PLAYBOOK PROJETO GRUPO 1
10º Módulo

inteli
instituto
de tecnologia
e liderança

1. INTRODUÇÃO PLAYBOOK

Este **Playbook** é um guia técnico-estratégico criado para consolidar o conhecimento gerado durante o desenvolvimento da **esteira de CI/CD da plataforma Kino**, iniciativa conduzida em parceria com a **TimeNow**. Ele reúne de forma estruturada os fluxos, decisões, boas práticas, padrões técnicos, artefatos e aprendizados produzidos ao longo do projeto, com o objetivo de servir como uma ferramenta de **documentação viva, reprodutível e aplicável** a novos contextos dentro da organização. Com foco na automação, segurança e eficiência do ciclo de vida do software, a esteira desenvolvida para o Kino foi projetada para validar código de forma automatizada, garantir a segurança dos ambientes, executar testes contínuos e padronizar entregas com rastreabilidade e monitoramento em tempo real.

Neste contexto, o Playbook opera como uma **ferramenta integradora**, permitindo que qualquer equipe técnica da TimeNow — atual ou futura — compreenda não apenas o funcionamento da esteira, mas também as decisões que a sustentam. Este material é dividido em seções que abordam desde a formação da equipe, o mapeamento de papéis e responsabilidades, até o detalhamento do processo de design, arquitetura técnica, estratégias de segurança, observabilidade e testes. Cada capítulo oferece uma visão conectada entre **intenção de projeto, práticas aplicadas e valor gerado**. O Playbook tem como principais objetivos:

- **Registrar e compartilhar o conhecimento institucional** construído ao longo das sprints;
- **Padronizar fluxos e decisões técnicas** com base em boas práticas DevOps e DesignOps;
- **Orientar squads futuros** na manutenção, expansão ou adaptação da esteira de CI/CD;
- **Alinhar expectativas entre áreas técnicas e de negócio**, facilitando a comunicação entre desenvolvedores, analistas, liderança técnica e stakeholders estratégicos;
- **Promover a reprodutibilidade e escalabilidade** do modelo de esteira, permitindo que os aprendizados do projeto Kino sirvam como base para outros produtos ou times da TimeNow;
- **Refletir a cultura de melhoria contínua**, expondo erros corrigidos, hipóteses testadas, métricas avaliadas e frameworks adotados (como DORA, Accelerate e REACH).

Ao final, este Playbook representa não apenas uma entrega documental, mas um **instrumento de governança técnica e estratégica**, que legitima o esforço colaborativo realizado pelos estudantes do Inteli e profissionais da TimeNow ao longo de todas as fases do projeto. Ele consolida aprendizados práticos e decisões conceituais que atravessam o ciclo de desenvolvimento moderno, promovendo uma base confiável para a continuidade, evolução e reuso da esteira CI/CD implementada. Sua estrutura foi pensada para não apenas documentar, mas também guiar futuras adaptações, encurtar ramp-ups técnicos e garantir que o conhecimento gerado se converta em valor estratégico para a organização.

2. EQUIPE

Nomes	Funções	Responsabilidades Técnicas e Estratégicas
Davi Arantes	**Desenvolvedor Backend / DevOps**	Implementação e manutenção da esteira CI/CD com foco em scripts YAML, integração com ferramentas de segurança (OWASP ZAP) e provisionamento de ambientes com boas práticas DevSecOps.

Eduarda Cardoso	**Analista de Produto / Métricas**	Responsável pela definição e acompanhamento de métricas técnicas e de negócio; criação de dashboards e painéis analíticos com foco em impacto e valor gerado pelas entregas.
Gabrielly Vitor	**DesignOps / Product Designer**	Estruturação do DesignOps, facilitação de sprints, documentação estratégica e governança de processos; responsável por construir e consolidar o Design Playbook do projeto.
João Sotto	**Segurança da Informação**	Implementação de práticas de segurança em pipelines (ex: Git Guardian, análise de secrets); apoio na criação de políticas de compliance e prevenção de falhas por exposição de dados.
Luigi Macedo	**QA Engineer / Research Ops**	Elaboração de estratégias de testes automatizados, coleta e análise de métricas de desempenho; suporte à avaliação técnica de hipóteses e validação de experimentos (REACH).
Paula Piva	**Scrum/ Desenvolvedora CI/CD**	Implementação das estratégias de deploy (CD), versionamento, rollback e controle de ambientes; responsável pela automação de entregas com testes e monitoramento em produção.

3. DESIGN SYSTEM

O Design System deste projeto foi concebido como um ecossistema completo de padronizações que articula elementos visuais, operacionais, técnicos e culturais do ciclo de desenvolvimento de software. Diferentemente de um guia puramente visual, ele incorpora práticas modernas de engenharia, princípios de DevOps e diretrizes de gestão ágil, constituindo-se como uma base estruturante para a entrega contínua e confiável de valor. Desenvolvido no contexto da colaboração entre o Inteli e a Timenow, o Design System atua como um contrato técnico e operacional entre os diferentes membros da equipe — designers, desenvolvedores, analistas de qualidade e engenheiros de infraestrutura — unificando as decisões quanto ao uso de ferramentas, fluxos de trabalho, nomenclaturas e critérios de qualidade.

Em projetos com alto grau de integração e múltiplos pontos de entrega, como os que envolvem pipelines de CI/CD, a ausência de padronização pode gerar retrabalho, inconsistências e perda de produtividade. Este documento busca justamente evitar esses riscos, consolidando boas práticas e promovendo transparência, previsibilidade e escalabilidade. Além disso, sua função educativa é relevante: ele facilita a integração de novos membros à equipe, serve como repositório vivo de decisões técnicas e ajuda a cultivar uma cultura de excelência e melhoria contínua.

3.1 Versionamento com Git e GitHub

- **Branching model**: main, develop, feature/*, hotfix/*, release/*
- **Commits**: Conventional Commits (`tipo: descrição`, ex: `feat: integração com Stripe`)
- **Ferramentas**: Git + GitHub, GitHub Actions
- **Benefícios**: Histórico claro, changelogs automatizados, revisão eficiente

3.2 Hooks de Qualidade (Husky + Lint-Staged)

- **Checks**: Lint, testes unitários
- **Ferramentas**: Husky, lint-staged
- **Benefícios**: Prevenção de erros, padronização automática, responsabilidade individual

3.3 Pipeline CI/CD (Azure DevOps)

- **Etapas**: dependências → lint → testes → build → deploy
- **Ferramentas**: Azure Pipelines, Azure Artifacts
- **Benefícios**: Redução de erros manuais, rastreabilidade, conformidade

3.4 Boards e Agile (Azure Boards)

- **Work Items**: Feature, Task, Bug, Test Plan
- **Sprints**: Iterações quinzenais com colunas New → Active → Blocked → Resolved → Closed
- **Benefícios**: Visibilidade e alinhamento entre áreas

3.5 Estimativas e Story Points

- **Método**: Planning, esforço relativo
- **Benefícios**: Previsão realista, priorização eficiente, base para métricas futuras

3.6 Testes (Manuais e Automatizados)

- **Abordagem**: Azure Test Plans
- **Benefícios**: Redução de regressões, validação contínua, rastreabilidade de bugs

3.7 Segurança e Proteção de Dados

- **Práticas**: GitGuardian, variáveis no Azure
- **Benefícios**: Conformidade LGPD, prevenção de vazamento de segredos

4. PROCESSO DE DESIGN (PLAYS)

O processo de design deste projeto foi conduzido com base em uma abordagem iterativa e colaborativa, inspirada em modelos ágeis adaptados às demandas técnicas de esteiras CI/CD. Cada etapa — do entendimento inicial ao aprendizado contínuo — foi pensada para alinhar as decisões de design com os objetivos estratégicos da TimeNow, incorporando práticas de co-criação, validação técnica e experimentação controlada.

A estrutura adotada segue a lógica dos *Design Plays*, proposta por Arturo Leal, que organiza o trabalho em quatro macrofases: **Entender**, **Descobrir**, **Design** e **Testar e Aprender**. Essa divisão permitiu à equipe balancear profundidade analítica com agilidade de execução, mantendo a rastreabilidade das decisões ao longo de todas as sprints. A seguir, descrevemos como cada uma dessas etapas foi aplicada no contexto do desenvolvimento da esteira de CI/CD da plataforma Kino.

a. Entender
A etapa inicial foi marcada por alinhamentos estratégicos com representantes técnicos da TimeNow, com o objetivo de compreender profundamente as necessidades da organização em relação à esteira de CI/CD da plataforma Kino. Para isso, foram realizadas entrevistas estruturadas com stakeholders-chave — incluindo engenheiros DevOps, profissionais de QA e tech leads — além de observações de rotinas operacionais e análise de fluxos existentes. Essa abordagem permitiu identificar não apenas requisitos explícitos, mas também lacunas operacionais, expectativas ocultas e indicadores de sucesso relevantes.

b. Descobrir
Na fase de descoberta, a equipe conduziu um benchmarking técnico com base em esteiras utilizadas por empresas referência no setor (como GitHub Actions, GitLab CI/CD e Jenkins X). A partir dessa análise comparativa, foram levantadas hipóteses sobre os principais gargalos enfrentados — como baixa rastreabilidade, deploys manuais frágeis e ausência de padronização entre squads. Essas hipóteses foram testadas por meio de simulações, protótipos de fluxo e walkthroughs técnicos, viabilizando validações ágeis e orientadas a contexto real.

c. Design
Com as hipóteses validadas, iniciou-se a modelagem da solução. Foram desenvolvidos protótipos funcionais de pipelines contemplando etapas essenciais como build, testes automatizados, deploy canário, rollback e integração com ferramentas de segurança (ex.: OWASP ZAP). A esteira foi estruturada em fases lógicas para facilitar a manutenção e o onboarding de novos membros, e os scripts YAML foram modularizados com templates reutilizáveis e convenções de nomenclatura padronizadas. O design priorizou a escalabilidade, rastreabilidade e compatibilidade com as ferramentas utilizadas pela TimeNow, como Azure DevOps e GitHub.

d. Testar e Aprender
A cada iteração, o pipeline foi validado com stakeholders por meio de walkthroughs técnicos, análises de logs e simulações controladas de falhas. Foram conduzidos testes de recuperação (Disaster Recovery), análise dinâmica de segurança e revisões por pares em Pull Requests integradas ao fluxo de CI. Os aprendizados obtidos foram registrados em retrospectivas e documentados no Playbook, permitindo ajustes iterativos na arquitetura dos pipelines, melhorias nos tempos de execução e aperfeiçoamento dos critérios de aprovação e rollback. Essa abordagem cíclica fomentou uma cultura de experimentação, aprendizado contínuo e melhoria progressiva da solução.

5. TESTES DE VALIDAÇÃO

A validação contínua foi um dos pilares da construção da esteira de CI/CD da plataforma Kino. Desde os primeiros commits até os deploys em ambiente de homologação, foram definidos mecanismos de teste

automatizados e controles manuais que assegurassem a integridade, a segurança e a rastreabilidade das entregas. A estratégia adotada visou garantir qualidade desde as fases iniciais de integração até a disponibilização da aplicação, permitindo também respostas rápidas a falhas e facilitando a manutenção do sistema em produção. Esta seção detalha os métodos de teste utilizados, os critérios de sucesso definidos e os aprendizados que impulsionaram melhorias ao longo do projeto.

a. Estratégia de Testes Adotada
A abordagem de testes foi estruturada em camadas, abrangendo testes unitários, de integração e end-to-end (E2E), com execução automatizada integrada à pipeline de CI/CD.
Na fase de integração contínua (CI), testes unitários foram utilizados para validar comportamentos isolados das funções implementadas, juntamente com linters para garantir aderência a padrões de estilo e formatação.

Na fase de entrega contínua (CD), a esteira executava testes de integração e testes end-to-end, reproduzindo interações reais do usuário e assegurando o correto funcionamento do sistema de forma sistêmica. Além disso, foram realizados testes de recuperação (Disaster Recovery) e validações de segurança com ferramentas como o OWASP ZAP, que permitiram análises dinâmicas em ambientes simulados.

b. Métricas e Critérios de Sucesso
Para medir o desempenho técnico da esteira e validar suas entregas, foram definidos os seguintes indicadores-chave de sucesso (KPIs):

- Pass Rate da pipeline igual ou superior a 95%
- Tempo médio de build inferior a 10 minutos (Percentil 95 ≤ 15 minutos)
- Cobertura de testes automatizados superior a 80% nos módulos críticos
- Tempo máximo de rollback em produção de até 5 minutos
- Taxa de falha em deploys (Change Failure Rate) inferior a 5%

Essas métricas foram monitoradas em dashboards customizados via Azure DevOps e Grafana, garantindo visibilidade contínua e suporte à tomada de decisão técnica.

c. Aprendizados e Ajustes
O processo de testes gerou uma série de aprendizados que orientaram ajustes estratégicos na arquitetura da esteira e nos procedimentos adotados:

- A etapa de build foi otimizada com a paralelização de processos e reorganização de tarefas de alta carga, como o build de containers, resultando em menor tempo de execução da pipeline.
- As regras de aprovação foram refinadas para permitir exceções justificadas, baseadas em logs, métricas ou evidências técnicas documentadas em Pull Requests.
- A camada de observabilidade foi fortalecida com a criação de dashboards operacionais em Grafana e alertas automatizados, permitindo respostas rápidas a falhas.
- A integração contínua de ferramentas de segurança possibilitou a identificação precoce de vulnerabilidades, promovendo maior conformidade com boas práticas de proteção de dados.

Essas melhorias demonstram o impacto direto da validação contínua na confiabilidade e eficiência da solução entregue, além de reforçarem a adoção de uma cultura DevOps baseada em dados, testes e aprendizado iterativo.

5. BOAS PRÁTICAS

Ao longo do desenvolvimento da esteira de CI/CD da plataforma Kino, a equipe identificou desafios recorrentes relacionados à padronização, segurança, rastreabilidade e qualidade das entregas. Esta seção documenta as práticas consolidadas que surgiram como resposta a esses desafios, destacando os ganhos técnicos e organizacionais gerados. O objetivo é fornecer um referencial prático e replicável que contribua para a escalabilidade, previsibilidade e melhoria contínua dos processos de desenvolvimento na TimeNow.

Situação Identificada	Prática Consolidada	Impacto Técnico e Organizacional
Complexidade na jornada de validação	Segmentação da esteira em fases lógicas: Build, Testes, Deploy e Observability	Redução da complexidade cognitiva; melhoria da manutenção contínua; maior onboarding de novos integrantes
Merge incorreto diretamente na branch principal	Proteção de branches com políticas de merge + obrigatoriedade de revisões via Pull Request integradas à CI	Aumento da confiabilidade em produção; padronização do fluxo de aprovação de código; mitigação de falhas críticas
Incidência de vulnerabilidades em bibliotecas	Integração de ferramentas como Snyk, Dependabot e OWASP ZAP para análise contínua de vulnerabilidades	Detecção antecipada de riscos de segurança; conformidade com a LGPD e normas internas de segurança
Dificuldade de rastrear versões em múltiplos ambientes	Versionamento semântico automatizado com identificadores únicos (buildId + hash SHA)	Rastreabilidade granular por build; facilidade de rollback; alinhamento entre release notes e deploy
Falta de padronização entre squads	Uso de Husky + Lint-Staged para enforcement de formatação, convenção de commits e execução de testes locais	Garantia de consistência entre branches; incremento de qualidade desde o push; redução de retrabalho
Visibilidade limitada sobre status de entrega	Implantação de dashboards customizados (Azure DevOps, Prometheus, Grafana) com indicadores-chave	Transparência operacional; antecipação de falhas; cultura organizacional orientada a dados e métricas contínuas
Inconsistência de ambientes	Implementação de infraestrutura como código com Terraform e ambientes efêmeros	Redução de variabilidade entre execuções; ambientes reprodutíveis e isolados; maior segurança operacional

6. GLOSSÁRIO

Durante o desenvolvimento da esteira de CI/CD, foram utilizados diversos conceitos, ferramentas e práticas específicas das áreas de DevOps, engenharia de software, segurança e metodologias ágeis. A fim de promover clareza, alinhamento conceitual e facilitar o onboarding de novos integrantes, este glossário reúne as principais definições técnicas adotadas no projeto. Os termos aqui apresentados refletem tanto a base teórica utilizada (como métricas DORA, versionamento semântico e testes), quanto às ferramentas práticas integradas à solução (como Azure DevOps, GitHub Guardian, OWASP ZAP, entre outras).

Ferramenta/Tecnologia	Descrição
Azure DevOps	Plataforma da Microsoft que integra ferramentas de versionamento, integração contínua, entrega contínua e gerenciamento ágil de projetos. Foi a principal ferramenta utilizada na esteira do projeto.
Branch	Ramificação de código em um repositório Git. Permite desenvolvimento paralelo sem afetar a base principal do projeto (ex: main, develop, feature/*, hotfix/*).
Build	Processo de compilação ou empacotamento de código-fonte para gerar um artefato executável. Pode incluir etapas de dependências, testes e validações.
Canary Release	Estratégia de deploy que libera a nova versão para uma pequena parcela dos usuários antes de uma liberação ampla, minimizando riscos.
CI/CD (Continuous Integration / Continuous Delivery)	Práticas DevOps que automatizam a integração, teste e entrega contínua de código para ambientes de produção com rapidez e segurança.
Change Failure Rate	Métrica que mede a proporção de deploys em produção que resultam em falhas. Parte das métricas DORA
DesignOps	Abordagem operacional para escalar práticas de design, promovendo padronização, governança e integração entre áreas de design e tecnologia.
Disaster Recovery (DR Test)	Testes que simulam falhas críticas (como perda de infraestrutura ou falhas de serviço) para avaliar a resiliência da aplicação.
DORA Metrics	Conjunto de quatro métricas (Deployment Frequency, Lead Time for Changes, Mean Time to Restore, Change Failure Rate) amplamente utilizado para medir performance em DevOps.
End-to-End (E2E)	Tipo de teste que simula o comportamento completo do usuário, validando a interação entre múltiplos componentes do sistema.
Git	Sistema de controle de versão distribuído utilizado para rastrear alterações no código e coordenar trabalho entre desenvolvedores.
GirGuardian	Ferramenta de segurança que monitora repositórios de código (públicos e privados) em busca de segredos expostos, como chaves de API, tokens, senhas e credenciais sensíveis. Pode ser integrada ao pipeline de CI/CD para realizar varreduras automáticas e prevenir vazamentos de dados antes do deploy, promovendo conformidade com boas práticas de segurança e a LGPD.
Husky	Ferramenta que permite configurar hooks Git, como execução de scripts antes de commits ou pushs, garantindo controle de qualidade desde o

	início do desenvolvimento.
Lint / Linter	Ferramenta que analisa o código para encontrar erros de sintaxe, formatação ou violação de padrões de estilo.
Observabilidade	Capacidade de monitorar e compreender o comportamento interno de um sistema com base em seus outputs (logs, métricas, rastreamento).
OWASP ZAP	Ferramenta de análise dinâmica de segurança (DAST) utilizada para identificar vulnerabilidades durante a execução de uma aplicação.
Pipeline	Fluxo automatizado de etapas técnicas (ex.: build, test, deploy) definidas para garantir entregas contínuas e seguras de software.
Rollback	Processo de reverter uma versão para o estado anterior após uma falha ou erro em produção.
Semantic Versioning (SemVer)	Sistema de versionamento baseado em três números: MAJOR.MINOR.PATCH (ex: 2.1.3), usado para comunicar mudanças e compatibilidade.
Story Points	Unidade relativa de medida usada em metodologias ágeis para estimar o esforço ou complexidade de uma tarefa, sem depender de tempo absoluto.
Test Plan / Test Suite	Conjunto estruturado de casos de teste, frequentemente utilizado em ferramentas como o Azure Test Plans para rastrear a execução e a cobertura de testes.
Work Item	Item de trabalho que representa uma unidade de rastreamento dentro do Azure Boards, como uma funcionalidade, tarefa, bug ou teste. Os *Work Items* são usados para planejar, acompanhar e gerenciar o progresso de atividades em metodologias ágeis, permitindo vinculação com commits, pull requests e builds. Estruturas comuns incluem tipos como: `Feature`, `User Story`, `Task`, `Bug` e `Test Case`.

7. LINHA DO TEMPO (SPRINT 1 A 5)

- **Sprint 1:** Definição de visão de produto e métricas
- **Sprint 2:** Criação de personas, blueprints e primeiros testes CI
- **Sprint 3:** Implantação de etapas CD e segurança
- **Sprint 4:** Medição de impacto, integração de segurança e deploy canário
- **Sprint 5:** Documentação, report e refinamento do Playbook

8. REFERÊNCIAS

- FORSYTH, Ian. DORA Metrics for DevOps. Google Cloud. Disponível em: https://cloud.google.com/devops. Acesso em: 28 abr. 2025.

- HUMBLE, Jez; FORSGREN, Nicole; KIM, Gene. *Accelerate: The Science of Lean Software and DevOps*. IT Revolution Press, 2018.
- SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. Pearson, 2019.
- SCHWABER, Ken; SUTHERLAND, Jeff. *The Scrum Guide*. Scrum.org, 2020.
- PRUITT, John; ADLIN, Tamara. *The Persona Lifecycle*. Elsevier/Morgan Kaufmann, 2006.
- COHN, Mike. *User Stories Applied: For Agile Software Development*. Addison-Wesley, 2004.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO/IEC 25010:2011 – SQuaRE: System and software quality models*. https://iso.org/standard/35733.html. Acesso em: 07 maio 2025.
- MIRO. “Customer Journey Map Template.” https://miro.com/templates/customer-journey-map/
- IBM. *Design Systems: How to Build and Use Them Effectively*. IBM Developer, 2019. https://developer.ibm.com/articles/design-systems-article/. Acesso em: 1 jun. 2025.
- CONVENTIONAL COMMITS. *Conventional Commits Specification*. https://www.conventionalcommits.org/. Acesso em: 1 jun. 2025.
- MICROSOFT. *Azure DevOps Services Documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/. Acesso em: 1 jun. 2025.

Observação: Outras informações, como User Stories, Tecnologias utilizadas e complementos sobre código, é necessário ler a documentação completa no repositório do projeto, disponível no GitHub.