregamento seguro nos ambientes UNIX por meio da montagem do diretório de destino como uma unidade lógica, usando o caminho associado ou o ambiente de "chroot".
12.9 Ao referenciar arquivos, usar uma lista de permissões (whitelist) de nomes e de tipos de arquivos permitidos. Realizar a validação do valor do parâmetro passado e, caso ele não corresponda ao que é esperado, rejeitar a entrada ou utilizar um valor padrão.
12.10 Não transmitir, sem nenhum tipo de tratamento, os dados informados pelo usuário a redirecionamentos dinâmicos. Se isso for necessário, o redirecionamento deverá aceitar apenas URLs relativas e validadas.
12.11 Não passar caminhos de diretórios ou de arquivos em requisições. Usar algum mecanismo de mapeamento desses recursos para índices definidos em uma lista pré-definida de caminhos.
12.12 Nunca enviar o caminho absoluto do arquivo para o lado cliente de uma aplicação ou para o usuário.
12.13 Certificar-se de que os arquivos da aplicação e os recursos estão definidos somente com o atributo de leitura.
12.14 Verificar os arquivos que os usuários submeterem através do mecanismo de carregamento em busca de vírus e malwares.
12.15 Limitar o tamanho máximo aceito para uploads de arquivos no servidor.

6.13 Requisito 13: Gerenciamento de memória

Informações sensíveis ou úteis a um atacante podem ser armazenadas ou acessadas na memória por meio de técnicas e funções que permitem esse acesso. Além disso, atacantes podem se utilizar da técnica de transbordamento de dados (buffer overflow) -na qual o programa, ao escrever dados em um buffer, ultrapassa o limite de tamanho do buffer, sobrescrevendo a memória adjacente – para acessar endereços de memória ou injetar códigos maliciosos.

Nesse contexto, é de suma importância que os desenvolvedores implementem técnicas e controles que garantam que a aplicação fará o gerenciamento e o uso adequado dos recursos de memória, visando com isso a evitar que processos legítimos sejam impactados por ações internas ou externas que possam comprometer o comportamento esperado do sistema ou degradar a performance da aplicação.

Subcontroles de Segurança Críticos - Gerenciamento de memória
13.1 Utilizar controle de entrada/saída para os dados que não sejam confiáveis.
13.2 Verificar se o buffer é tão grande quanto o especificado.
13.3 Ao usar funções que aceitem determinado número de bytes para realizar cópias, como strncpy(), estar ciente de que, se o tamanho do buffer de destino for igual ao tamanho do buffer de origem, ele não pode encerrar a sequência de caracteres com valor nulo (null).
13.4 Verificar os limites do buffer caso as chamadas à função sejam realizadas em ciclos e verificar se não há nenhum risco de ocorrer gravação de dados além do espaço reservado.
13.5 Truncar todas as strings de entrada para um tamanho razoável antes de passá-las para as funções de cópia e concatenação.
13.6 Na liberação de recursos alocados para objetos de conexão, identificadores de arquivo etc., não contar com o "garbage collector" e realizar a tarefa explicitamente.
13.7 Usar pilhas não executáveis, quando disponíveis.

Subcontroles de Segurança Críticos - Gerenciamento de memória
13.8 Evitar o uso de funções reconhecidamente vulneráveis, como printf(), strcat(), strcpy() etc.
13.9 Liberar a memória alocada de modo apropriado após concluir a sub-rotina (função/método) e em todos os pontos de saída.

6.14 Requisito 14: Práticas Gerais de Codificação

Com o advento da metodologia de desenvolvimento "DevOps", os procedimentos de segurança, adotados outrora pela equipe de arquitetura de segurança, devem ser readequados para que a rotina de análise de segurança de aplicações seja introduzida nesse novo contexto de desenvolvimento ágil.

Os procedimentos de análise de segurança são vistos normalmente como inflexíveis e excessivamente assertivos pelos profissionais de desenvolvimento de aplicações, já que estes podem, em diversas oportunidades, argumentar que existem várias maneiras de resolver um problema. De fato, quando se aborda a arquitetura de softwares e aplicações, não existe solução única e simples para determinado problema.

É provável que uma função específica de uma aplicação web seja revisada continuamente ao longo de sua vida útil, mas a arquitetura geral raramente mudará e pode passar por evoluções graduais. O mesmo quadro acontece quando se fala de arquitetura de segurança. Ao desenvolver uma aplicação web com controles de segurança desde sua concepção, a organização interessada provavelmente irá poupar tempo e dinheiro, reduzindo as chances da ocorrência de incidentes de segurança.

Subcontroles de Segurança Críticos - Práticas Gerais de Codificação
14.1 Para tarefas comuns, utilizar sempre código testado, gerenciado e aprovado, ao invés de criar código novo e não gerenciado.
14.2 Utilizar APIs que executem tarefas específicas para realizar operações do sistema operacional. Não permitir que a aplicação execute comandos diretamente no sistema operacional, especialmente através da utilização de "shells" de comando iniciadas pela aplicação.
14.3 Utilizar mecanismos de verificação de integridade por "checksum" ou "hash" para verificar a integridade do código interpretado, bibliotecas, arquivos executáveis e arquivos de configuração.
14.4 Utilizar mecanismos de bloqueio para evitar requisições simultâneas para a aplicação ou utilizar um mecanismo de sincronização para evitar condições de concorrência (race conditions).
14.5 Proteger as variáveis compartilhadas e os recursos contra acessos concorrentes inapropriados.
14.6 Instanciar explicitamente todas as variáveis e dados persistentes durante a declaração, ou antes da primeira utilização.
14.7 Quando a aplicação tiver que ser executada com privilégios elevados, aumentar os privilégios o mais tarde possível e revogá-los logo que seja possível.
14.8 Evitar erros de cálculo decorrentes da falta de entendimento da representação interna da linguagem de programação usada e de como é realizada a interação com os aspectos de cálculo numérico.
14.9 Prestar bastante atenção nas discrepâncias de tamanho de byte, precisão, distinções de sinal (signed/unsigned), truncamento, conversão e "casting" entre os tipos, cálculos que devolvam erros do tipo "not-a-number" e, também, como a linguagem de
programação trata a representação interna de números muito grandes ou muito pequenos.

Subcontroles de Segurança Críticos - Práticas Gerais de Codificação
14.10 Não transferir diretamente dados fornecidos pelo usuário para qualquer função de execução dinâmica sem antes realizar o tratamento dos dados de modo adequado.
14.11 Restringir a geração e a alteração de código por parte dos usuários.
14.12 Revisar todas as aplicações secundárias, códigos e bibliotecas de terceiros para determinar a necessidade do negócio e validar as funcionalidades de segurança, uma vez que estas podem introduzir novas vulnerabilidades.
14.13 Implementar atualizações de modo seguro. Se a aplicação precisar realizar atualizações automáticas, utilizar mecanismos de assinatura digital para garantir a integridade do código e garantir que os clientes façam a verificação da assinatura após descarregarem as atualizações. Usar canais criptografados para transferir o código a partir do host do servidor.

7. Exceções

Exceções às regras deste Procedimento serão tratadas pela Gerência de Infraestrutura e Operações juntamente com os Gestores apropriados a cada caso.

8. Utilização de Dados Pessoais (LGPD)

N/A

9. Anexos

Não há anexos

Anexo 3: Guia de Utilização GIT

1. Objetivo/Introdução

Esse documento tem por objetivo oferecer um guia padronizado de utilização do GIT para usuários internos e externos.

2. Pré-requisitos para Execução das Atividades

O usuário do GIT deve ter acesso prévio criado pelo administrador.

3. Referências

https://git-scm.com/

4. Descrição das Atividades

4.1 Instalação

O GIT deve ser instalado na máquina localmente a partir dos link abaixo:

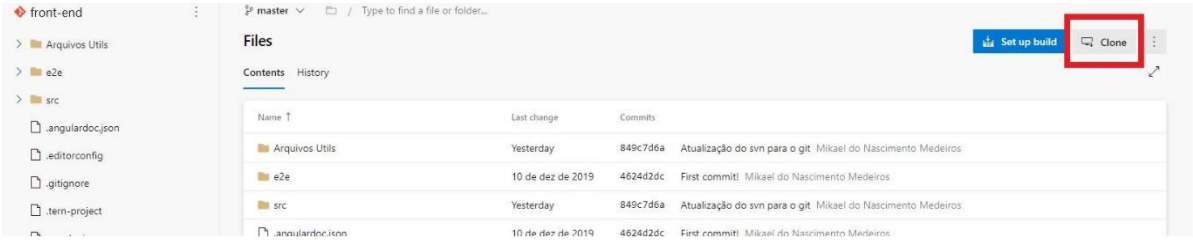
https://git-scm.com/downloads

Para realizar a configuração corretamente, deve ser o guia padrão de instalação do GIT a partir do link:

https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

4.2 Clonando projetos

Passo 1: Para ter acesso à URL do projeto acesso o repositório na Azure Dev ops e clique em "Clone" no canto superior direito.



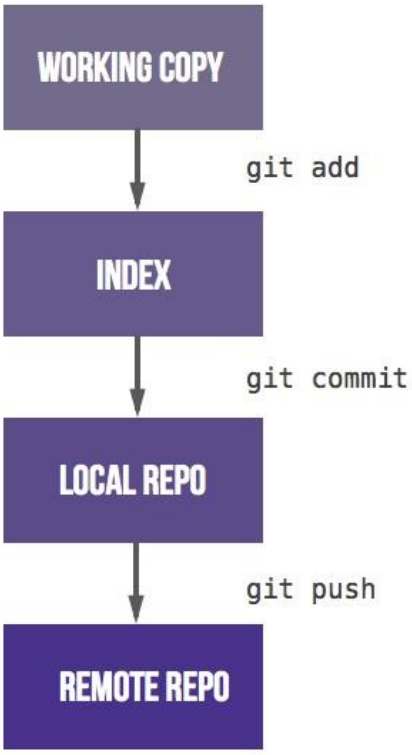
Passo 2: Para copiar a URL do projeto basta clonar no botão "*Copy clone URL to clipboard*" com na imagem abaixo.

Passo 3: Para baixar o projeto para a máquina local é necessário acessar o diretório desejado por linha de comando e executar o seguinte comando.

```
git clone + url do repositório remoto
```

4.3 Fluxo de *commit* e *push*

O GIT possui três áreas para três estágios diferentes no processo de *commit*.



A primeira é o **Working Directory** onde ficam as alterações no código.

Para verificar os arquivos que foram alterados digite o comando:

```
git status
```

Os arquivos alterados serão destacados na cor vermelha conforme imagem abaixo:

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseProjeto.java
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseSituacao.java

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>
```

4.3.1 GIT add

Antes de fazer o commit é necessário adicionar os arquivos alterados no **Index**, que funciona como uma área pré-commit.

Adicionando arquivos individuais ao index:

Para adicionar um único arquivo ao index na intenção de fazer o commit apenas dele execute comando abaixo:

git add + path do arquivo

A imagem abaixo mostra como o path deve ser informado.

```
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseProjeto.java
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseSituacao.java

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git add src/main/java/br/org/cni/sgf/domain/TabelaRepasseProjeto.java
```

O arquivo adicionado ficará destacado na cor verde conforme imagem abaixo.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseProjeto.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseSituacao.java

PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>
```

Adicionando todos os arquivos alterados ao index ao mesmo tempo:

Para adicionar todos os arquivos alterados ao mesmo tempo basta digitar o comando:

git add

Todos os arquivos ficarão destacados na cor verde conforme a imagem abaixo:

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git add .
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseProjeto.java
        modified:   src/main/java/br/org/cni/sgf/domain/TabelaRepasseSituacao.java

PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>
```

4.3.2 GIT commit

Agora que todos os arquivos estão no index ou seja estão prontos para o commit, basta digitar o comando abaixo:

`git commit -m + "mensagem do commit"`

Todo commit precisa ter uma mensagem resumida do contexto das alterações e isso é possível através da flag -m seguido da mensagem entre aspas.

Após a execução do commit será informada a quantidade de arquivos alterados e inserções no código conforme a imagem abaixo.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git commit -m "mensagem do commit"
[master 01ae13d] mensagem do commit
 2 files changed, 2 insertions(+)
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>
```

4.3.3 GIT push

Por enquanto os arquivos estão no Local Repository e é necessário subir para o repositório na Azure Dev Ops.

Para fazer isso se utiliza o comando push acompanhado do nome repositório e do nome da branch.

`git push + nome repositorio + nome da branch`

O nome default para os repositórios do git é "origin" porém esse nome pode ser outro. No caso do exemplo abaixo o default foi utilizado.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao> git push origin master
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 12 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 804 bytes | 268.00 KiB/s, done.
Total 12 (delta 5), reused 0 (delta 0)
remote: Analyzing objects... (12/12) (193 ms)
remote: Storing packfile... done (115 ms)
remote: Storing index... done (26 ms)
To https://dev.azure.com/CNI-STI/Sistema%20de%20Gest%C3%A3o%20de%20Apoio%20Financeiro%202.0/_git/sgf-administracao
   30d9af9..01ae13d  master -> master
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>
```

4.4 Atualizando o Código com o Pull

É muito importante manter o repositório local atualizado com o remoto. Para esse procedimento é necessário utilizarmos o seguinte comando:

git pull + nome do repositório + nome da branch

No exemplo abaixo estamos atualizando nossa branch "master" local de acordo com a branch master no repositório remoto.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git pull origin master
```

Caso existam atualizações, os arquivos atualizados serão listados indicando a inclusão ou remoção de linhas de código. A imagem abaixo nos dá um exemplo dessa situação.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git pull origin master
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 14 (delta 5), reused 14 (delta 5), pack-reused 0
Unpacking objects: 100% (14/14), 1.93 KiB | 34.00 KiB/s, done.
From https://github.com/rodrigopereiradev/excel-export
 * branch            master     -> FETCH_HEAD
   ff16211..a797c13  master     -> origin/master
Updating ff16211..a797c13
Fast-forward
 .../exportexcel/planilhas/PessoaPlanilha.java      |  81 +++++++++++++++
 .../exportexcel/planilhas/Planilha.java            |  64 ++++++++++++
 .../exportexcel/services/PlanilhaService.java      | 111 +++--------------------
 3 files changed, 160 insertions(+), 96 deletions(-)
 create mode 100644 src/main/java/br/com/rodrigopereira/exportexcel/planilhas/PessoaPlanilha.java
 create mode 100644 src/main/java/br/com/rodrigopereira/exportexcel/planilhas/Planilha.java
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

Caso não existam atualizações a mensagem "Already up to date" será apresentada conforme exemplo abaixo.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git pull origin master
From https://github.com/rodrigopereiradev/excel-export
 * branch            master     -> FETCH_HEAD
Already up to date.
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

4.5 Criação e mudanças de branches

4.5.1 Criando uma nova branch

Para criarmos uma branch é bem simples:

git branch + nome da nova branch

A imagem abaixo tem um exemplo prático da execução do comando.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git branch branch_teste
```

Para verificar a lista de branchs locais existentes basta executar o seguinte comando:

git branch

Após a execução do comando uma lista de branchs será exibida. A branch na qual o usuário estiver será destacada com uma asteriscos e na cor verde conforme exemplo abaixo.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git branch
  branch_teste
* master
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

4.5.2 Mudando de branch

Para mudar de branch basta digitar o comando abaixo.

git checkout + nome da branch

A imagem abaixo exemplifica o comando.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git checkout branch_teste
```

Uma mensagem informando que houve a mudança de branch será informada.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git checkout branch_teste
Switched to branch 'branch_teste'
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

4.5.3 Criando e mudando para a nova branch ao mesmo tempo

É possível o usuário criar uma branch e mudar para ela ao mesmo tempo. Para isso basta utilizar a flag -b ao executar o comando abaixo.

git checkout -b + nome da nova branch

O exemplo abaixo demonstra como o comando é executado com uma mensagem informando que houve uma mudança para a nova branch.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git checkout -b nova_branch
Switched to a new branch 'nova_branch'
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

4.5.4 Criando branches no repositório remoto

Agora que criamos uma branch local é preciso criar uma versão remota dela. Para isso basta executar o comando abaixo.

git push + nome do repositorio remoto + nome da branch local

Na imagem abaixo mostra um exemplo de como dever ser executado o comando. Ressaltamos que "origin" é o nome default do repositório remoto e ele pode mudar de acordo com o projeto.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git push origin nova_branch
```

Após execução alguma informações serão apresentadas e uma delas é de que uma nova branch foi criada no repositório.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git push origin nova_branch
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'nova_branch' on GitHub by visiting:
remote:      https://github.com/rodrigopereiradev/excel-export/pull/new/nova_branch
remote:
To https://github.com/rodrigopereiradev/excel-export.git
 * [new branch]      nova_branch -> nova_branch
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

Para verificarmos a lista de branch remotas basta digitarmos o comando normal acompanhado da flag **-a**.

```
git branch -a
```

As branchs remotas serão apresentadas destacadas na cor vermelha conforme o exemplo abaixo.

```
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export> git branch -a
  branch_teste
  master
* nova_branch
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
  remotes/origin/nova_branch
PS C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\excel-export>
```

Caso o comando não esteja listando todas as branchs remotas você pode rodar o comando abaixo para atualizar essa consulta.

```
git fetch
```

A imagem abaixo mostra a execução do fetch.

```
C:\Users\rodrigo.pereira\desenvolvimento\git_repositories\sgf-administracao>git fetch
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
sh: warning: setlocale: LC_ALL: cannot change locale (enu): No such file or directory
remote: Azure Repos
remote: Found 3 objects to send. (1 ms)
Unpacking objects: 100% (3/3), 2.46 KiB | 84.00 KiB/s, done.
From https://dev.azure.com/CNI-STI/Sistema%20de%20Gest%C3%A3o%20de%20Apoio%20Financeiro%202.0/_git/sgf-administracao
   01ae13d..60662a0  master     -> origin/master
 * [new branch]      develop    -> origin/develop
 * [new branch]      feature/#362_informacoes_dr -> origin/feature/#362_informacoes_dr
```

Observe que o comando mapeou duas novas branches.

4.6 Erros GIT

Caso ocorra o erro "*File name to long*" abrir o power shell como administrador e executar o seguinte comando: git config --system core.longpaths true. Tente clonar o repo novamente.

5. Utilização de Dados Pessoais (LGPD)

Não contempla a utilização de Dados Pessoais, conforme a Lei Geral de Proteção de Dados Pessoais (LGPD).

6. Anexos

N/A.

7. Controle de Versão

Histórico (para uso da Gerência de Governança):

Versão	Data Publicação	Elaborador	Motivo da Versão

Aprovação da Versão Atual:

Nome	Cargo

Anexo 4: Padrões de Versionamento

1. Objetivo

Ser uma instrução de trabalho para padronizar o versionamento de código com a utilização da ferramenta GIT e AzureDevops.

2. Pré-requisitos para Execução das Atividades

Projeto e Repositório criados na Azure DevOps.

Conhecimento de GIT. Esse conhecimento pode ser adquirido acessando o nosso GUIA.

3. Referências

Review and merge code with pull requests - Azure Repos | Microsoft Docs

Azure Repos Git Documentation | Microsoft Docs

Fork Azure DevOps

4. Definições

1. Versionamento 4 dígitos

O versionamento do código fonte será feito com quatro dígitos para melhor controlar quais funcionalidades estão presente naquela versão do projeto.

a) 0 - Mudanças negociais
b) 1 - Novos módulos
c) 2 - Novas Funcionalidades
d) 3 - Correções

a. Mudança negocial

Mudança de versão comercial definida principalmente por quem está à frente do projeto, como Gerentes, Product Owner, etc.

b. Novos módulos

Conjunto de novas funcionalidades que englobam regas de negócios ainda não desenvolvidas no sistema.

c. Novas Funcionalidades

Criação ou alteração estrutural de funcionalidades dentro de um modulo, ou seja, nova ou alteração da regra de negócio de um modulo previamente desenvolvido.

d. Correções

Correções de bugs e mudanças não estruturais, como botões e posições de elementos na tela.

2. HOTFIX

São branchs de correções pontuais como bugs e pequenos ajustes (mudança de uma label, um posicionamento do botão, uma margem, etc) que não alteram as estruturas da aplicação.

3. FEATURE

São branchs de desenvolvimento de regras de negócio dentro da aplicação.

4. MAIN

É a branch principal do projeto e nela que fica o código mais testado e confiável do projeto, é nesta branch que se encontra o versionamento tagueado do projeto.

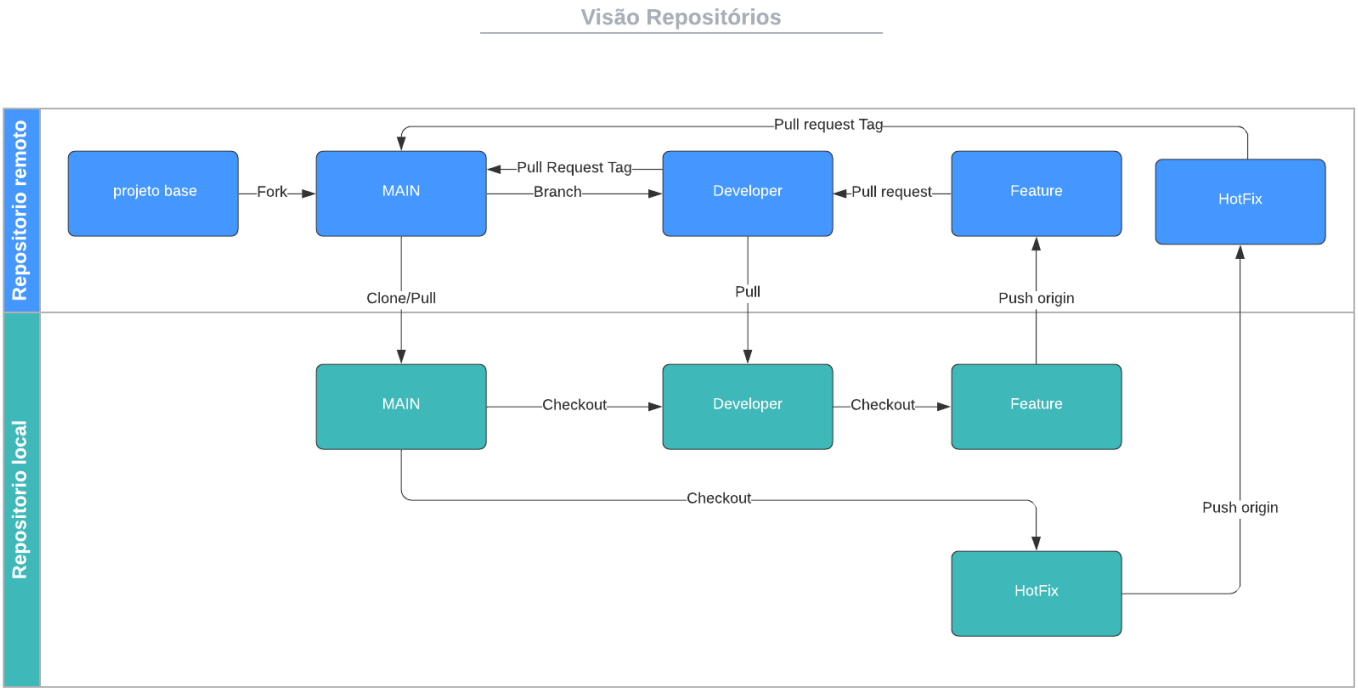
5. DEV

É a branch onde é feita o merge das features, também sendo a mais atualizada do projeto.

6. Pull request

É a solicitação para o merge para as branchs principais do projeto, geralmente estas estão bloqueadas para alteração direta.

Figura 1 - Processo de versionamento de código

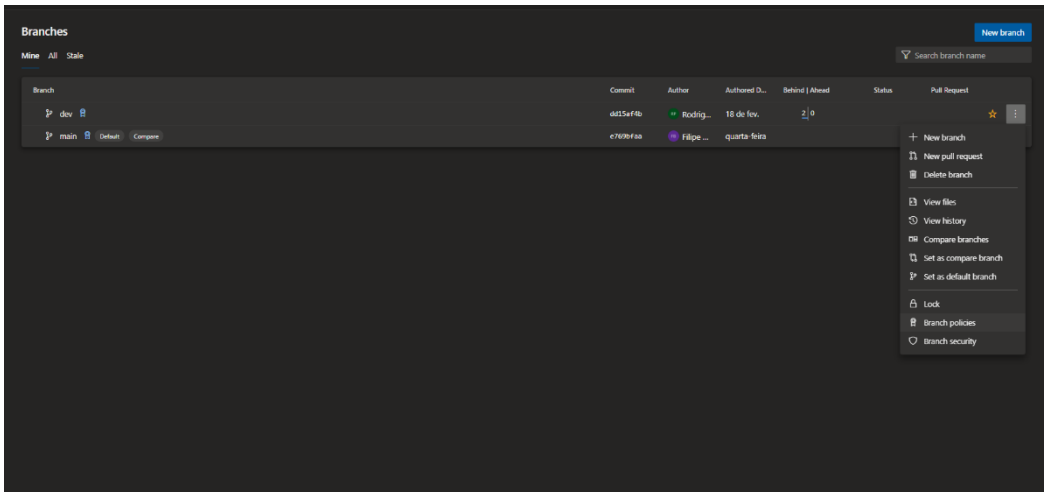


5. Descrição de Atividades

1. Iniciando o projeto

- Realizar um fork do projeto base. Caso tenha dúvidas, consultar a sessão de referências.
- Após o fork do projeto base ir em Repos > Branches > Branches polices
- Ativar a opção - Branch Policies Require a minimum number of reviewers – colocar um e marcar a opção when new changes are pushed Reset all code reviewer votes
- (Opcional) Ir em Automatically included reviewers e adicionar pelo menos um membro do projeto
- Criar a Branch Dev com base na Main e remarcar as mesmas opções em Branches polices

Figura 2 – Tela Branches



2. Trabalhando com Feature

- Antes de criar a **feature**, deve ser feita a atualização da **dev local** de acordo com a **origin**.
- Criar a **branch feature** a partir da **dev local** atualizada.
- O nome desta **branch** deve ser **feature**/<nome-da-feature>

1. Fazer **clone** do projeto remoto para a máquina local
2. Criar uma **branch** a partir da **dev**
3. Todo dia durante o desenvolvimento fazer pelo menos um **pull** da **dev** na **feature** para manter o código atualizado
4. Após finalizar a **feature** fazer um **merge** da **dev** na **feature**
5. Enviar a **feature** para o repositório remoto
6. Solicitar um **pull request** para a **dev**

3. Solicitando um Pull Requests

- **Pull request** é a ação de mesclar o código desenvolvido com as **branchs** principais

- Após subir a **branch feature** ou **hotfix** o desenvolvedor deve solicitar um **pull request**
- Os **pulls requests** das **features** nunca devem ocorrer na **main**, qualquer **pull request** para **main** deve ser rejeitado pelo avaliador
- Todos os **pull request** dos **hotfixs** devem na **main**
- Ao solicitar um Pull Request é de **suma importância** que tenha um título e uma descrição coerente com o que foi desenvolvido e está sendo solicitado para o merge.
    1. Entrar no Repositório do projeto ir em **pull request**
    2. Apertar o botão new **pull request**
    3. Escolher a **branch source** e a **branch de destino** (lembrando da regra de **hotfix** e **feature**)
    4. Escolher pelo menos um revisor do código que também será responsável pelo merge
    5. Ao Abrir o **pull request**, escrever o título que corresponda ao que foi desenvolvido
    6. Na descrição dar preferência para o **commit** da **branch** de **origem**, isso não impede que a descrição seja aprimorada pelo desenvolvedor
    7. Se houver **work item** selecioná-lo

4. Trabalhando com Hotfix

- **hotfix** sempre são criadas a partir da **main**
- Toda branch hotfix será excluída após seu término, isso é feito no momento que o aprovador realizar o merge, na aprovação a própria Azure DevOps disponibilizará um checkbox.
    1. Fazer **clone** do projeto remoto para a máquina local
    2. Criar uma **branch** a partir da **main**
    3. O nome desta **branch** deve ser **hotfix**/<nome-da-hotfix>
    4. Após a correção fazer uma atualização da **main (Pull)**
    5. Efetuar o **merge** na **branch main** com a **hotfix**
    6. Resolver os possíveis conflitos
    7. Fazer um **push origin** hotfix/<nome-da-hotfix> para o servidor remoto
    8. Solicitar um **pull request**
    9. Após aprovado, gerar uma **tag**
    10. Efetuar o **merge** na **branch origin** de **dev** com a atualização da **branch main**.

5. Aprovando um Pull Request

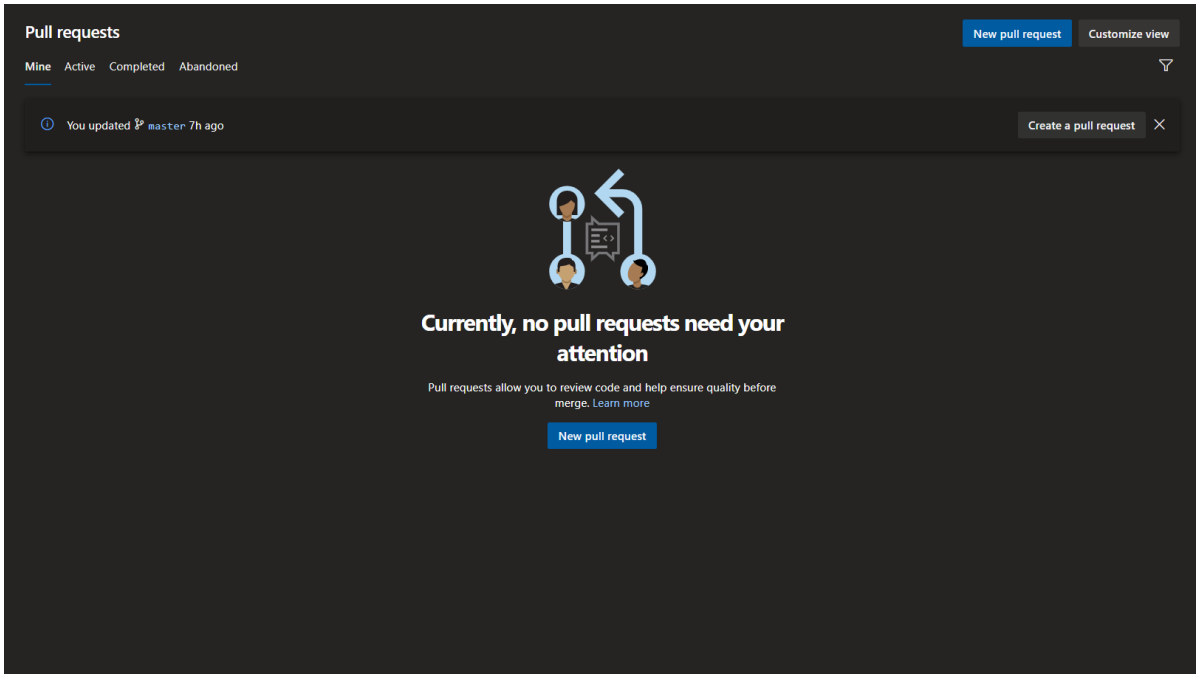
- O Aprovador do pull request não pode ser a mesma pessoa que solicitou
- Os aprovadores devem ser desenvolvedores do projeto
    1. Ao receber uma solicitação de análise com conflito recusar imediatamente
    2. Verificar a concordância do código, padrões de projetos e boas práticas
    3. O aprovador em caso de aprovação deve finalizar o merge.

6. Executando o Pull request da Dev para Main

- No final da sprint ou rodada de entrega o projeto que está na dev deve ser efetuado o merge para a **main**
- Somente neste caso será possível que a mesma pessoa que abriu o **pull request** aprove (mas é sempre recomendado que outra pessoa avalie o **pull request**)

- Ao fazer o **merge** a pessoa deve gerar uma **tag** conforme a entrega da sprint / versão
- **Pull request** da **dev** para a **main** deve ser feito por quem estiver mais inteirado do projeto

Figura 3 – Tela Pull requests



6. Utilização de Dados Pessoais (LGPD)

Não contempla a utilização de Dados Pessoais, conforme a Lei Geral de Proteção de Dados Pessoais (LGPD).

A utilização de dados pessoais é utilizada ao logar na ferramenta e na solicitação de aprovação do merge do código fonte.

Dados de Entrada	Fonte de Dados	Tipo de Tratamento do Dado	Dados de Saída
Nome	Active Directory	Comunicação	Azure DevOps
E-mail	Active Directory	Comunicação	Azure DevOps

O tratamento de dados é realizado exclusivamente para identificação dos envolvidos e envio de notificação para aprovação da solicitação de merge do código fonte.
Toda e qualquer informação a respeito de clientes e fornecedores (pessoa natural) somente serão repassadas mediante aprovação expressa destes ou por ordem judicial.

7. Anexos

N/A

8. Controle de Versão

Histórico (para uso da Gerência de Governança de TI):

Versão	Data Publicação	Elaborador	Motivo da Versão
1.0	19/03/2021	Mikael do Nascimento Medeiros	Padronizar e dar conhecimento sobre o uso do versionamento de código com a utilização da ferramenta GIT e Azure DevOps na STI.

Aprovação da Versão Atual:

Nome	Cargo

Anexo 5: Qualidade de Código

1. Qualidade de Código:

1.1. O nível da qualidade de código será avaliado por meio de métricas extraídas da ferramenta de avaliação de qualidade de código SonarQube 7.5 ou superior.

1.1.1. Algumas métricas, em função de características arquiteturais, poderão ser redefinidas individualmente, nas demandas de desenvolvimento ou manutenção evolutiva, a partir de dados históricos ou valores utilizados em projetos de características semelhantes.

1.1.2. As metas serão definidas no início do serviço de desenvolvimento e manutenção evolutiva.

1.2. O Índice de Qualidade do Software (IQS) será obtido a partir do parâmetro Maintainability Rating da ferramenta SonarQube, versão 7.5 ou superior.

1.3. O SESI pode dispensar uma ou mais métricas do critério de avaliação de qualidade de código justificadamente, devendo esta decisão ser registrada na demanda de desenvolvimento ou manutenção evolutiva. Compete ao SESI elencar quais demandas estarão sujeitas a esta avaliação.

1.4. Será responsabilidade do CIS a realização de entregas parciais e finais que estejam com IQS de acordo com a régua de qualidade definida (que são as metas a serem definidas de acordo com o item ‘1.1.2’), e, no caso de solução pré-existente, que estas mesmas entregas não depreciem o IQS já apurado para a solução.