Guia Yuri Garcia Pardinho - Método Amarelo

"Simplicidade segura, código claro, impacto real."

1. Objetivo

Este guia estabelece um padrão profissional mínimo para repositórios, propostas técnicas, produtos digitais e onboarding de times,
com foco em simplicidade, clareza, segurança, acessibilidade e pragmatismo.

Serve para:
- Repositórios
- Documentação técnica
- Projetos web, mobile e APIs
- Onboarding e alinhamento com times internos e parceiros

Nem todo item será aplicável a todo projeto.
Cada equipe deve adaptar o guia ao contexto, ao risco, ao tamanho do sistema e às restrições técnicas e regulatórias, documentando exceções.
Em caso de conflito entre este guia e lei, contrato ou norma oficial, prevalece o mais restritivo.

2. Princípios Fundamentais de Código

2.1 Simplicidade e Foco
- KISS (Keep It Simple, Stupid): evitar complexidade desnecessária.
- YAGNI (You Ain't Gonna Need It): não implementar hoje o que não tem demanda clara.
- DRY (Don't Repeat Yourself): evitar duplicar regras de negócio, principalmente em cálculos financeiros, regras fiscais e contratuais.
  Se houver duplicação por motivos técnicos (performance, isolamento, legados), documentar a decisão.

2.2 Organização e Responsabilidades
- Separation of Concerns (SoC): separar UI, aplicação, domínio e infraestrutura.
- Single Level of Abstraction Principle (SLAP): funções com um único nível de abstração.
- Principle of Least Surprise (PoLS): código, APIs e telas funcionam de forma previsível.
- GRASP: usar como referência mental para atribuição de responsabilidades.

2.3 SOLID (Uso Pragmático)
Aplicar quando tornar o sistema mais claro e testável, nunca por formalismo:
- SRP: cada módulo com uma razão clara de mudança.
- OCP: facilitar extensão sem reescrever o núcleo estável.
- LSP: substituições não quebram expectativas.
- ISP: interfaces pequenas e específicas.
- DIP: depender de abstrações para facilitar testes e evolução.

2.4 Inversão de Controle e Flexibilidade
- Uso de DI/IoC recomendado para integrações externas, testes desacoplados e sistemas em crescimento.
- Em projetos simples, pode ser simplificado, desde que não prejudique testes e segurança.

2.5 Arquitetura por Domínio
- DDD para domínios complexos.
- CQRS apenas quando houver benefício real em escala, relatórios ou leitura complexa.
- Para CRUD simples ou MVPs, manter arquitetura enxuta.

2.6 Gerenciamento de Recursos
- Garantir liberação de recursos (conexões, arquivos, locks), usando RAII ou

equivalentes, conforme a linguagem.

2.7 Transações e Consistência
- Usar ACID para dinheiro, obrigações legais e dados críticos.
- Se adotar consistência eventual, documentar riscos, compensações e estratégias de compensação.

2.8 Security & Privacy by Design
- Segurança incorporada desde o desenho da solução.
- Privacidade por padrão (Privacy by Default).
- Defesa em profundidade (Defense in Depth).

2.9 Segurança como Requisito Não Funcional
- Requisitos de segurança, privacidade e acessibilidade descritos no backlog e na Definition of Done.

3. Cybersecurity Web/Mobile

3.1 Fundamentos
- Autenticação forte, com 2FA em funções e dados sensíveis.
- Princípio do menor privilégio e segregação de funções.
- Validação de entrada em todas as camadas e sanitização/encode de saída.
- Monitoramento, logs seguros e plano de resposta a incidentes documentado.

3.2 Segurança Web
- Seguir OWASP Top 10 e OWASP API Security Top 10.
- Prevenir injection usando ORM ou queries parametrizadas.
- Prevenir XSS via output encoding e sanitização de conteúdo dinâmico.
- Proteger contra CSRF em rotas com estado.
- Utilizar cabeçalhos de segurança, como:
  - Content-Security-Policy sem 'unsafe-inline', usando nonces ou hashes.
  - X-Frame-Options: DENY
  - X-Content-Type-Options: nosniff
  - Referrer-Policy: no-referrer (ou equivalente seguro)
  - Strict-Transport-Security com max-age adequado
  - Permissions-Policy restritiva para recursos sensíveis.

3.3 Segurança Mobile
- Certificate pinning em APIs críticas.
- Mecanismos de anti-tampering e anti-debug quando adequado.
- Uso de armazenamento seguro (KeyStore/Keychain).
- Autenticação biométrica nativa para ações sensíveis.
- Detecção de root/jailbreak quando relevante ao risco.

3.4 Proteção de Dados
- Criptografia em trânsito com TLS 1.3 (aceitar 1.2 apenas com suites fortes, se indispensável).
- Criptografia em repouso com algoritmos seguros (ex: AES-256-GCM).
- Minimização de coleta de dados e prazos de retenção definidos.
- Uso de anonimização/pseudonimização para métricas e analytics.
- Separação de ambientes (dev, homologação, produção) e dados mascarados em ambientes não produtivos.

3.5 Autenticação e Autorização
- Tokens de acesso com expiração curta; refresh tokens com revogação segura.
- RBAC ou ABAC bem definidos, com granularidade adequada.
- Rate limiting e proteção contra brute force.
- Step-up authentication para ações de alto risco.
- Sessões protegidas contra fixation, hijacking e reutilização.

4. Responsividade Multi-dispositivo

4.1 Mobile-First
- Projetar primeiro para telas pequenas, expandindo para tablets e desktops.
- Garantir legibilidade, espaçamento adequado e hierarquia visual clara.

4.2 Breakpoints Semânticos
- Definir breakpoints por contexto de uso (mobile, tablet, desktop, widescreen), não por modelos específicos de dispositivo.

4.3 Layouts Adaptativos
- Uso de grids fluidos e componentes flexíveis.
- Imagens responsivas com srcset/picture e textos alternativos adequados.

4.4 Touch e Mouse
- Tamanhos mínimos de alvo (>= 44px).
- Estados visuais claros (hover, focus, active, disabled).
- Ajustes distintos para dispositivos com pointer fino (mouse) e coarse (touch).

5. Acessibilidade

5.1 Diretrizes Gerais
- Alinhamento a WCAG 2.2 Nível AA.
- Foco visível e não encoberto.
- Navegação completa por teclado.
- Suporte a leitores de tela.

5.2 ARIA e Semântica
- Uso correto de roles, landmarks e atributos ARIA.
- Anúncio de erros, alertas e mudanças de estado para leitores de tela.

5.3 Teclado e Leitores de Tela
- Gerenciamento adequado de foco em modais, menus, diálogos e componentes dinâmicos.
- Evitar armadilhas de teclado.

5.4 Contraste e Preferências do Usuário
- Contraste mínimo 4.5:1 para textos comuns e 3:1 para textos grandes.
- Respeitar prefers-reduced-motion, oferecendo experiências sem animações excessivas.

6. UX, UX Writing e Teoria das Cores

6.1 UX Writing
- Clareza, concisão e voz ativa.
- Evitar jargões técnicos para usuários finais.
- Mensagens de erro específicas, com orientação sobre o que fazer.

6.2 Psicologia das Cores
- Cores alinhadas a significados claros (sucesso, aviso, erro, informação, neutro).
- Uso consistente da paleta sem comprometer acessibilidade.
- Atenção a contextos culturais brasileiros sem reforçar estereótipos.

6.3 Paletas Acessíveis e Design Tokens
- Definir paletas com contraste adequado.
- Centralizar cores, tipografia, espaçamentos e sombras em design tokens.
- Usar componentes compartilhados alimentados por tokens para garantir consistência.

6.4 Design System e Microinterações
- Componentes documentados (botões, inputs, alerts, modais, etc.).
- Estados para hover, focus, disabled, loading e erro.
- Microinterações sutis, significativas e compatíveis com prefers-reduced-motion.

7. Qualidade, Testes e Entrega Contínua

7.1 Testes
- Testes unitários para regras de negócio.
- Testes de integração para integrações críticas.

- Testes end-to-end para fluxos-chave (login, cadastro, pagamento, etc.).

7.2 CI/CD
- Pipeline com lint, testes, scan de segurança e deployment rastreável.
- Para sistemas críticos, estágios obrigatórios de segurança e acessibilidade.
- Para protótipos, pipeline enxuto, com plano de amadurecimento progressivo.

7.3 Segurança no Pipeline
- Uso de ferramentas de análise estática, SCA (análise de dependências), verificação de segredos e políticas de branch protegidas.
- Bloqueio para CVEs críticos não tratados.

8. Conformidade Legal e Normas Brasileiras

8.1 LGPD
- Mapear dados pessoais e sensíveis.
- Definir base legal para cada tratamento.
- Garantir direitos dos titulares (acesso, correção, portabilidade, exclusão, revogação de consentimento).
- Implementar controles de segurança compatíveis com o risco.
- Manter registro das operações de tratamento.
- Estabelecer fluxo de comunicação de incidentes de segurança com prazos, responsabilidades e registro.

8.2 Marco Civil da Internet
- Manter termos de uso e política de privacidade claros.
- Guardar registros conforme prazos legais aplicáveis.
- Proteger registros com sigilo, integridade e controle de acesso.

8.3 Acessibilidade e Lei Brasileira de Inclusão
- Acessibilidade tratada como obrigação técnica e ética.
- Produtos digitais com navegação, leitura e interação acessíveis.

8.4 Segurança da Informação
- Gestão de segredos (não armazenar chaves em código).
- Criptografia adequada.
- Plano de resposta a incidentes.
- Políticas de backup, restauração e continuidade de negócio.

9. Comentários, Idioma e Documentação

9.1 Comentários de Código
- Comentários explicam o porquê, não o óbvio.
- Evitar dados reais e piadas inadequadas.
- Manter comentários atualizados; remover o que não for válido.

9.2 Idioma de código e de conteúdo
- Termos de domínio de negócio podem ser em português para aderência ao contexto.
- Termos técnicos consolidados podem permanecer em inglês.
- Manter consistência dentro do projeto.

9.3 Documentação
- README com visão geral, setup e formas de execução.
- Visão arquitetural de alto nível.
- Documentação de APIs (OpenAPI/Swagger ou equivalente).
- Guia de contribuição quando houver colaboração.

10. Checklists

10.1 Início de Projeto
- Stack definida e justificada.
- Ambiente de desenvolvimento padronizado.
- Estratégia de logs, métricas e monitoramento.
- CI/CD configurado proporcional ao risco.

- Mapeamento de dados pessoais e requisitos legais.
- Diretrizes de acessibilidade e segurança definidas.

10.2 Pull Request
- Código legível, coeso e aderente aos princípios.
- Testes atualizados e passando.
- Lint e ferramentas automáticas sem erros relevantes.
- Impactos em segurança, performance e acessibilidade considerados.
- Documentação ajustada quando o comportamento muda.

10.3 Pré-Produção
- Testes end-to-end nos fluxos críticos.
- Testes de carga para cenários relevantes.
- Plano de rollback testado.
- Checklist de LGPD, segurança e acessibilidade validado.
- Termos de uso, política de privacidade e avisos legais revisados.

10.4 Segurança Web/Mobile
- 2FA onde aplicável.
- Rate limiting em endpoints sensíveis.
- TLS configurado corretamente.
- CSP aplicada.
- Dados sensíveis criptografados.
- Rotação de chaves e segredos.
- Backups criptografados e testados.

10.5 Responsividade e Acessibilidade
- Testes práticos em diferentes tamanhos de tela.
- Navegação por teclado completa.
- Contraste validado.
- Comportamento adequado em prefers-reduced-motion.
- Textos alternativos em mídias relevantes.

11. Instruções para Fluxo de Trabalho

11.1 Commits e Rastreabilidade
- Utilizar padrão consistente (ex.: Conventional Commits).
- Associar PRs a issues quando fizer sentido.
- Manter histórico compreensível.

11.2 Fluxo de Git
- Evitar push direto em branches de produção.
- Adotar estratégia simples e clara (ex.: trunk-based, GitHub Flow).

11.3 Documentação Automatizada
- Automatizar geração e publicação de documentação quando possível.
- Atualizar documentação junto com mudanças de comportamento.

12. Glossário

ACID, BDD, CQRS, DDD, DRY, GRASP, KISS, LGPD, PoLS, RAII, SLAP, SoC, SOLID, TDD, WCAG,
CSP, HSTS, 2FA, RBAC, ABAC, MASVS, SBOM, Design Token, Microinteraction, ARIA, Screen Reader,
Focus Management, Mobile-First, Touch Target.

13. Registro de Decisões Técnicas (ADR)

Usar ADRs para registrar escolhas arquiteturais, de segurança, UX e infraestrutura.

Template sugerido:

Título da decisão
Status (Proposto | Aceito | Obsoleto)

Contexto (problema, restrições, riscos, impacto em UX e segurança)
Decisão (o que será feito, como e por quê)
Consequências (benefícios, riscos, mitigações, impacto em performance e experiência)
Testes de validação (pentest, acessibilidade, responsividade, usabilidade, carga)
Links relacionados (issues, PRs, docs)