UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE CRATEÚS

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

LUCAS LIMA MOTA

DO CONCEITO À INTERPRETAÇÃO DA PRÁTICA: EXPLORANDO O USO DE ROLE-PLAY PARA ENSINAR OS FUNDAMENTOS DE DEVOPS

CRATEÚS

2024

LUCAS LIMA MOTA

DO CONCEITO À INTERPRETAÇÃO DA PRÁTICA: EXPLORANDO O USO DE ROLE-PLAY PARA ENSINAR OS FUNDAMENTOS DE DEVOPS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Ítalo Mendes da Silva Ribeiro

Coorientador: Prof. Dr. Allysson Allex de Paula Araújo

CRATEÚS

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M871c Mota, Lucas Lima.
Do Conceito à Interpretação da Prática: Explorando o Uso de Role-Play Para Ensinar os Fundamentos de DevOps / Lucas Lima Mota. – 2024.
78 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Sistemas de Informação, Crateús, 2024.
Orientação: Prof. Me. Ítalo Mendes da Silva Ribeiro.
Coorientação: Prof. Dr. Allysson Allex de Paula Araújo.

1. Aprendizagem ativa. 2. Role-Play. 3. Engenharia de Software. 4. DevOps. I. Título.

CDD 005

LUCAS LIMA MOTA

DO CONCEITO À INTERPRETAÇÃO DA PRÁTICA: EXPLORANDO O USO DE ROLE-PLAY PARA ENSINAR OS FUNDAMENTOS DE DEVOPS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em:

BANCA EXAMINADORA

Prof. Me. Ítalo Mendes da Silva Ribeiro (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Allysson Allex de Paula Araújo (Coorientador)
Universidade Federal do Cariri (UFCA)

Prof. Dr. Awdren de Lima Fontão (Coorientador)
Universidade Federal do Mato Grosso do Sul (UFMS)

Prof. Me. Francisco Anderson de Almada Gomes
Universidade Federal do Ceará (UFC)

Prof. Dr. Rodrigo Pereira dos Santos
Universidade Federal do Estado do Rio de Janeiro
(UNIRIO)

Dedico este trabalho a minha família e aos amigos que acompanharam minha trajetória e torceram por todas as minhas conquistas.

AGRADECIMENTOS

Agradeço, primeiramente, a minha família, mas principalmente a minha mãe, Galvany, por nunca ter desistido de incentivar meus estudos.

Agradeço a todos os professores que contribuíram na minha formação.

Aos amigos de infância, e aqueles que fiz durante minha jornada acadêmica, por estarem sempre ao meu lado e me ajudaram a permanecer forte.

Ao prof. Dr. Allysson Allex de Paula Araújo por me coorientar e por toda a dedicação e paciência durante a caminhada do desenvolvimento do meu trabalho.

Ao prof. Me. Ítalo Mendes da Silva Ribeiro por sempre me auxiliar e além disso fazer parte da minha banca e também coorientar este trabalho.

Ao prof. Dr. Awdren de Lima Fontão por aceitar ser meu coorientador e contribuir com seus conhecimentos para o desenvolvimento deste trabalho.

“Todo progresso ocorre fora da zona de conforto.”

(Michael John Bobak)

RESUMO

A fim de amenizar a problemática de aulas demasiadamente expositivas no ensino de Engenharia de Software (ES), muitos docentes exploram o uso de metodologias ativas, entre elas o *Role-Play* (RP). Em suma, RP proporciona que estudantes interpretem papéis específicos dentro de um contexto, promovendo a construção por meio de uma experiência prática, colaborativa e crítica. Todavia, verifica-se que, apesar de proeminente, tal metodologia ainda não foi explorada no contexto de DevOps, área esta bastante importante e utilizada no contexto de ES. Diante dessa oportunidade, o presente trabalho tem como objetivo investigar as experiências e percepções de estudantes de graduação quanto a adoção de um modelo de ensino baseado em RP para aprendizado de DevOps. Sob o ponto de vista metodológico, enquadra-se esta pesquisa em uma perspectiva empírica, experimental e descritiva. Para coleta de dados, utilizou-se questionários estruturados e observação-participante com três turmas distintas de discentes, totalizando uma amostra de 30 pessoas. A análise de dados explorou o uso de estatística descritiva e Análise Temática de Conteúdo. Em termos de resultados, verificou-se que, 93,3% dos participantes concordaram que a dinâmica de RP foi um método de ensino adequado. Além disso, 93,3% dos alunos também concordaram que a dinâmica de RP contribuiu para a aprendizagem. Resultados qualitativos também evocaram evidências positivas sobre a experiência vivenciada, especialmente em decorrência da aprendizagem cooperativa e adoção de ferramentas da indústria. Em termos de contribuição para academia, o modelo proposto baseado em RP oferece uma oportuniza rica para estudantes vivenciarem situações reais aderentes à demandas do mercado de trabalho. Por sua vez, para a indústria, tem-se que o usufruto de RP promove a interação e colaboração aos estudantes, bem como a vivência em situações análogas à realidade, formando assim, profissionais mais preparados sob o ponto de vista social e técnico.

Palavras-chave: Aprendizagem ativa. *Role-Play*. Engenharia de Software. DevOps

ABSTRACT

In order to alleviate the problem of excessively expository classes in the teaching of Software Engineering (SE), many professors explore the use of active methodologies, including *Role-Play* (RP). In short, RP allows students to interpret specific roles within a context, promoting construction through a practical, collaborative and critical experience. However, it appears that, despite being prominent, this method has not yet been explored in the context of DevOps, an area that is quite important and used in the context of SE. Given this opportunity, this work aims to investigate the experiences and perceptions of undergraduate students regarding the adoption of a RP based teaching model for learning DevOps. From a methodological point of view, this research is framed within an empirical, experimental, and descriptive perspective. For data collection, structured questionnaires and participant observation were used with three different groups of students, totaling a sample of 30 people. Data analysis explored the use of descriptive statistics and Thematic Content Analysis. In terms of results, it was found that 93.3% of participants agreed that RP dynamics was an appropriate teaching method. Furthermore, 93.3% of students also agreed that the RP dynamics contributed to learning. Qualitative results also evoked positive evidence about the experience, especially as a result of cooperative learning and adoption of industry tools. In terms of contribution to academia, the proposed model based on RP offers a rich opportunity for students to experience real situations that adhere to the demands of the job market. In turn, for the industry, the use of RP promotes interaction and collaboration for students, as well as experience in situations analogous to reality, thus forming more prepared professionals from a social and technical point of view.

Keywords: Active learning. *Role-Play*. Software Engineering. DevOps

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CD	Implantação Contínua
CDE	Entrega Contínua
CI	Integração contínua
ES	Engenharia de Software
GA	Gamificação
JS	Jogos sérios
NPC	*Non-Player Character*
PO	*Product Owner*
REST	*Representational State Transfer*
RFs	Requisitos Funcionais
RP	*Role-Play*
RPG	*Role-Playing Game*
SCV	Sistema de Controle de Versão
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de Software (ES) trata da aplicação sistemática, disciplinada e quantificável de princípios de engenharia no desenvolvimento, operação, manutenção e evolução de software (VALENTE, 2020). Tal área, bastante dinâmica, lida com diferentes desafios para a formação adequada dos discentes rumo a uma carreira de sucesso (ZUPPIROLI *et al.*, 2012). Logo, há uma necessidade constante de estruturar o processo de ensino-aprendizagem de forma a tornar o ambiente acadêmico mais engajador e alinhado com as situações e dificuldades da realidade do mercado de trabalho (SCHOTS *et al.*, 2009).

Dentre as áreas exploradas no ensino e na prática de ES, DevOps tem se destacado como um tópico de grande relevância (JABBARI *et al.*, 2016). De acordo com Kim *et al.* (2014), o DevOps é um movimento que busca unificar as culturas de desenvolvimento (*Dev*) e operação (*Ops*) para permitir a implantação mais rápida e ágil de sistemas. Ela envolve o compartilhamento de tarefas em uma equipe com total responsabilidade por seu serviço e sua pilha de tecnologia, desde o desenvolvimento até a implantação e suporte (PERERA *et al.*, 2017).

A implementação do DevOps varia entre as organizações, mas seus fundamentos são baseados em três pilares essenciais: Integração contínua (CI) (em inglês *Continuous Integration*), Entrega Contínua (CDE) (em inglês *Continuous Development*) e Implantação Contínua (CD) (em inglês *Continuous Deployment*) (FOWLER, 2006; PITTET, 2021; UNITY, 2023). Nesse sentido, recentemente diferentes pesquisadores (CHRISTENSEN, 2016; PINNA-DÉRY, 2019; ALVES; ROCHA, 2021) tem buscado explorar propostas de cursos aderentes ao ensino de DevOps que podem ser usados como material para educadores iniciantes nessa área.

Diante desse cenário, tem-se apontado como necessário e urgente discutir abordagens pedagógicas eficientes para o ensino de DevOps, considerando os desafios específicos envolvidos (por exemplo, complexidade conceitual, dificuldade em simular ambientes reais, falta de recursos educacionais, etc.) (RONG *et al.*, 2017; KUUSINEN; ALBERTSEN, 2019; CAPOZUCCA *et al.*, 2019). Tal questão converge com a premissa de que a metodologia de ensino-aprendizagem de ES deve ir além dos conceitos, técnicas e métodos, compreendendo habilidades e atitudes necessárias para o mercado de trabalho e buscando a excelência na educação (LIMA *et al.*, 2020).

Nesse contexto, as metodologias ativas de aprendizagem se consolidam como fundamentais para envolver os estudantes e promover competências baseadas na ação-reflexão-ação (BONWELL; EISON, 1991). Em particular, o *Role-Play* (RP) tem se destacado como uma metodologia ativa que contribui significativamente para o aprendizado e interação entre os alunos

(ALKIN; CHRISTIE, 2002). De acordo com Rabelo e Garcia (2015), o RP envolve os alunos atuando em papéis específicos dentro de um contexto determinado, promovendo a construção do conhecimento por meio da reflexão crítica. No contexto do ensino de ES, o RP tem sido explorado de forma proeminente devido à sua capacidade de oferecer uma experiência próxima à realidade do mercado de trabalho e promover a aprendizagem ativa (ZOWGHI; PARYANI, 2003; HENRY; LAFRANCE, 2006). Não obstante, apesar da pertinência quanto ao uso do RP para fins educacionais, percebe-se, na literatura de ES, uma lacuna quanto ao estudo de abordagens de ensino baseada em RP para o contexto específico de DevOps.

Considerando a motivação apresentada, este trabalho se justifica perante a relevância em compartilhar experiências sobre o uso do RP como ferramenta pedagógica para aprimorar o ensino de DevOps. Assim, a presente pesquisa assume um escopo experimental e quali-quantitativo com o objetivo de responder à seguinte questão de pesquisa: *Quais são as experiências e percepções de estudantes de graduação sobre a adoção de RP para o aprendizado de DevOps?* Para lidar com tal indagação, este trabalho inicialmente propõe um modelo de ensino baseado em RP e, em seguida, discute-se uma avaliação com três turma, totalizando 30 alunos, visando investigar a adoção desse modelo como abordagem pedagógica.

Do ponto de vista acadêmico, o modelo de ensino proposto oferece uma oportunidade para os estudantes de ES vivenciarem situações reais e desafiadoras do mercado de trabalho. Ao atuar em um contexto simulado, interpretando papéis específicos e lidando com problemas e decisões do dia-a-dia de uma equipe DevOps, os discentes têm a chance de aplicar os conceitos teóricos aprendidos em sala de aula de forma prática e reflexiva. Isso ajuda a consolidar o conhecimento adquirido, além de desenvolver habilidades essenciais, como trabalho em equipe, comunicação, resolução de problemas e tomada de decisões. Além disso, o uso do RP estimula a participação ativa dos alunos(as), promovendo o engajamento e a motivação para aprender.

Em relação à contribuição para a prática/indústria, a adoção do RP como estratégia de ensino de DevOps contribui para a formação de profissionais mais preparados e aptos a lidar com os desafios da área. Ao vivenciar situações similares às encontradas em um ambiente de trabalho real, os alunos desenvolvem competências e ganham experiência na aplicação das práticas e ferramentas do DevOps. Além disso, o usufruto de RP promove a interação e a colaboração entre os estudantes, estimulando a construção coletiva de conhecimento e a capacidade de lapidar as habilidades interpessoais que são altamente valorizadas na indústria de software.

1.1 Objetivos

O presente trabalho tem como objetivo geral investigar as experiências e coletar as percepções de estudantes de graduação sobre a adoção de RP para o aprendizado de DevOps.

Em relação aos objetivos específicos, tem-se:

- Propor um modelo pedagógico baseado em RP para ensino de DevOps;
- Analisar a percepção de estudantes de graduação em relação à adoção do RP para o aprendizado de DevOps;
- Explorar as experiências vivenciadas por estudantes de graduação durante as sessões de RP, destacando os aspectos positivos e perspectivas de melhorias percebidas sobre tal processo.

1.2 Estrutura do Trabalho

Em sua organização esse trabalho possui um total de nove capítulos, incluindo a presente introdução, onde os mesmos são estruturados da seguinte forma:

- **Capítulo 2 - Fundamentação Teórica:** são abordados os principais conceitos acerca do contexto que constituem este trabalho: DevOps , CI, CDE e CD, RP como Estratégia de Ensino-Aprendizagem e GitHub Actions;
- **Capítulo 3 - Trabalhos Relacionados:** busca-se analisar os principais trabalhos que ajudaram na construção desta pesquisa;
- **Capítulo 4 - Procedimentos Metodológicos:** tem por objetivo descrever os aspectos relacionados ao método científico adotado neste trabalho;
- **Capítulo 5 - D&O: Modelo de Ensino** tem por objetivo descrever a proposta do modelo de ensino baseado em RP que será usado para aprimorar o ensino de DevOps;
- **Capítulo 6 - Resultados e Análises:** são apresentados os próximos passos planejados para conclusão da presente pesquisa.
- **Capítulo 7 - Discussão:** promove-se uma discussão geral sobre os achados e resultados obtidos à luz da literatura;
- **Capítulo 8 - Limitações:** aborda-se uma análise sobre as limitações da pesquisa a estratégias para mitigação de tais ameaças;
- **Capítulo 9 - Considerações Finais:** são discutidas as contribuições, limitações e trabalhos futuros desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem por objetivo expor uma fundamentação que possibilite a compreensão dos elementos teóricos que embasam esta pesquisa. De início, na Seção 2.1, é exposto os fundamentos do DevOps, sendo enfatizado respectivamente os conceitos de Integração contínua (CI), Entrega Contínua (CDE) e Implantação Contínua (CD). Posteriormente, na Seção 2.2, são apresentados os conceitos do *Role-Play* (RP) como estratégia de ensino-aprendizagem. Finalmente, na Seção 2.3, contextualiza-se o GitHub Actions e sua utilidade enquanto ferramenta para viabilizar a prática de DevOps.

2.1 Fundamentos de DevOps

Valente (2020) destaca que o conceito de *DevOps* surgiu mediante uma problemática existente nas organizações tradicionais da área de Tecnologia da Informação, onde possuíam uma divisão entre dois departamentos: o departamento de sistemas (desenvolvimento), formado por desenvolvedores, programadores, analistas, arquitetos, etc. E o departamento de suporte (operações), no qual ficavam alocados os administradores de rede, administradores de bancos de dados, técnicos de suporte, técnicos de infraestrutura, etc. Existia uma tensão natural entre as duas equipes, uma vez que os desenvolvedores eram responsáveis pela escrita e alteração de programas, já o time de operações eram reesposáveis pela implantação e estabilidade dos mesmos. Dessa problemática surgiu então a ideia de reunir as duas equipes para que cooperassem com a entrega rápida e confiável de software o que provou ser eficaz, levando o papel do engenheiro de software a abranger o ciclo de vida completo do sistema, desde o projeto, ao desenvolvimento e o suporte à produção (KIM *et al.*, 2014).

No entanto, é importante salientar que *DevOps* não intervém necessariamente na criação de um novo profissional, que teria como função tanto o desenvolvimento como também a implantação de sistemas, mas sim a proximidade entre as equipes de desenvolvimento e a de operações, objetivando uma implementação muito mais ágil e menos angustiante (VALENTE, 2020). Sendo assim, *DevOps* se posiciona como uma metodologia de desenvolvimento que visa preencher a lacuna entre o departamento de sistemas (desenvolvimento) e o departamento de suporte (operações), enfatizando comunicação e colaboração, integração contínua, garantia de qualidade, e entrega com implantação automatizada utilizando um conjunto de práticas de desenvolvimento (JABBARI *et al.*, 2016)

Farroha e Farroha (2014) elenca alguns objetivos do DevOps como, por exemplo, a entrega de valor comercial mensurável por meio da prestação de serviços contínua e de alta qualidade, ênfase na simplicidade e a agilidade em todas as áreas, incluindo tecnologia, processo e fatores humanos e a transposição de barreiras entre o desenvolvimento e as operações, permitindo confiança e propriedade compartilhada, apoiando a inovação e incentivando a colaboração. Um motivo importante que justifica a adesão ao *DevOps* em um fluxo de trabalho é a possibilidade de tornar as atualizações menores, frequentes e automatizar partes do processo de software, para que os times de desenvolvimento e operações possam trabalhar com mais eficiência, em vez de ter que fazer, por exemplo, vários testes de compilação para cada pequena atualização (SWARTOUT, 2012). Dessa forma, o tempo necessário para implantar alterações de código no ambiente é reduzido e a empresa e o cliente ganharão mais valor com o aumento da qualidade do trabalho (KIM *et al.*, 2021).

No cenário ideal de uma abordagem baseada em DevOps desse modelo, os desenvolvedores recebem *feedback* constante sobre seu trabalho, o que permite que os desenvolvedores resolvam possíveis erros mais rapidamente, permitindo também que implementem, validem e implantem novas alterações de código no ambiente de produção em um ritmo constante (PRIYADARSINI *et al.*, 2020). Com isso, é importante dizer que *DevOps* defende sempre a automatização de todas as etapas existentes na construção de um sistema, objetivando uma melhor corretude, até chegar no ambiente de produção e também monitorar o seu correto funcionamento (VALENTE, 2020). Tal perspectiva integrada, complementar e pluri conceitual pode ser vista no ciclo de vida sintetizado na Figura 1 (REDONDO *et al.*, 2022).

Figura 1 – Ciclo de vida do DevOps.

Fonte: Deploy (2023)

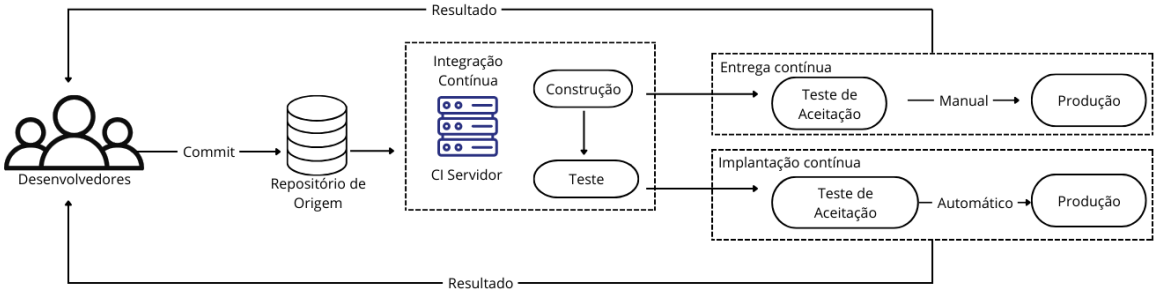
No ciclo de vida do *DevOps* apresentado na Figura 1, é possível obter uma visão geral sobre as fases que o compõe, sendo elas:

- **Processo de planejamento (*Plan*)**: É criado um relatório inicial de investigação para construir um *business case*, é feito o levantamento de requisitos funcionais e não funcionais, é construído um glossário, é também definido um modelo conceitual inicial, projetar arquitetura e priorizar as funcionalidades e distribuí-las entre as iterações (NICOLLETTI, 2002).
- **Processo de codificação (*Code*)**: A codificação é a transformação das ideias e requisitos em um sistema funcional. As definições de como o sistema deverá ser, incluindo seu comportamento, sua aparência, sua interface, são transformadas em código de máquina funcional, executando efetivamente suas designações (GOMES, 2013).
- **Processo de construção (*Build*)**: O processo de *build* de um software é responsável por verificar se todos os componentes do nosso código fonte estão sendo integrados de maneira correta, onde esse processo composto pelos seguintes passos: Recuperar a última versão do controlador de fontes e Executar o *build*; Executar os planos de teste (unitários, integração, interface, etc); Executar o *Deploy* para o ambiente (MAGAZINE, 2012).
- **Processo de teste (*Test*)**: há inclusão de várias fases de testes como os testes feitos pelo próprio programador durante a codificação, sendo teste de unidade, teste funcional e teste de componente (NICOLLETTI, 2002).
- **Processo de liberação (*Release*)**: Um *release* de sistema é uma versão que é distribuída para os clientes. Cada *release* de sistema deve incluir nova funcionalidade ou se destinar a uma diferente plataforma de hardware (SOMMERVILLE, 2013).
- **Processo de implantação (*Deploy*)**: É o processo de liberar uma nova versão de um sistema para seus usuários (VALENTE, 2020).
- **Processo de operação (*Operate*)**: Envolve a manutenção, a monitorização e a resolução de problemas das aplicações em ambientes de produção (MICROSOFT, 2022).
- **Processo de monitoramento (*Monitor*)**: Envolve o processo de coleta, análise e uso de informações para acompanhar aplicativos e infraestrutura, para guiar decisões de negócios. O monitoramento é um recurso importante, porque fornece informações sobre seus sistemas e trabalho (CLOUD, 2022).

De modo complementar, por meio da Figura 2, é possível visualizar as etapas de DevOps e a presença de três conceitos fundamentais: Integração contínua (CI), Entrega Contínua(CDE) e da Implantação Contínua (CD). O fluxo inicial começa com com os desenvolvedores efetuando o *commit* das alterações e em seguida, após ter passado pelo repositório de origem,

começa propriamente dita a fase da CI. Dentro do CI, ocorre a execução do *build* e a realização dos testes automatizados. Por sua vez, no estágio seguinte, é realizada a CD, quando ocorre a implantação em ambiente de homologação e a realização dos teste de integração, por fim na fase de CD, é feita a implantação das alterações em ambiente de produção (STÅHL; BOSCH, 2017). Dessa forma, a diferença básica entre os três é que no CI o time de desenvolvimento ao integrar o código no Sistema de Controle de Versão (SCV) é verificado por uma compilação automatizada para detectar erros de integração o mais rápido possível (FOWLER, 2006). Já o CDE, sua *release* fica disponível a qualquer momento para implantação, porém precisa ser implantada manualmente, enquanto na CD a implantação é feita de forma totalmente automatizada (AZERI, 2017).

Figura 2 – A relação entre CI, CDE e CD.



Fonte: Adaptada de (SUNDMAN, 2013; PRINCE, 2016).

2.1.1 Integração Contínua

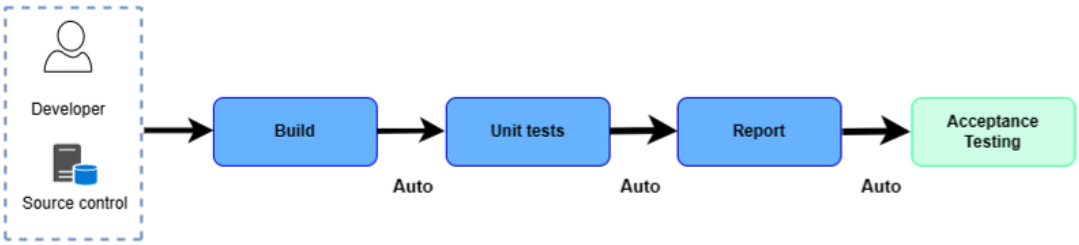
As primeiras discussões em torno de CI foram retratadas em um contexto de projetos fundamentados em orientação a objetos por Booch (1990). Nesse sentido, estabeleceu-se como objetivo principal a busca por integrações em curtos espaços de tempo, com novas funcionalidades executáveis a cada atualização. Tal concepção foi amplamente explorada e divulgada na metodologia *Extreme Programming* (XP) (BECK, 1999). Entretanto, Martin Fowler, com uma publicação em seu *blog*, foi o responsável por propagar o entendimento de CI, ampliando assim o nível de aceitação (HILTON *et al.*, 2016).

Em suma, CI é uma prática de desenvolvimento amplamente estabelecida na indústria de desenvolvimento de software a qual inclui controle de versão, gerenciamento de configuração de software, construção automatizada e teste de regressão (VASILESCU *et al.*, 2015; FITZGERALD; STOL, 2017). É importante salientar que a CI permite uma equipe, integrar e mesclar o trabalho de desenvolvimento, por exemplo código, com uma maior frequência, podendo ser

uma vez ao dia ou até mesmo várias vezes em um único dia (SHAHIN *et al.*, 2017). Cada integração é verificada por um teste automatizado de construção e regressão para detectar falhas o mais rápido possível (YANG *et al.*, 2020). Sem CI, cada desenvolvedor estaria trabalhando em diferentes versões do mesmo produto de software e não teria um produto de software integrado (MELLADO *et al.*, 2010).

A Figura 3 ilustra como ocorre as etapas da CI. Segundo Neto (2021), o processo tem seu início com uma submissão ao gerenciador de repositórios de uma modificação (integração) no código fonte do projeto. Em geral, o código submetido passa por algumas etapas antes de ser incorporado na versão final, são elas: *build*, teste e geração de relatório. O *build* consiste na compilação do código-fonte da aplicação a fim de gerar um arquivo executável, que em caso de sucesso, o processo segue para o estagio de teste. No estágio de teste, são executados testes automatizados a fim de verificar se o software não apresentou falhas após a incorporação das mudanças. Por fim, um relatório da CI é gerado informado os resultados obtidos nos estágios anteriores. Dessa forma, a CI consegue prover aos desenvolvedores *feedbacks* em cada integração, permitindo que erros possam ser detectados mais rápidos e proporcionando uma manutenção e evolução do sistema.

Figura 3 – Fluxo de CI.



Fonte: Adaptada de (CLOUD, 2023a).

Sendo assim, a CI permite que as empresas de software tenham um ciclo de *releases* mais curto e frequente, melhorem a qualidade do software e aumentem a produtividade de suas equipes (FITZGERALD; STOL, 2017). Esta prática inclui a construção e teste automatizados de software (LEPPÄNEN *et al.*, 2015). Para Duvall *et al.* (2007), algumas práticas são necessárias para que a CI funcione de forma adequadas:

- **Submeter com frequência no repositório as alterações**: um dos princípios centrais da CI é a integração precoce e frequente. Para que ela aconteça, é importante que os desenvolvedores subam seus códigos com frequência, mas sem alterar muitos componentes

de uma única vez, o ideal é que façam pequenas alterações, sempre executando os seus testes e em seguida enviando para o repositório.

- **Não submeter ao repositório código com defeito**: algo que não deve ser feito por um desenvolvedor é subir seus códigos com problemas para o repositório de controle de versão. O correto é sempre possuir um bom *script* de construção, que compile e teste o código de maneira repetível.
- **Restaurar erros no *build* imediatamente:** sempre que o *build* principal apresentar problemas, como erro de compilação, teste ou inspeção com falhas, incoerência com banco de dados, ou uma implementação errada, a solução para esse problema deve ter uma prioridade maior por parte da equipe. De acordo com Fowler (2006), não é preciso que toda a equipe se concentre no problema: basta a quantidade de pessoas necessárias para resolvê-lo. Entretanto, é importante que se aja com discernimento de que a priorização é fundamental.
- **Escrever testes automatizados:** os testes que devem ser executados para um sistema de CI devem ser automatizados. Escrever seus testes em uma estrutura *xUnit*, como *NUnit* ou *JUnit*, fornecerá a capacidade de executar esses testes de maneira automatizada.
- **Todos os testes e as inspeções devem passar**: para que o *build* seja aprovado, é preciso que os testes automatizados não apresentem problemas.
- **Executar builds privados**: com a intenção de precaver *builds* com erros, os desenvolvedores têm de simular uma integração local logo após realizar os testes unitários. Localmente, é feita uma integração a partir da obtenção das mudanças dos outros desenvolvedores que estão disponíveis no repositório, antecipando os possíveis problemas, tornando mais provável que a integração não falhe.
- **Evitar integrar códigos falhos**: se o *build* mais recente apresentar algum erro, é melhor esperar que seja consertado ou ajudar nessa tarefa pois utilizar essa *build* fará com que se perca tempo.

A partir do que foi discutido anteriormente, pode-se compreender que o CI é um processo padrão das organizações modernas de desenvolvimento de *software* de alta eficiência. As melhores empresas têm pipelines de CI robustos e não hesitam em mais investimentos em eficiência. Os benefícios da CI não se limitam à equipe de engenharia e se aplicam à toda a organização (ATLASSIAN, 2021).

2.1.2 Entrega Contínua

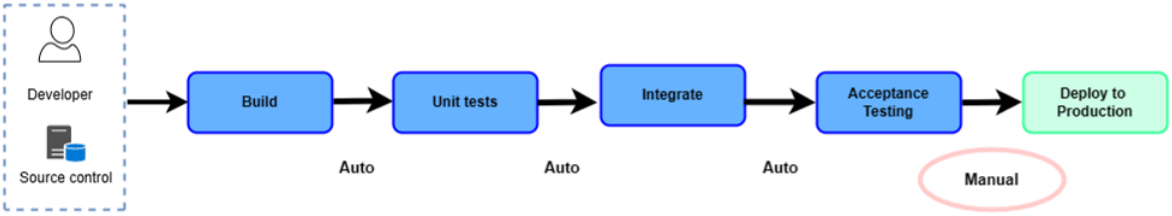
A perspectiva da CDE é relativamente nova e alcançou uma maior notoriedade em 2010, quando Humble e Farley publicaram um livro intitulado “Entrega contínua: Versões de software confiáveis por meio de construção, teste e automação de implantação” (HUMBLE; FARLEY, 2010). A CDE promove que equipes consigam produzir softwares valiosos, em ciclos curtos e garantam que esse sistema possa ser lançado de forma confiável a qualquer momento (CHEN, 2015). A CDE Ela visa simplificar o lançamento através de ciclos de *feedbacks* mais curtos através da automatização e utilização do processo de construção, implantação, teste e lançamento, entre desenvolvedores (KRUSCHE; ALPEROWITZ, 2014).

O proposito da CDE é garantir que um software se mantenha sempre pronto para o ambiente de produção, após passar por testes automatizados e verificações de qualidade (HUMBLE, 2010a; WEBER *et al.*, 2016). Para Chen (2015) e Humble (2010b), esta prática oferece vários benefícios, alguns deles são: redução de risco de implantação, redução de custos e a garantia de um *feedback* do usuário com mais rapidez. No entanto, implementar a CDE pode ser bastante desafiador (CHEN, 2015; CLAPS *et al.*, 2015; LEPPÄNEN *et al.*, 2015). O caminho para adoção da CDE não é tão trivial, mas, de acordo com Chen (2015), alguns dos desafios fundamentais incluem os seguintes pontos: 1) Alcançar adesão de uma ampla gama de partes interessadas cujos objetivos podem ser aparentemente diferentes, ou até mesmo conflitar com os da equipe que está conduzindo a implementação da CDE; 2) Obter suporte sustentado em um ambiente empresarial complexo e dinâmico; 3) Manter o ímpeto da equipe de desenvolvimento de aplicativos quando a migração de seus aplicativos para o CDE exige um esforço extenuante adicional por um longo período de tempo.

Em teoria, com a CDE, pode-se decidir fazer lançamentos diários, semanais, quinzenais ou o que for mais adequado aos requisitos de negócios. No entanto, se realmente for necessário obter os benefícios da CDE, devemos implantar na produção o mais cedo possível para se certificar de liberar pequenos lotes que são fáceis de solucionar em caso de problema (PITTET, 2021).

A CDE é uma extensão da CI, uma vez que a mesma implementa de modo automático todas as alterações de código em um ambiente de teste e/ou produção após o estágio de *build*. Ou seja, além de testes automatizados, você tem um processo de *release* automatizado e pode implementar seu aplicativo a qualquer momento clicando em um botão (PITTET, 2021). Um exemplo desse *pipeline* pode ser visto na Figura 4.

Figura 4 – Fluxo de CDE.



Fonte: Adaptada de (CLOUD, 2023a; KHAN *et al.*, 2020).

2.1.3 Implantação Contínua

CD é uma atividade que envolve a implantação de alterações em ambientes de produção ou do cliente de forma automática e contínua (WEBER *et al.*, 2016). Entretanto, o que diferencia a CD daCDE é seu ambiente de produção (ou seja, clientes reais): a finalidade da prática de CD é implantar de forma automática e constante cada mudança no ambiente de produção (SHAHIN *et al.*, 2017), mas é importante notar que a prática de CDE implica na prática de CD, mas o inverso não é verdadeiro (HUMBLE, 2010a).

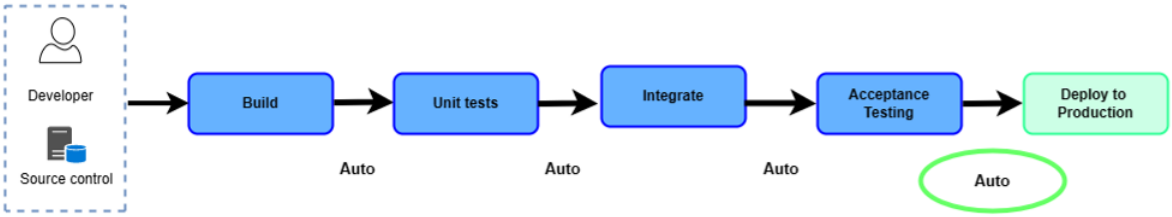
Ainda que a implantação final na CDE seja uma etapa manual, não se deve haver nenhuma etapa manual na CD. Assim sendo, na CD, no momento em que os desenvolvedores fazem uma alteração, a mesma é implantada na produção automaticamente por meio de um *pipeline* de CD. A prática da CDE é um procedimento baseado em *Pull* (procedimento responsável por baixar conteúdo de repositórios remotos fornecendo atualizações imediatas a repositórios locais) para o qual uma empresa decide quando implantar, enquanto a prática de CD é um procedimento baseado em *Push* (procedimento usado para enviar conteúdo de um repositório local para um repositório remoto) (SKELTON; O'DELL, 2016). Em outras palavras, o escopo da CDE não inclui o lançamento frequente e automatizado, pois possui um passo manual, que pode ocorrer a cada 15 dias geralmente efetuado por um gerente de projetos. Como resultado, o CDE tem o CD como uma continuação, mas como meconiado o CD possui todos seus passos automáticos.

Para Pittet (2021), a CD pode ser uma ferramenta poderosa para organizações de engenharia modernas. A implantação é a etapa final do '*pipeline* contínuo' geral que consiste na integração, entrega e implantação. A verdadeira experiência da CD é a automação até o nível em que o código é implantado na produção, testado para correção e automaticamente revertido quando errado ou aceito se correto.

Como descrito na Figura 5, a CD vai um passo além da CDE. Com essa prática, toda alteração que passa em todos os estágios do seu pipeline de produção é lançada para os clientes.

Nesse caso, entretanto, preconiza-se que não haja intervenção humana e apenas um teste com falha vai impedir a implementação de uma nova alteração na produção (PITTET, 2021).

Figura 5 – Fluxo da CD.



Fonte: Adaptada de Cloud (2023a), Khan *et al.* (2020).

Dentre as vantagens de CD, Valente (2020) destaca:

- CD reduz o tempo de entrega de novas funcionalidades. Por exemplo, suponha que as funcionalidades estão previstas para uma nova *release* de um sistema. No modo tradicional, todas elas devem ser implementadas e testadas, antes da liberação da nova *release*. Por outro lado, com CD, as funcionalidades são liberadas assim que ficam prontas. Ou seja, CD diminui o intervalo entre *release*. Sendo assim há um aumento de *release*, mas agora com um menor número de funcionalidades.
- CD torna novas *release* (ou implantações) um não-evento. Explicando melhor, não existe mais um dia D ou um *deadline* para entrega de novas *release*. Deadlines são uma fonte de stress para desenvolvedores e operadores de sistemas de software. A perda de um *deadline*, por exemplo, pode fazer com que uma funcionalidade somente entre em produção meses depois.
- Além de reduzir o stress causado por deadlines, CD ajuda a manter os desenvolvedores motivados, pois eles não ficam meses trabalhando sem receber *feedback*. Em vez disso, os desenvolvedores rapidamente recebem retorno — vindo de usuários reais — sobre o sucesso ou não de suas tarefas.
- Em linha com o item anterior, CD favorece experimentação e um estilo de desenvolvimento orientado por dados e *feedback* dos usuários. Novas funcionalidades entram rapidamente em produção. Com isso, recebe-se retorno dos usuários, que podem recomendar mudanças na implementação ou, no limite, o cancelamento de alguma funcionalidade.

2.2 Role-Play como Estratégia de Ensino-Aprendizagem

De acordo com Cheron (2019), a técnica de RP foi idealizada por Moreno e Moreno (2006) com a finalidade de "praticar" competências consideradas importantes para o mundo do trabalho, facilitando a comunicação e desenvolvendo habilidades entre os participantes. Cheron (2019) ainda relata que o RP mostra sua eficacia em qualquer modalidade dentro de instituições de ensino, onde a respectiva técnica é bastante eficaz na interpretação de uma determinada situação do dia a dia, onde o aluno, através de uma simulação, dramatiza ou interpreta papéis. Como resultado da encenação, todos os alunos envolvidos na atividade irão aprender algo sobre a situação, o contexto proposto e/ou os personagens (NESTEL; TIERNEY, 2007).

Nesse contexto, há também de se destacar a diferença entre *Role-Play* (RP), Jogos sérios e Gamificação. Conforme mencionado anteriormente, no RP, a pessoa interpreta de acordo com um papel, sendo ele próprio ou personagem em uma situação específica (LADOUSSE, 1987). Ou seja, é uma forma de narrativa interativa em que os participantes assumem diferentes papéis e se envolvem em diálogos e ações improvisados ou roteirizados para simular as experiências desses personagens. Por sua vez, os Jogos sérios, são jogos utilizados com propósito de ensino-aprendizagem ou treinamento e não apenas de entretenimento (TANAKA *et al.*, 2013; GEDIGAMES; GAMES, 2014). Eles envolvem o uso de tecnologias de jogos digitais com o propósito de simular problemas do mundo real. Suas aplicações têm sido eficientes nas áreas de saúde, defesa, negócios, turismo, entre outras (TANAKA *et al.*, 2013; MATTAR, 2010; NOVAK, 2011). Por outro lado, BUSARELLO *et al.* (2014) diz que a Gamificação tem como base a ação de se pensar como em um jogo, utilizando as sistemáticas e mecânicas do ato de jogar em um contexto fora de jogo. Tanaka *et al.* (2013) considera que gamificação abrange a utilização de mecanismos de jogos para a resolução de problemas e para a motivação e o engajamento de determinado publico.

Para Porter (1989), a palavra que dá nome a técnica, RP, é autoexplicativa já que *role* é o papel que o aluno irá desempenhar na atividade, que pode ser ele mesmo ou um personagem, e *play* significa que a execução ocorrerá em um ambiente seguro no qual os alunos podem ser criativos e lúdicos. Na visão de Cechin (2002), com o RP, os alunos desenvolvem valores modelando e usando exemplos da vida real, desempenhando papéis atribuídos sobre uma determinada situação social a fim de compreendê-la de forma mais completa. Kuśnierek (2015) relata que estas estratégias são parte de um ensino em que as crenças e valores do aluno são usados na interpretação da realidade e da sociedade, contribuindo para tornar a aprendizagem

um processo ativo de construção de significado a partir da experiência e da reflexão sobre essa experiência do mundo.

O RP não precisa de muitas ferramentas para acontecer. Papéis ou cartões com as designações dos personagens, e o ambiente no qual a atividade se passará, já se caracterizam como o ponto de partida para que os alunos deixem fluir a imaginação e comecem a atividade (NASCIMENTO, 2019). Os personagens podem ser frequentemente designados pelo professor, porém os alunos têm a liberdade de escolher quem irão interpretar, assim como de criar suas falas. De acordo com Kuśnierek (2015), o primeiro objetivo do RP é impulsionar a interação dos alunos em sala de aula. Através do RP, os professores podem treinar as habilidades dos alunos em qualquer situação social (KUŚNIEREK, 2015).

De acordo com Haycraft (2010), o RP possui três elementos característicos, a saber: 1) o que os personagens desejam; 2) quem eles são; 3) e seu humor ou suas atitudes na situação estabelecida. Estes mesmos três elementos são afetados por como a situação se desenrola, permitindo que os alunos entrem em negociação e produção de sentido. Para este autor, quanto maior a caracterização, mais os alunos se envolvem na atividade. Além disso é apropriado dizer que o mesmo necessita de regras, para garantir que as ações dos personagens tenham coerência com o universo no qual a história se passa. Uma aventura de RP assemelha-se a uma brincadeira de faz de conta, porém com um sistema de regras para normatizar a brincadeira, garantindo que a imaginação de um jogador não ocupe o espaço da imaginação do outro, garantindo que todos os jogadores possam ter uma participação ativa na aventura (MARINS, 2017). Estas normas podem ser brandas, ou rigorosas, dependendo do sistema de regras escolhido pelos jogadores. As regras em uma aventura de RP além de manterem a coerência na aventura, também auxiliam em manter o fator dificuldade no jogo, o desafio, o que estimula a elaboração de estratégias e ações cooperativas entre os jogadores (SILVA *et al.*, 2009). Não diferente de outros métodos de aprendizagem ativa, o RP gera também benesses aos seus participantes.

Conforme se pode perceber, o RP como método de ensino possibilita que o aluno assuma em sala de aula o papel de uma pessoa da vida real, a fim de proporcionar a imersão em situações do cotidiano, preparando o aluno para aplicar no seu dia a dia o aprendizado adquirido durante as técnicas (RABELO; GARCIA, 2015). Além de encorajar o aluno a partilhar o seu ponto de vista em sala, o RP, também contribui para o aumento da responsabilidade e da capacidade crítica (CHERON, 2019). Em pesquisa realizada por Cardoso *et al.* (2009), mostra-se que a utilização de RP fez aumentar significativamente a participação dos alunos durante as

aulas, criando um ambiente favorável para a aprendizagem, permitindo assim que os alunos desenvolvam competências. Ao usufruir do RP, o aluno age de acordo com um papel pré-definido em uma conjuntura específica (CARDOSO *et al.*, 2009). Sendo assim, a definição que melhor estabelece a condução das tarefas executadas na presente pesquisa, pois com o uso do RP os participantes devem ter autonomia para praticar os conhecimentos obtidos e desenvolver as aptidões necessárias para cada situação.

2.3 GitHub Actions

Lima *et al.* (2014) relatam que, há alguns anos, o *GitHub*, uma plataforma de hospedagem para projetos de software, passou a ter muita popularidade entre um número considerável de desenvolvedores de software em todo o mundo. Esta plataforma oferece hospedagem de código e controle de versão, como outras plataformas já fizeram no passado, como por exemplo, *SourceForge*, *Assembla* e *BitBucket*. Na prática, o *GitHub* não oferta simplesmente um serviço de hospedagem de código, como seus concorrentes vinham fazendo há muito tempo. Também é uma ferramenta online fácil de usar e barata (ou mesmo gratuita em sua versão básica) para desenvolvimento colaborativo de software e muitos recursos de suporte à comunidade de desenvolvedores (LIMA *et al.*, 2014)

É importante salientar que no *GitHub*, um usuário pode criar um repositório de código e efetuar um *push* (envio de alterações do repositório local para repositório remoto) para ele. Cada repositório pode possuir uma lista de colaboradores, onde eles podem fazer alterações no conteúdo do repositório e efetuar revisões das contribuições que são enviadas ao repositório uma vez que podem ser aceitas ou recusadas. Uma outra forma de contribuir com um projeto é fazendo um *fork*, onde essa ação irá duplicar o repositório, permitindo que os colaboradores trabalhem de forma independente, armazenando as mudanças apenas para seu próprio *Fork*. Após a conclusão da tarefa, poderá ser enviada às alterações para o repositório original através de um *pull request* (proposta para mesclar as alterações de um *branch* para o outro) (LIMA *et al.*, 2014). Em um outro momento, um colaborador do repositório original revisa as mudanças contidas na solicitação de *pull* e decide se deseja aceitá-las ou recusá-las, e, uma vez que o novo código é aceito no repositório original, seu autor se torna um dos colaboradores do projeto.

Além de todas as facilidades do *GitHub* supracitadas, pode-se contar com o *GitHub Actions*, cujo foco consiste em prover uma plataforma de CI/CD que permita automatizar as compilações, testar o *pipeline* de implantação, sendo ainda possível criar fluxos de trabalho

que criam e testam cada *pull request* do repositório, ou implantar *pull request* mesclados em produção. A automação das tarefas acontece com base em vários *triggers*, como, por exemplo: *commits*, *pull request*, *issues*, *comments* e etc. Podendo ser facilmente compartilhadas de um repositório para outro, torna-se mais fácil automatizar o que desenvolvedores criam, testam e implantam projetos de software (KINSMAN *et al.*, 2021). Canorea (2021) expressa que o *GitHub Actions* usa pacotes de código em contêineres *Docker*, que são executados em servidores *GitHub* e que, por sua vez, são compatíveis com qualquer linguagem de programação. Isso os faz rodar em servidores locais e nuvens públicas.

Visando entender como funciona e de quais partes ou ações é composto o *GitHub Actions*, Canorea (2021) define os princípios básicos das *Actions* do *GitHub*:

- ***Step***: Consiste em um conjunto de tarefas para poder executar um trabalho. Eles podem executar comandos ou ações.
- ***Work***: É um conjunto de etapas que executam nosso processo. Os trabalhos podem ser executados de forma independente ou sequencial dependendo se o sucesso do nosso trabalho depende do anterior.
- ***Workflow***: É um procedimento automatizado composto por uma ou mais tarefas que são adicionadas a um repositório e podem ser ativadas por um evento. Eles são definidos por arquivos *YAML* e com eles você pode construir, testar, empacotar, retransmitir ou implantar um projeto.
- ***Event***: Essas são atividades específicas que acionam a execução de um fluxo de trabalho.
- ***Actions***: É o menor bloco de construção de um fluxo de trabalho e pode ser combinado como etapas para criar um trabalho.
- ***Runner***: É uma máquina com o aplicativo *GitHub Actions* já instalado, cuja função é aguardar a disponibilização do trabalho para então poder executar as ações e reportar o andamento e os resultados.

De acordo com Canorea (2021), uma das características que determina o sucesso do *GitHub Actions* é sua flexibilidade, que se traduz a uma abertura e também uma ampla gama de possibilidades nas quais os desenvolvedores podem liberar sua criatividade. Nesse sentido, algumas vantagens podem ser destacadas: **1) Desenvolvimento dentro do próprio *GitHub***; a opção *Actions* está totalmente integrada ao *GitHub*, portanto, não requer um site externo. Isso significa que podemos gerenciar tudo no mesmo lugar onde temos as funções relacionadas ao repositório; **2) Uma grande variedade de modelos de CI** a plataforma oferece diversos

modelos para todos os tipos de configurações de CI, o que facilita muito o início do trabalho. Além disso, tem a opção de criar os próprios modelos e publicá-los no *GitHub Marketplace*; e **3) Permite testar as configurações de vários contêineres**, uma vez que adiciona-se o *Docker* e suporte a arquivos de composição ao fluxo de trabalho.

3 TRABALHOS RELACIONADOS

Com a finalidade de elucidar os trabalhos relacionados à presente pesquisa, identifica-se a seguir uma explanação orientada nas pesquisas empíricas e exploratórias que tratam sobre o uso de *Role-Play* (RP) no ensino de Engenharia de Software. Por fim, é feita uma síntese comparativa entre as referências visando evidenciar a lacuna de pesquisa explorada.

Dixon e Jagodzinski (2003) fizeram uma análise qualitativa de dados de exercícios de RP usufruindo de uma etnografia não convencional com base no método *thinking aloud* onde foi utilizado em estudos de caso simulados dos domínios de aplicação. A abordagem analítica adotada no trabalho centrou-se em torno de um exercício de dramatização de pensamento em voz alta com subsequente análise de dados qualitativos. A prática exigiu que um participante (um especialista no domínio) desempenhe um papel, onde o mesmo deve "pensar em voz alta" e durante a execução do exercício o mesmo foi gravado e transcrito para posteriormente ser analisado permitindo a viabilidade de análises nos sues dados. Os autores concluíram que os exercícios de RP de pensar em voz alta tornaram visíveis as práticas de trabalho do usuário, revelando detalhes anteriormente ocultos sobre o que estava sendo feito e como estava sendo executado.

Henry e LaFrance (2006) apresentaram cinco exercícios de RP que podem ser explorados em um curso de ES. Dentro desses exercícios foram incluídas instruções para que os alunos pudessem administrar os exercícios de RP e um conjunto de perguntas para uma sessão de *debriefing*. Os autores relataram que nos exercícios, os alunos desempenharam papéis de apresentador e revisor, onde os grupos de projeto se revezaram apresentando alguma parte de seu projeto para um dos outros grupos. Ao final de cada exercício, foi apresentado um questionário a cada aluno que participou das práticas. Houve então um total de 175 respostas. A maior parte dos alunos, 96%, avaliaram os exercícios como um bom aproveitamento do tempo das aulas, já 98% disseram que os exercícios deveriam ser usados em cursos futuros e 99% acharam que os exercícios eram relevantes para o material do curso, 89% relataram ter conseguido aprender o que foi proposto com os exercícios. Entretanto, 5% dos alunos disseram que preferiam ter uma aula do que realizar os exercícios. Os resultados mostraram que o RP se mostrou uma ferramenta rica e que pode introduzir formas singulares de aplicabilidades das questões que compõem a ES, onde quase todos os alunos que consideraram uma ferramenta de aprendizagem valiosa.

Nakamura *et al.* (2012) propos um exercício de RP no qual um agente de software incentiva os alunos a se concentrarem no desempenho de seus papéis, a compartilhar informações

e a resolver problemas de exercícios a partir da perspectiva de seus papéis. Para a melhor execução dos exercícios de RP se estabeleceu que os alunos consigam: 1) coletar e trocar informações para solucionar os problemas surgidos no projeto virtual; 2) Compreender o papel e a perspectiva da parte interessada que lhe foi designada para desempenhar e afirmar os interesses do papel na resolução dos problemas; 3) Equilibrar os interesses das diferentes partes, encerrar as discussões e tomar uma decisão *win-win* para todas as partes. Com base nos questionários aplicados, 96% dos participantes entenderam que o exercício de RP ajudou a entender o que foi ensinado na aula, 96% também afirmaram que o RP permitiu os mesmo serem mais eficaz em situações reais e 77% responderam de forma afirmativa que o RP permitiu-lhes imaginar como será o seu futuro trabalho. Os resultados mostraram que a aplicação dos métodos propostos obteve respostas mais positivas dos alunos do que a utilização dos métodos convencionais, indicando que os métodos propostos foram eficazes.

Redondo *et al.* (2014) relataram uma experiência na combinação de aprendizagem colaborativa e estratégias de RP com Tecnologias Web 2.0 em disciplinas de ES. O objetivo principal da atividade proposta foi reforçar as competências sociais e de design, que são essenciais para o sucesso de qualquer processo de desenvolvimento de software. A atividade proposta seguiu um ensino centrado no aluno, onde os autores adotaram uma abordagem baseada em projetos combinados com uma perspectiva pedagógica imersiva que incorpora RP. Os autores salientaram que as experiências foram colocadas em prática em um período de três anos, onde alguns resultados positivos, puderam ser observados, como um grau de motivação elevado por parte dos discentes por terem de resolver um problema real num contexto de jogo, foi constatado também que a utilização de um ambiente Web 2.0 foi um componente chave de aprendizagem do sistema e particularmente atrativa para estudantes de engenharia de telecomunicações, especialmente interessados em novas tecnologias.

Decker e Simkins (2016) debateram a utilização de RP em um curso de Processos de Desenvolvimento de Jogos (PDJ), a fim de promover um maior envolvimento por parte dos discentes buscando um alto grau de interações para explorar o conteúdo do curso da melhor maneira possível. Os autores buscaram criar uma estrutura de RP que se adequasse às necessidades do curso de PDJ, onde o mesmo dispôs da seguinte estrutura: 1) Primeira semana, os alunos tiveram conhecimento da narrativa principal do semestre; 2) Segunda semana, houve uma prática sobre as tendências emergentes na indústria de jogos; 3) Terceira semana, houve um *brainstorm* com o objetivo de incentivar os alunos a desenvolverem suas ideias para os projetos; 4) Quarta

semana, focou-se em questões comerciais; 5) Quinta semana, foram abordado os projetos que a empresa tinha atualmente em produção; 6) Na sexta e sétima semanas, os alunos s receberam uma tarefa para apresentar descrições de desenvolvimento de software; 7) Décima semana, os alunos viram temas e leituras sobre questões jurídicas e análise de riscos; 8) Na décima terceira e décima quarta houve a apresentação final dos projetos desenvolvidos ao longo do semestre. As observações se deram através de questionamentos informais, feitos aos alunos no final do semestre, também foi perguntado aos alunos sobre os cenários de dramatização e o impacto dos cenários no aprendizado dos tópicos do curso e por fim, foi solicitado um *feedback* específico sobre os cenários e a forma como foram integrados ao curso. As opiniões dos alunos foram positivas sobre a inclusão do RP no currículo, proporcionando um sentido de autenticidade aos exercícios e os alunos encorajaram a inclusão destes exercícios em iterações futuras do curso. Verificou-se que a experiência com RP em sala de aula melhorou as capacidades de comunicação e interação dos alunos.

Maxim *et al.* (2017) abordaram as lacunas existentes nas habilidades de ES dos alunos Universidade de Michigan-Dearborn e de como essas lacunas impactam no início dos seus principais projetos. Diante dessa motivação, os autores discutiram o uso de atividades de RP em equipe para simular a experiência de trabalho em um estúdio profissional de desenvolvimento de jogos como um meio de aprimorar um curso de *design* de jogos de graduação avançada. Em conjunto com o RP, uma estrutura de gamificação foi criada e utilizada dentro do curso para permitir que os discentes personalizassem sua participação no curso. Para a execução do projeto, os alunos se reuniram um dia por semana durante 3 horas e por um período de 14 semanas. Em seguida, os estudantes executaram a primeira tarefa, que consistia em criar um documento de *design* do jogo e um plano de negócios para o jogo. Já a segunda entrega foi um protótipo alfa do jogo que incluía um caminho lógico completo, um rascunho do documento do usuário e um instalador funcional. A terceira entrega foi um protótipo beta que precisava acomodar uma mudança de requisito e produto final foi o protótipo de lançamento e uma apresentação de marketing que incluiu um vídeo para promover o produto do jogo. A avaliação do curso estava em andamento e posteriormente os alunos iriam preencher um formulário de avaliação.

Com o propósito de possibilitar uma perspectiva geral e sintetizada dos trabalhos relacionados que foram previamente discutidos, apresenta-se no Quadro 1 uma sucinta comparação entre os mesmos. Em relação ao método de pesquisa, pode-se perceber que a abordagem mais utilizada foi relato de experiência (com 3 trabalhos). Em termos cronológicos, tem-se um

horizonte de publicações cobrindo 2003, 2006, 2012, 2014, 2016 e 2017. A principal disciplina avaliada nos trabalhos foi ES, porém, dois trabalhos abordaram o contexto de *design* de jogos. Os estudos foram aplicados em diferentes países como, Estados Unidos da América, Japão, Espanha e Inglaterra. O uso de questionários se revelou como a abordagem predominante, estando presente em todos os estudos com exceção de Dixon e Jagodzinski (2003), o qual utilizou observação para coleta de dados.

Tabela 1 – Síntese comparativa dos trabalhos relacionados.

Trabalhos relacionados	**Método de pesquisa**	**Disciplina avaliada**	**País de realização do estudo**	**Tipo de pesquisa**	**Método para coleta de dados**
Dixon e Jagodzinski (2003)	Etnografia	Engenharia de Software	Inglaterra	Qualitativa	Observação
Henry e LaFrance (2006)	Relato de experiência	Engenharia de Software	EUA	Quali-quantitativa	Questionário
Nakamura *et al.* (2012)	Relato de experiência	Gerenciamento de Projetos	Japão	Quali-quantitativa	Questionário
Redondo *et al.* (2014)	Estudo de caso	Engenharia de Software	Espanha	Quantitativa	Questionário
Decker e Simkins (2016)	Estudo de caso	*Design* de jogos	EUA	Qualitativa	Questionário
Maxim *et al.* (2017)	Relato de experiência	*Design* de jogos	EUA	Quali-quantitativa	Questionário

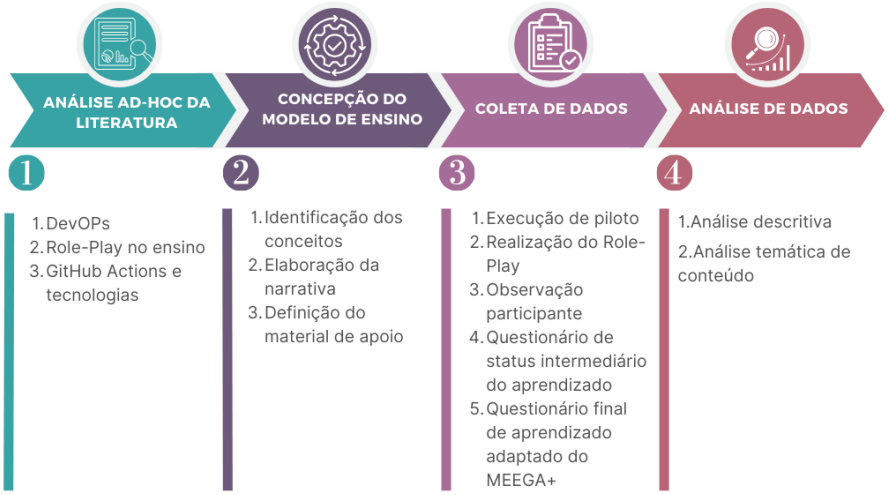
Fonte: Autoria própria (2023).

Conforme pode-se observar através das discussões anteriores, apesar de já existirem trabalhos envolvendo uso do RP para ensino de ES, nota-se que não há trabalhos que fazem o uso do RP para ensino de DevOps, apesar do uso extensivo do DevOps mercado de desenvolvimento de software. Refletindo sobre a relevância em averiguar tal lacuna de pesquisa, o presente trabalho busca investigar o uso do RP como prática pedagógica para fornecer experiencias próximas às demandas do mercado de software em relação ao contexto de DevOps.

4 PROCEDIMENTOS METODOLÓGICOS

A trajetória metodológica explorada neste trabalho assume um caminho quali-quantitativo, dispondo-se a entender as experiências e percepções de estudantes de graduação sobre a adoção de *Role-Play* para o aprendizado de DevOps. Quanto aos procedimentos, estabeleceu-se um perspectiva empírica baseada em pesquisa experimental. Ademais, este estudo se caracteriza como uma pesquisa descritiva, tendo a coleta de dados pautada pelo uso de questionários estruturados e observação-participante. Em relação à análise de dados, tem-se estabelecido o uso de estatística descritiva e Análise Temática de Conteúdo (CRUZES; DYBA, 2011). Aderente às convenções de condensação de dados, exibição de dados e desenho/verificação de conclusões propostas por Miles *et al.* (1994), a presente pesquisa assume um escopo multimétodo organizado por quatro macro-estágios (ver Figura 6): 1) análise *ad hoc* da literatura, 2) concepção do modelo de ensino, 3) coleta de dados e 4) análise de dados.

Figura 6 – Procedimentos metodológicos



Fonte: Autoria própria

Inicialmente, cumpriu-se uma **análise *ad hoc* da literatura** objetivando alcançar um entendimento abrangente acerca dos componentes teóricos e práticos relevantes à proposta. Para esse processo de análise bibliográfica, realizou-se, entre os meses de Setembro de 2021 a Março de 2022, uma busca manual no *Google* e *Google Scholar* utilizando combinações de palavras-chave relacionadas à pesquisa (*Role-play*, DevOps, Integração Contínua, etc). Dessa forma, abordou-se majoritariamente artigos aprovados em conferências/periódicos científicos e trabalhos de conclusão de curso, escritos em português ou inglês. Em especial, neste momento, identificou-se dois trabalhos fundamentais para esta pesquisa. O primeiro foi o *survey* elaborado

por Leite *et al.* (2019), o qual mapeou uma lista de conceitos estruturantes de DevOps. O segundo trabalho foi a pesquisa de Cherif e Somervill (1995) a qual apresentou uma série de diretrizes para condução de RP no contexto de ensino.

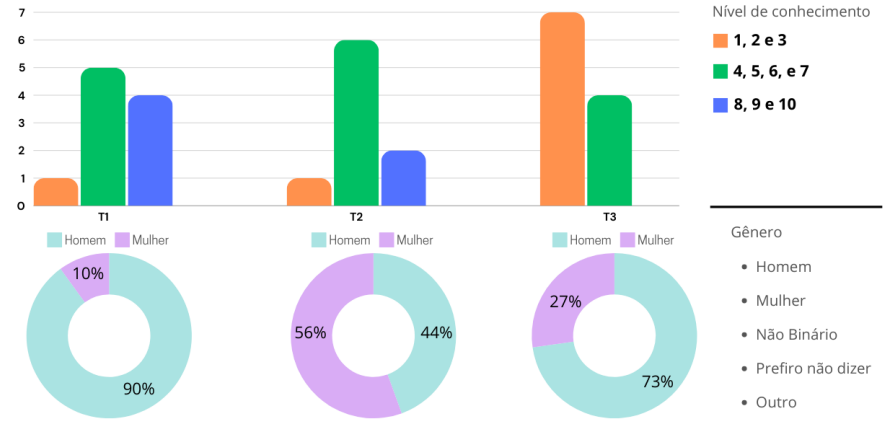
Após assimilação dos conceitos fundamentais demandados para este trabalho, tornou-se possível iniciar o macro-estágio referente à **concepção do modelo de ensino** baseado em RP. Tal modelo foi inspirado nas diretrizes definidas por Cherif e Somervill (1995). Em relação ao conteúdo didático, optou-se por respaldar a proposta a partir do *survey* elaborado por Leite *et al.* (2019), auxiliando, assim, na identificação das categorias, ferramentas e conceitos relevantes em DevOps. Com as informações extraídas e adaptadas para este trabalho, elaborou-se uma narrativa composta por três fases cobrindo o possível *onboarding* de desenvolvedores de software júnior em um uma empresa fictícia de pequeno porte com foco em desenvolvimento de software.

No que se refere à **coleta de dados**, esta foi iniciada *durante* e *após* realização do RP, onde os participantes (discentes representando desenvolvedores de softwares), orientados pelo condutor (primeiro autor deste artigo representando o *Tech Lead* na empresa) engajaram no roteiro estabelecido no modelo de ensino. Nesse caso, o modelo de ensino foi investigado a partir de um experimento educacional composto por três turmas de participantes de semestres distintos dos cursos de Ciência da Computação e Sistemas de Informação (T1, T2, T3) com respectivamente 10, 9 e 11 estudantes em cada sessão, totalizando 30 participantes. Antes de conduzir o experimento com as três turmas, entretanto, realizou-se dois execuções-piloto para validação dos procedimentos adotados, incluindo instrumentos para coleta de dados, dinâmica do RP, etc.

Após a devida validação dos procedimentos, ocorreram as sessões de avaliação do RP com as três turmas de forma separada em um laboratório da Universidade Federal do Ceará (UFC) - *Campus* Crateús, respectivamente nos dias 01/11/23, 16/11/2023 e 29/11/2023. Durante as sessões, os estudantes foram organizados em duplas. Na Figura 7 tem-se uma visão geral sobre a caracterização dos participantes por turma (T1, T2 e T3) no que se refere ao nível auto-avaliado de conhecimento sobre DevOps e distribuição por gênero. Conforme pode-se perceber, a T1 apresentou o conjunto de discentes com maior percepção de conhecimento sobre DevOps: 90% declararam ter nível 7, 8 ou 9. Tal turma também teve maior quantidade de homens (90%). A T2, por sua vez, teve um cenário mais equilibrado quanto ao nível de conhecimento e gênero (com uma quantidade maior de mulheres: 56%). Por fim, a T3 foi a turma com mais pessoas que declararam nível 1 de conhecimento (45,45%.) Em termos de gênero, verificou-se, novamente,

uma maior quantidade de homens (73%). Para mais detalhes sobre os participantes, verificar os dados granulares disponíveis no Apêndice B.

Figura 7 – Caracterização dos participantes quanto nível de conhecimento sobre DevOps e gênero.



Fonte: Autoria própria

Sobre a coleta de dados, usufruiu-se de duas técnicas distintas: questionários estruturados e observação-participante. Em relação aos questionários estruturados, a coleta foi realizada em dois momentos distintos (com dois questionários distintos). Primeiramente, de forma periódica, após a conclusão de cada fase do RP, ou seja como um status intermediário cobrindo a experiência do participante (ao invés de somente no final). Para tais momentos, aplicou-se o primeiro questionário, composto por duas perguntas: 1) Com base no seu entendimento dos conceitos abordados nesta fase, por favor, classifique seu nível de compreensão de 1 a 5, sendo 1 o nível mais baixo e 5 o nível mais alto; 2) Com base na sua experiência durante esta fase, existe algum comentário aberto que você deseja fazer? Dessa forma, a partir das respostas obtidas, torna-se possível futuramente realizar ajustes pontuais em cada fase do modelo.

O segundo momento de coleta de dados ocorreu com base em um segundo questionário o qual foi aplicado após a conclusão de toda a experiência, ou seja, após o estudante ter percorrido todas as fases do modelo de ensino. Nesse caso, aplicou-se um questionário inspirado no modelo MEEGA+ (WANGENHEIM *et al.*, 2018), o qual, por sua vez, já tem sido explorado no contexto de educação em Engenharia de Software (SILVA *et al.*, 2021; FERREIRA *et al.*, 2021). De acordo com Petri *et al.* (2017), o MEEGA+ possibilita avaliar jogos educacionais digitais e não digitais para ensino de Computação, considerando a percepção da qualidade em termos de experiência do jogador e percepção da aprendizagem do ponto de vista de estudantes e instrutores. O foco principal das alterações do MEEGA+ foram as nomenclaturas (de cu-

nho generalistas) das perguntas feitas aos alunos, para enunciados focados na experiência do participante dentro do RP.

Outra maneira utilizada para a coleta de dados foi por meio da observação participante. Tal coleta ocorreu durante o cenário do *onboarding* dos desenvolvedores de software júnior da empresa fictícia, que foi planejado para o RP. Sendo assim, as interações foram delineadas para ocorrerem entre duplas de alunos ou entre as demais duplas, possibilitando que os participantes verbalizassem seus conhecimentos prévios ou adquiridos durante o RP sempre que necessário. Para garantir que as interações ocorressem da melhor maneira possível, o aplicador, na figura de *Tech Lead*, mediou algumas interações ao tirar dúvidas pontuais dos alunos e fornecer conhecimento primordial para a execução do RP. Essas interações resultaram em um material que foi colhido após a conclusão de cada seção, o qual foi composto por *feedbacks* que variaram da expressões verbais e sociais dos participantes.

Por fim, no que diz respeito à **análise de dados**, averiguou-se que as deliberações a respeito dos questionários em complemento aos dados obtidos via observação-participante constituem uma multiplicidade rica de discussões. Para a análise dos resultados quantitativos desfrutou-se de estatística descritiva, enquanto os dados qualitativos foram expostos ao método Análise Temática de Conteúdo (CRUZES; DYBA, 2011; BARDIN, 1979), cujo proposito consiste em evidenciar os itens de significação a partir da descrição do *corpus* obtido. Motivado por Braun e Clarke (2006) e Holton (2007), estabeleceu-se o procedimento de análise qualitativa em quatro passos comuns. No primeiro passo, buscou-se a assimilação com os dados no decorrer da transcrição das entrevistas realizadas e a leitura e releitura das anotações. Posteriormente, progrediu-se com a *codificação aberta* que teve como proposito de estabelecer falas e observações que sejam correlatos, constituindo-os de acordo com o tema abordado. Em seguida, no terceiro passo, houve a *codificação axial* objetivando anexar as falas e os temas emergidos preliminarmente com a finalidade de identificar subtemas e criar categorias que contenham similaridade para, assim, mesclar os resultados. Tal processo foi preliminarmente desempenhado de forma dinâmica e indutiva pelo autor e, em seguida, aprimorado pelos orientadores (viabilizando a comparação interpretativa). Por fim, se realizou uma avaliação geral nas informações analisadas para materializar as informações (cujo *codebook* encontra-se disponível no Apêndice C) de modo que seja possível compreender as contribuições deste trabalho.

5 D&O: PROPOSTA DE MODELO DE ENSINO

O modelo de ensino de DevOps baseado em *Role-Play* (RP) proposto neste trabalho, denominado *Dev & Ops* - **D&O** (título inspirado no famoso RPG *Dungeons and Dragons* - D&D), foi inicialmente influenciado pelas diretrizes definidas por Cherif e Somervill (1995). De acordo com tal pesquisa, existem cinco elementos necessários para condução do RP: 1) a formulação do problema ou tópico a ser abordado; 2) os personagens ou papéis que irão ser desempenhados; 3) o contexto e as informações que os alunos devem conhecer ou pesquisar; 4) as funções ou etapas que cada estudante ou grupo de estudantes deve cumprir e 5) os procedimentos para a apresentação da dramatização da peça. Nesse contexto, o D&O ainda se respalda na Taxonomia de Bloom (BLOOM *et al.*, 1984) no que se refere aos objetivos educacionais, sendo eles divididos em três domínios: cognitivo, afetivo e psicomotor. À luz desses direcionamentos pedagógicos, elucida-se a seguir o funcionamento do D&O.

Quanto à **formulação do problema**, tem-se aqui definido a necessidade de prover aos alunos condições de aprender, em uma perspectiva teórico-prática, os conceitos básicos explorados no contexto de DevOps. Visando definir quais conceitos seriam explorados (seja de forma conceitual ou prática), este trabalho adotou o *survey* mapeado por Leite *et al.* (2019) como base estruturante para o que será explorado no RP, conforme ilustrado na Tabela 2. Tal tabela traz uma lista de conceitos abordados em DevOps aderentes à diferentes categorias. Conforme pode-se perceber, existem três colunas na direita cobrindo se o referido conceito foi mencionado, explicado ou praticado durante a dramatização idealizada neste trabalho. Por restrições de tempo e visando não ficar demasiadamente cansativo/complexo, alguns conceitos foram mencionados de forma superficial, enquanto outros foram explicados com maior detalhamento teórico e, subsequentemente, praticados. Entretanto, conforme pode-se observar na tabela, buscou-se cobrir, em termos de práticas, a maioria de conceitos por categoria.

De modo mais específico e convergente com os conceitos expostos na Tabela 2, estabeleceu-se uma estrutura narrativa baseada no *onboard* de engenheiros de software júnior recém-contratos em uma empresa de desenvolvimento de software de pequeno porte que adota práticas de *Scrum* e DevOps. Esse time será recebido por um *Tech Lead* o qual explicará o *pipeline* de desenvolvimento adotado na organização e a necessidade requisitada pelo *Product Owner* (PO). Tal demanda reside na construção de uma *Application Programming Interface* (API) baseada em *NodeJS* que contemple três Requisitos Funcionais (RFs): RF1) criar um usuário, RF2) consultar uma lista de usuários e RF3) consultar um usuário específico.

Tabela 2 – Conceitos de DevOps abordados na presente proposta

Categoria	Conceito	Mencionado	Explicado	Praticado
Conhecimento compartilhado	Cultura de colaboração		✓	✓
	Compartilhando conhecimento		✓	✓
	Quebrando silos	✓		
	Colaboração entre departamentos	✓		
Gerenciamento de código-fonte	Controle de versão		✓	✓
	Cultura de colaboração		✓	✓
	Compartilhando conhecimento		✓	✓
	Quebrando de silos	✓		
	Colaboração entre departamentos	✓		
Processo de construção	Engenharia de lançamento		✓	✓
	Entrega contínua	✓		
	Automação	✓		
	Automação de teste		✓	✓
	Análise estática		✓	✓
Integração contínua	Processo de lançamento frequente e confiável		✓	✓
	Engenharia de lançamento	✓		
	Integração contínua		✓	✓
	Pipeline de implantação		✓	✓
	Entrega contínua, automação	✓		
	Gerenciamento de artefatos		✓	✓
Automação de implantação	Processo de lançamento frequente e confiável	✓		
	Engenharia de lançamento	✓		
	Gerenciamento de configurações		✓	✓
	Implantação contínua		✓	✓
	Infraestrutura como código	✓		
	Virtualização, conteinerização	✓		
	Serviços em nuvem, automação	✓		
Monitoramento e registro	Você construiu, você executa		✓	✓
	Suporte fora do expediente para Devs	✓		
	Monitoramento contínuo do tempo de execução	✓		
	Desempenho, disponibilidade, escalabilidade	✓		
	Resiliência, confiabilidade, automação	✓		
	Métricas, alertas, experimentos		✓	✓
	Gerenciamento de logs, segurança		✓	✓

Fonte: Tabela adaptada de (LEITE *et al.*, 2019)

De forma alinhada ao que foi explanado anteriormente, em relação aos **personagens ou papéis que irão ser desempenhados na dinâmica**, estabeleceu-se três opções possíveis se inspirando nas orientações do *Scrum*. A primeira, denominada *Tech Lead*, tem o propósito de conduzir toda a dinâmica, tanto sob uma perspectiva pedagógica, quanto em uma orientação técnica. Orienta-se que tal papel seja exercido pelo docente ou monitor da disciplina. Por sua vez, o segundo papel será o de *Product Owner*, o qual não será interpretado por nenhum participante da dinâmica, mas representará um *Non-Player Character* (NPC) para ilustrar a influência do PO em num contexto de um projeto de software. Finalmente, tem-se o *Dev Team*, a ser interpretado por um time de discentes, que exercitará os conceitos e práticas orientados pelo *Tech Lead*.

Quanto ao **contexto e as informações que os alunos devem conhecer ou pesquisar**, tem-se como premissa que os alunos já tenham conhecimento de programação *web* e ES, sendo esses concluintes da disciplina ou já aprovados, por exemplo. Com isso posto, espera-se

que os discentes disponham de uma base prévia sobre os pilares do processo de desenvolvimento de software de uma forma geral, em especial, conhecimentos dos conceitos da construção de uma API, conhecimentos no padrão *Representational State Transfer* (REST), conhecimento sobre gerenciamento de pacotes *Node Package Manager* (NPM), conhecimento em *Git* e *GitHub*. Entretanto, a dinâmica aqui desenhada se propõe a aprofundar especificamente os conceitos de DevOps. Logo, os alunos podem conhecer previamente e brevemente os elementos básicos da referida área, mas não é obrigatório.

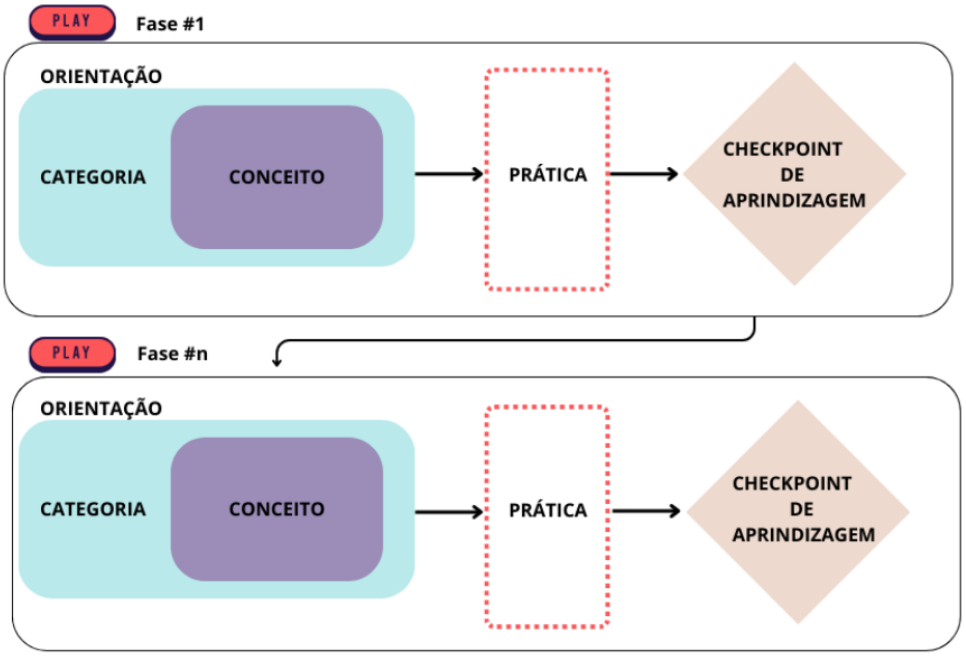
Sobre **as funções que os estudantes devem cumprir**, tem-se a premissa que o *Dev Team* (formado pelos discentes) vai ser responsável por participar do *onboard* na empresa e, consequentemente construir a demanda requisitada pelo PO de forma aderente ao *pipeline* de desenvolvimento adotado na *startup* a qual eles foram recém-contratados(as). Nesse caso, no contexto de uma disciplina com muitos alunos sugere-se que eles sejam organizados em duplas por computador, assemelhando-se à prática de programação em pares.

Finalmente, detalha-se a seguir os **procedimentos para a apresentação da dramatização da peça**. Conforme ilustrado na Figura 8, o roteiro sugerido e estabelecido para a dramatização (com previsão de duração de até 100 minutos) tem três fases distintas visando ser simples de replicar de um ponto de vista pedagógico, mas robusto o suficiente para cobrir os principais conceitos de DevOps elencados por Leite *et al.* (2019). Entretanto, há de se destacar que, em se tratando de uma proposta baseada em RP, há margem para possíveis adaptações e ajustes, seja previamente em termos de planejamento por parte do docente ou mesmo de forma improvisada durante a dinâmica conforme a dramatização avançar. Logo, as fases do roteiro servem apenas como uma base instrumental para guiar a dinâmica de forma técnica e narrativa. Além do roteiro, o presente trabalho também disponibiliza um uma apresentação (disponível no repositório de apoio (MOTA, 2023) em formato de *slides* para guiar o *Tech Lead*.

Conforme ilustrado na Figura 8, inspirando-se em *microlearning* (GABRIELLI *et al.*, 2017), cada fase do roteiro é composta por três momentos distintos e busca ser objetivamente conciso para amenizar a fadiga dos alunos. Inicialmente, tem-se a **Orientação de Conceitos** (por parte do *Tech Lead*) o qual visa prover ao *Dev Team* um entendimento claro sobre o que será explorado na fase em questão de forma análoga ao *onboard* de funcionários em um ambiente real de desenvolvimento de software. O segundo momento, denominado **Prática dos Conceitos**, visa promover ao *Dev Team* condições de exercitar na prática o que foi explanado anteriormente. Tal exercício deve auxiliar os discentes na resolução técnica da demanda do PO, bem como

explorar o aprendizado de *soft skills*. O terceiro e último momento reside num ***Checkpoint* de Aprendizado** realizado com os alunos para avaliar o aprendizado da fase como um todo. Tal *checkpoint* se dá através da verificação, junto aos discentes, do cumprimento de determinados objetivos da fase. Pretendendo que toda a dinâmica transcorra dentro do tempo previsto e atingindo os objetivos de cada fase, serão permitidas intervenções no momento da dinâmica. As intervenções podem ser feitas dentro do mesmo time, onde os participantes que possuam mais experiência auxiliem de imediato os demais companheiros. Caso ainda haja dificuldades nas conclusões, as demais duplas, que por ventura estejam conseguindo progredir com as demandas, poderão prestar apoio aos demais participantes das duplas diferentes, com o proposito de manter a rigidez do tempo estabelecido. Caso não haja uma evolução da resolução dos objetivos dentro das equipes e nem entre elas, o *Tech Lead* deve de imediato intervir, mostrando para todos a resolução das atividades propostas.

Figura 8 – Visão geral do Modelo de Ensino D&O.



Fonte: Autoria própria.

Em conformidade com o modelo apresentado na Figura 8, detalha-se na Tabela 3 como foi conduzido a dinâmica no caso da presente pesquisa a qual, inclusive, pode ser adotada por outros praticantes. Uma descrição textual detalhada sobre a Tabela 3 encontra-se disponível no Apêndice A.

Tabela 3 – Síntese da narrativa estabelecida a partir do modelo D&O

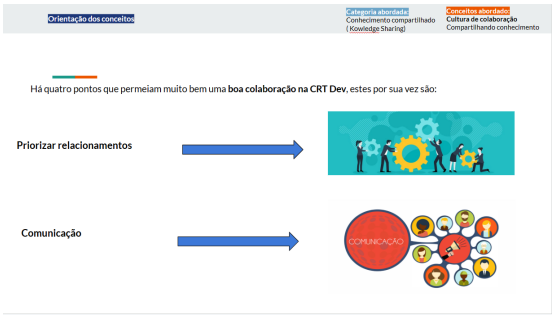
Fases	**Objetivo do aprendizado**	**Orientação dos Conceitos**	**Prática de Conceitos**	***Checkpoint* de aprendizado**
Fase #1	Aprender sobre Compartilhamento de conhecimento & Gerenciamento de código-fonte (GCF)	A orientação dos conceitos se caracteriza pela explicação dos conceitos de Cultura de colaboração, Compartilhando conhecimento e GCF, onde GCF tem como premissa a explicação do Controle de Versão	1)A prática terá início com a divisão dos alunos em grupos onde haverá uma apresentação dos mesmo. 2) Após a apresentação ocorrerá um stand-up meetings para debater sobre as demandas do PO. 3) Os grupos devem usar o conceito de programação em pares visando obter os conhecimentos explícitos e tácitos. 4) Utilizando dos conceitos de Cultura de colaboração e compartilhamento de conhecimento os mesmos devem efetuar o fork do projeto. 5) Instalar os pacotes Nodemon e Express. 6) Concluir as demandas solicitadas pelo PO. 7) Ao concluir cada demanda deve-se seguir o padrão de Versionamento Semântico e a Convenção de Commits.	1) Conhecimento sobre os pontos informados pelo PO. 2) Apresentação dos integrantes das equipes. 3) Programação em pares dentro das 4) Compartilhamento de conhecimento entre os pares 5) Ter a versão do arquivo package.joson atualizada com a convenção SemVer. 6) ter efetuado os Commits com o padrão adequado da convenção.
Fase #2	Aprender sobre de Processo de construção (PC), Integração Contínua (CI) e Automação de implantação	A orientação dos conceitos da fase atual agrupa as explicações sobre três conceitos: Engenharia de lançamento(EL) Entrega Contínua (CD) Automação Automação de testes(AT) Análise estática(AEs) Processo de lançamento frequente e confiável Engenharia de lançamento Integração contínua Pipeline de implantação Gerenciamento de artefatos Gerenciamento de configurações Implantação contínua Infraestrutura como código Virtualização, Conteinerização Serviços em nuvem, Automação	1) Instalação dos pacotes: Mocha, Prettier e ESLint para a execução dos testes e análise estática de código. 2) Criação do arquivo YAML. 3) Explicação do arquivo YAM e alterações finais. 4) Instalar o Fly.io. 5) Criar a secret key. 6) Inclusão do trecho de código no arquivo YAML para a execução do deploy automático da aplicação.	1) Instalação de todos os pacotes informados para construção dos casos de testes. 2) Construção dos arquivo dos testes automatizados e ter obtido sucesso em todos os casos. 3) Instalação e configuração o Prettier e ESLint 4) Ter construindo o arquivo que irá possibilitar a construção do pipeline de CI 5) Obtenção de 100% de sucesso 6) Obtenção de êxito em todos os passos para instalaro Fly.io 7) Adicionar os scripts necessários no nosso arquivo yml para as devidas configurações do CD. 8) Efetuar mais um commit para acionar o pipeline de CI/CD e ver a implantação da API no Fly.io
Fase #3	Aprender sobre Monitoramento e Registro	Em sua orientação será tratado: Você construiu, você executa Suporte fora do expediente para Devs Monitoramento Contínuo Desempenho, Disponibilidade, Escalabilidade Resiliência, Confiabilidade, Automação Métricas e Logse sua definição e sua definição	1) A prática tera inicio ao baixar o Sonarqube e configurá-lo bem como o SonarScanner 2)Para a prática de Logs deve-se instalar os pacotes body-parser e morgan-body. 3) Criar uma pasta na raiz do projeto com o nome de "logs”. 4) No arquivo “index.js” deve ser acrescentado um conjunto de constantes	1)Entendimento dos conceitos de Você construiu, você executa 2)Entendimento dos conceitos de Monitoramento contínuo 3) Entendimento dos conceitos de Métricas e ter concluído sua respectiva prática. 4)Correta implantação dos pacotes e obtenção do arquivo de Log.

Fonte: Autoria própria (2023).

Visando apoiar o processo de RP, este trabalho também disponibiliza uma apresentação sob a forma de slides para guiar o *Tech Lead* (simulando o *onboard* de desenvolvedores em uma *starup* fictícia de desenvolvimento de software) e um repositório no *GitHub* com a parte técnica explorada na proposta. Todos esses materiais e *links* se encontram disponíveis no repositório de apoio (MOTA, 2023). A Figura 9 ilustra os exemplos de slides utilizados na apresentação. Conforme pode-se perceber, os *slides* foram elaborados em uma cronologia que contempla a apresentação e, para cada fase, a **Orientação dos Conceitos**, a **Prática dos Conceitos** Figura 9 e, por fim, tem-se o ***Checkpoint* de Aprendizado** contemplando o objetivo de aprendizado da fase (e o *link* com o formulário de avaliação intermediária sobre aprendizado).

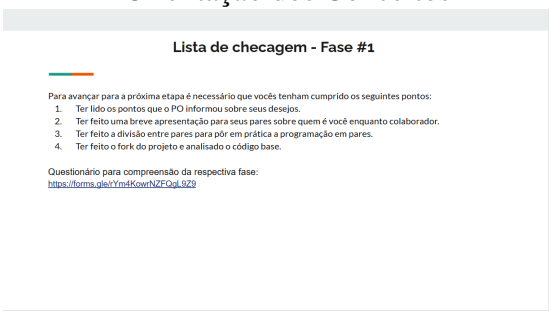
Figura 9 – Exemplos ilustrativos dos slides de *onboarding* utilizados durante o RP.

Apresentação



Orientação dos Conceitos

Práticas dos Conceitos



Checkpoint de Aprendizado

Fonte: Autoria própria.

6 RESULTADOS

A seguir é descrito e analisado os resultados deste trabalho. Inicialmente são apresentados os resultados do piloto da dinâmica de *Role-Play* (RP) e, posteriormente, uma seção dedicada para análise das fases do RP, uma seção para análise do MEEGA+ e, finalmente, uma seção para análise de *feedbacks* gerais sobre a experiência como um todo.

6.1 Lições aprendidas na aplicação de execuções-piloto

Antes de iniciar a dinâmica com as três turmas, optou-se por utilizar a estratégia de execuções-piloto, tendo como propósito validar os procedimentos adotados, por exemplo, a duração da prática, instrumentos para coleta de dados, obter *feedbacks* e eventuais necessidades de mudanças nos slides de apoio. Inicialmente através do primeiro o piloto, o qual ocorreu no dia 13/10/2023, pode-se constatar que o tempo para a execução da prática estava demasiadamente longo, devido a configurações das máquinas utilizadas e a quantidade demasiada de falas por parte do papel do *tech lead* (ao todo o piloto durou aproximadamente 120 minutos). Tal questão acabou impactando, por exemplo, no tempo de preenchimento dos questionários. Sendo assim, houve a necessidade de ajustes nos slides utilizados para conduzir a prática e uma dedicação na configuração dos computadores que foram empregados no RP. Após as devidas alterações realizou-se um segundo piloto no dia 17/10/2023, com uma dupla distinta. A segunda aplicação do piloto possibilitou que os objetivos fossem atingidos, garantindo uma execução dentro do tempo estipulado (100 minutos), o que possibilitou o preenchimento dos formulários para coletas de informações. Além disso, verificou-se, com base nos *feedbacks* dos estudantes, que houve uma assimilação suficiente sobre a execução do RP.

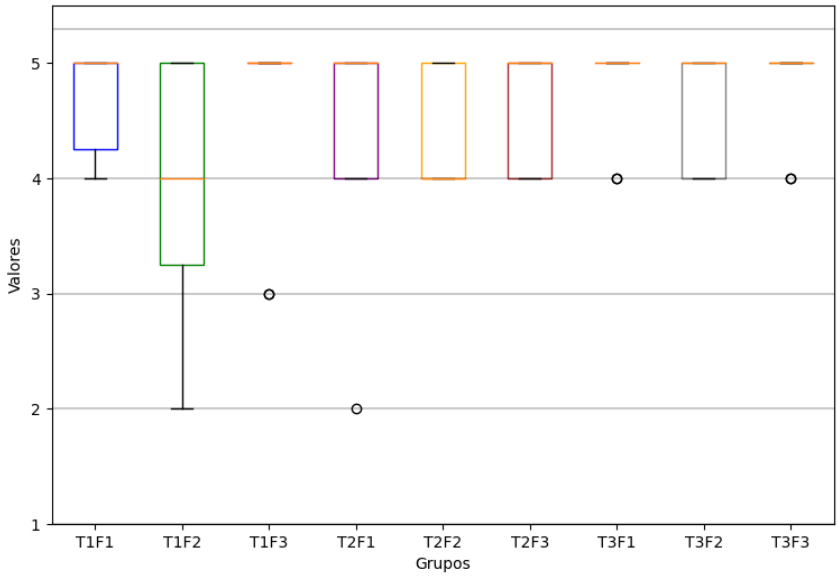
6.2 Análise das Fases

Na presente seção são apresentados os resultados relativos à análise do questionários aplicados entre as fases. Essa análise foi organizada a partir das duas perguntas do referido questionário: 1) Com base no seu entendimento dos conceitos de DevOps abordados nesta fase da dinâmica de RP, por favor, classifique seu nível de compreensão de 1 a 5, sendo 1 o nível mais baixo e 5 o nível mais alto e 2) Com base na sua experiência durante a fase, existe algum comentário aberto que você deseja fazer?

6.2.1 Avaliação quantitativa do nível de entendimento sobre os conceitos de cada fase

A Figura 10 apresenta, sob a forma de um gráfico *box plot*, os resultados obtidos por cada turma (T1, T2 e T3) para cada uma das três fases do RP (F1, F2 e F3). Por exemplo, T1F1 refere-se aos resultados obtidos pela turma T1 com base nos conceitos aprendidos na F1.

Figura 10 – Resultados sobre o nível de compreensão a cada fase por turma.



Fonte: Autoria própria.

Em relação à T1 (com 10 participantes) verificou-se que 70% dos estudantes alegaram um nível de entendimento entre 4 e 5 para a F1 foi possível observar que para essa fase a mediana foi igual a 5, o primeiro quartil foi 4 e o terceiro quartil foi 5. A porcentagem quanto ao nível de conhecimento permaneceu a mesmo em F2, desta forma a mediana desta fase foi 4, com o primeiro quartil em 3 e terceiro quartil em 5. Por sua vez, F3 demonstrou que 80% dos estudantes concordaram ter tido um alto nível (5) de entendimento, obtendo assim uma mediana em 5, assim como o primeiro quartil e terceiro quartil que também foram 5.

No que se refere à T2 (com 9 participantes) verificou-se que 88,8% dos estudantes argumentaram terem tido um nível de entendimento entre 4 e 5 para a F1 com uma mediana de 5, o primeiro quartil foi de 4 e o terceiro quartil foi 5. Para as fases F2 e F3, observou-se que 100% os alunos citaram terem tido um nível de entendimento entre 4 e 5. Desta forma, a mediana de F2 foi igual a 4, com seu primeiro quartil em 4 e o terceiro quartil em 5. Já F3 em seu primeiro quartil apresentou um valor igual a 4, a mediana teve o valor de 5 assim como o terceiro quartil foi de 5.

Quanto à T3 (com 11 participantes) observou-se que 100% dos estudantes revelaram

um nível de entendimento entre 4 e 5 para a F1. Sendo assim, a mediana apresentou valor igual a 5, o primeiro e segundo quartil tiveram o mesmo valor da mediana. Os valores referentes ao nível de conhecimento foram iguais tanto para F2 como Para F3. Desta maneira F2 teve uma mediana igual a 5, já o primeiro quartil foi igual a 4 e terceiro quartil foi igual a 5. Por conseguinte, F3 apresentou os valores da mediana em 5, assim como também o primeiro e terceiro quartil tiveram o mesmo valor da mediana.

Os resultados dos questionários aplicados no decorrer das três fases da dinâmica de RP puderam revelar progressos no que diz respeito ao entendimento dos conceitos de DevOps por parte dos alunos nas três fases da dinâmica. Desta forma, houve um crescimento geral na pontuação das turmas ao longo das três fases. A T1 mostrou uma elevação no seu nível de compreensão, uma vez que 80% indicaram o nível mais alto de conhecimento na ultima fase. Ao analisar os dados por meio do gráfico de *box plot*, foi possível visualizar de forma mais clara as tendencias ao longo do tempo, onde as turmas T2 e T3 demonstraram melhorias consistentes quanto ao entendimento, uma vez que a maioria dos alunos classificaram seu nível de compreensão entre 4 e 5 em todas as fases. Sendo assim, esses resultados demonstram eficácia da dinâmica ao longo do tempo corroborando o uso do RP para o ensino dos conceitos de DevOps.

6.2.2 Avaliação qualitativa da experiência durante cada fase

A presente seção tem como foco analisar qualitativamente a experiência dos estudantes a cada fase da dinâmica proposta a partir das respostas obtidas para a questão aberta do primeiro questionário. Com base em nas respostas obtidas, foram identificados temas e códigos, os quais encontram-se sintetizados no Quadro 1.

Quadro 1 – Síntese da Análise Temática de Conteúdo referente à experiência durante cada fase.

Tema	**Códigos**	**Participantes**
Experiências vivenciadas em cada fase	Aprendizagem Cooperativa	P8,P10,P18,P22,P23,P26,P27,P29
	Aprendizagem com Ferramentas da Indústria	P4,P5,P8,P19,P12,P28,P29
	Desafios e Pontos Positivos	P4,P15,P14,P23,P27,P20,P22

Fonte: Autoria própria (2023).

Em relação ao tema, relacionado às **Experiências vivenciadas em cada fase**, emergiu-se três códigos diferentes. O primeiro código, denominado Aprendizagem Cooperativa, refere-se ao feedback dos participantes sobre a oportunidade de explorar práticas cooperativas como

mecanismo didático, vide o uso de programação em pares. Os participantes discorreram sobre suas impressões no que concerne aos conteúdos teóricos e práticos do RP. Sendo assim, na F1, P8 explanou que: "*A dinâmica de programação em pares foi muito produtiva pois mostrou a importância da cooperação e comunicação entre os pares tanto do time, como também, de toda a empresa*"; já P10 relatou que: "*Foi legal saber que o DevOps inicia com algo mais colaborativo*; durante a ocorrência da F2, P18 discorreu que: "*Foi interessante ver que antes mesmo de se preocupar com código houve um momento para mostrar como trabalhar em equipe e compartilhar o conhecimento*"; já P23 relatou que foi "*Interessante o assunto abordado tendo em vista que não é muito debatido dentro da universidade, além disso, a metodologia auxilia no aprendizado de uma forma clara e coesa*".

Um dos objetivos da dinâmica foi justamente fornecer aos participantes uma imersão de conceitos e práticas que se assemelham com aquilo que a indústria espera, mas sem esquecer da importância de orientações teóricas. Por sua vez o código Aprendizado com Ferramentas da Indústria pode mostrar a receptividade dos alunos quanto ao uso de práticas que são abordadas na indústria e as ferramentas atreladas a elas. Desta forma, sobre a F1, P4 destacou que: "*Foi interessante ver na prática o processo que se aplica em grandes empresas de maneira mais resumida e rápida, assim, fortalecendo ainda mais os conceitos.*"; para P5: "*Foi interessante colocar em prática os primeiros passos para subir um projeto*; P19 em seu relato destacou: "*Interessante a atenção dada ao padrão de nomenclatura relacionado ao Git. Pois, facilita a visualização do projeto, juntamente com o histórico.*"; P28 ao aprender sobre versionamento relatou que: "*Houve um bom aproveitamento quanto ao entendimento e prática sobre o versionamento, contribuindo para o conhecimento em DevOps*".

Durante a F2 houve uma maior percepção de quanto ao uso de ferramentas como *Git*, *Fly.io* e *pipelines* de CI/CD. Isso ficou claro para P8, onde esclareceu: "*Foi muito interessante subir a aplicação no Git e ver toda a integração funcionando conforme o esperado, dando erro quando os teste não passaram e sendo aprovado quando os teste foram aprovados com sucesso.* ; P12 ficou surpreso ao usar uma ferramenta de hospedagem, o mesmo disse que: "*Foi muito interessante, eu não conhecia o processo do Fly.io. Porém, com as instruções ficou claro*; já P19, em sua análise pode discorrer que: "*A engenharia de lançamento é algo muito importante dentro de um projeto de software. É mais organizado e suas atividades são específicas; ajudando também na liberação de mais versões para produção, para que atualizações não sejam perdidas no processo e facilitando a visualização dos códigos*.

Sobre a F3, P29 disse que: "*A criação e utilização de pipelines automatizados para compreender todo o fluxo de formatação,teste e deploy de nossas aplicações é algo muito benéfico que tende a ajudar o time de desenvolvimento como um todo, se bem utilizado*; P8 enfatizou sobre o uso das ferramentas de monitoramento, uma vez que: "*A utilização da análise estática com SonarQube foi impressionante, uma ferramenta indispensável para o bom desenvolvimento de software*; sobre a prática de *logs*, P19 refletiu: "*É algo muito importante entre a equipe de desenvolvimento de teste, e foi muito importante abordar. É útil em diversas situações, mas uma delas é quando o sistema dá algum problema ou o servidor cai, os logs ajudam a mapear de onde veio esse problema, com informações precisas, que ajudam também em testes, como data, hora, máquina, SO, etc*.

O código Desafios e Pontos Positivos refletiu um um panorama sobre os desafios e pontos positivos associados ao uso de RP. Durante a execução da F1, P15 ponderou que: "*As nomenclaturas iniciais foram um pouco complexas mas gostei bem da didática*; por sua vez, P23 relatou: "*Interessante o assunto abordado tendo em vista que não é muito debatido dentro da universidade, além disso, a metodologia auxilia no aprendizado de uma forma clara e coesa*; sobre a F2, P4 salientou que achou: "*Mais complexo, mas muito compreensivo do ponto de vista de todo o processo*; por sua vez, P14 enfatizou a existência de: "*Muitos detalhes que podem gerar erros facilmente se não seguir o passo a passo corretamente*; ao explanar seu posicionamento,o P27 disse: "*Gostei da teoria sobre o assunto junto com a aplicação prática dele pois ajuda a fixar melhor o entendimento*".

Quanto à F3, P4 pontuou que: "*Foi muito satisfatório e bem dinâmico, apesar de alguns erros, todas as fases até seu deploy foi muito bom o aprendizado!!*; de maneira similar, P20 em sua fala pediu: "*Por mais atividades nesse direcionamento*; semelhante a fala anterior, P22 e P27, disseram respectivamente o seguinte: "*Acredito que a prática foi relativamente simples por causa do conhecimento prévio, em geral, achei super bacana. Achei uma forma muito boa e dinâmica de compreender o conteúdo*".

De forma geral, através da análise dos dados previamente relatados, pode-se observar e emergência de códigos cobrindo aprendizagem cooperativa, aprendizagem com ferramentas da indústria e a identificação de desafios e pontos positivos. Em particular, ficou claro que os alunos valorizaram a oportunidade de explorar todas as práticas presentes no RP, incluindo a programação em pares as ferramentas necessárias para a operacionalização de DevOps.

6.3 Análise das categorias oriundas do Modelo MEEGA+

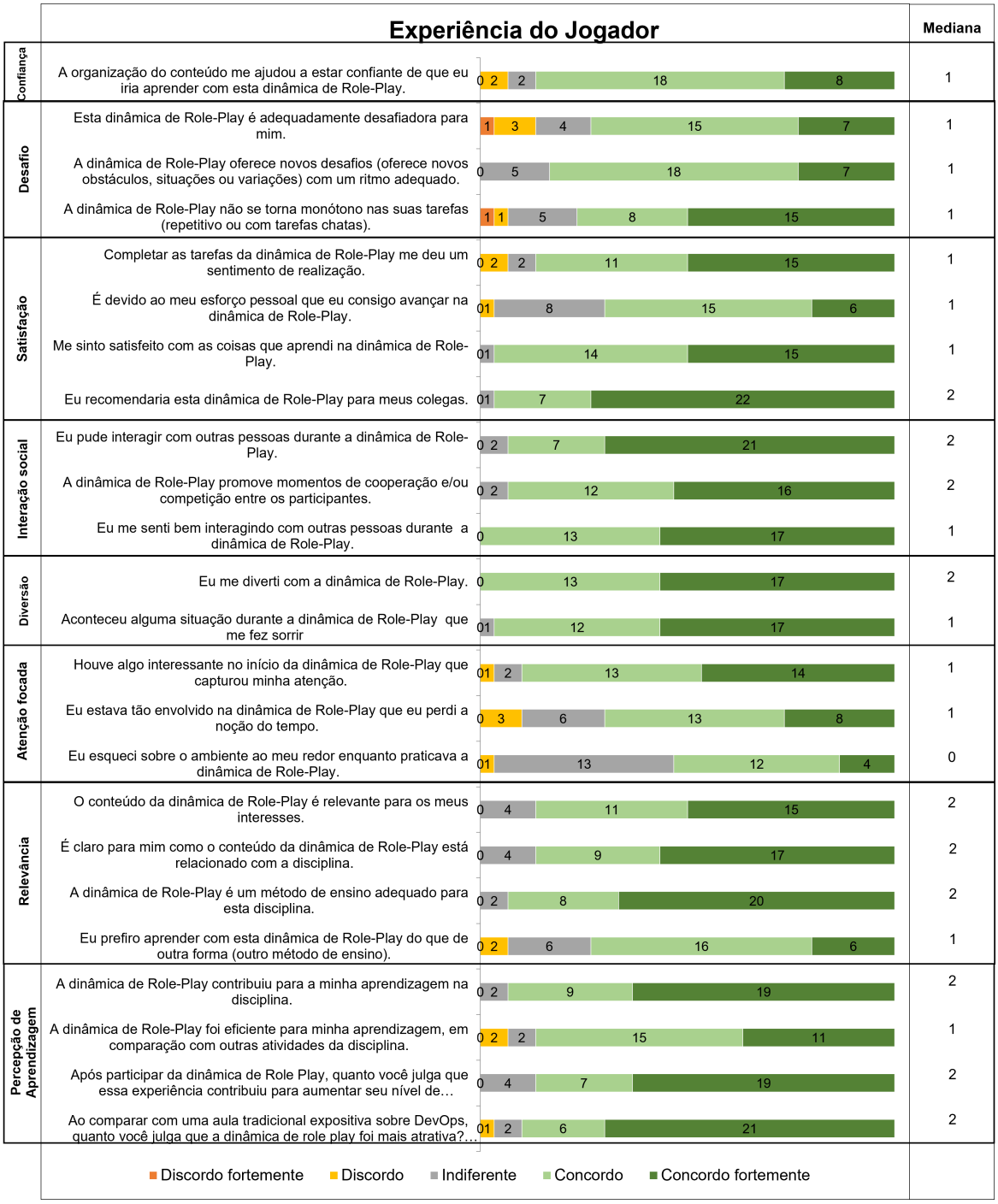
Conforme explicado na Seção 4, o presente trabalho explorou o modelo MEEGA+ para avaliação da experiência do jogador (PETRI *et al.*, 2019). O questionário proposto originalmente no MEEGA+ foca em duas perspectivas distintas, a usabilidade e a experiência do jogador. Nesse sentido, a presente avaliação optou por focar na experiência do jogador, tendo em vista que os elementos avaliados na usabilidade não seriam pertinentes para o objetivo deste estudo, bem como em relação a como o RP foi proposto. Sobre a experiência do jogador, os seguintes itens foram avaliados: Confiança, Desafio, Satisfação, Interação social, Diversão, Atenção Focada, Relevância e Percepção de Aprendizagem. Tais elementos serão analisados abaixo de forma separada, porém, na Figura 11 tem-se uma visão geral dos resultados obtidos.

No que concerne aos dados relativos a **Confiança**, aproximadamente 86,6% dos participantes concordaram (parcialmente ou fortemente) que a organização dos conteúdos os deixam mais confiantes em relação aos seus aprendizados. Assim, entende-se que a organização dos slides utilizados na dinâmica do RP se demonstrou bem aceita, uma vez que a disposição desses recursos ofereceu uma estrutura linear dentro de uma cronologia temporal, outro artefato que contribuiu para uma melhor confiança dos alunos foi a utilização de uma *wiki*, onde os mesmo poderiam consultar instruções importantes para o andamento da prática.

Os dados quanto ao **Desafio** revelaram que 73,3% dos alunos concordaram que a dinâmica apresentou um nível satisfatoriamente desafiador, enquanto 83,3% também expressaram concordar quanto à introdução de novos desafios em um ritmo adequado. Além disso, 76,6% dos estudantes, afirmaram concordar ou concordar fortemente que a dinâmica conseguiu evitar a monotonia. Essa categoria demonstrou resultados promissores, uma vez que os desafios foram elaborados para evitar dúvidas e hesitação durante as execuções. Houve uma combinação de programação, gerenciamento de pacotes, construção de *pipelines* de DevOps e monitoramento de aplicações. Para facilitar o acesso a todas as informações necessárias, foi desenvolvida e disponibilizada uma *wiki* abrangente para todos os participantes.

No que diz respeito à **Satisfação**, observou-se que 86,6% dos alunos concordaram ao afirmar que se sentiram satisfeitos ao concluir as tarefas da dinâmica. Outros 70% afirmaram concordar que, por meio de esforço próprio, conseguiram progredir nas atividades. Além disso, 96,6% dos participantes manifestaram concordar ou concordaram fortemente em relação à satisfação com o aprendizado adquirido. Por último, notou-se que 73% concordaram ou concordaram fortemente que recomendariam a dinâmica aos seus colegas. Esses resultados

Figura 11 – Resultados baseados no MEEGA+ com foco na experiência do jogador



Fonte: Autoria própria.

positivos são um reflexo do conteúdo planejado para o RP, sendo assim buscou-se uma construção que permitisse que todos os alunos conseguissem concluir a dinâmica sem um excesso de trabalho, a medida que os alunos foram estimulados a elaborar soluções para situações programadas, o que incluiu a resolução de pequenos erros.

Aos valores sobre a **Interação Social**, verificou-se que cerca de 70% dos participantes expressaram concordar em relação à viabilidade da interação durante a dinâmica.

Ademais, 93,3% dos participantes concordaram que a atividade promoveu situações de cooperação, enquanto 100% afirmaram sentir-se positivamente envolvidos ao interagir com os demais participantes. Os resultados dessa categoria foram animadores podendo serem atribuídos a progressão da programação em pares a medida que o RP e a sendo elaborado, além de incentivar a colaboração e o compartilhamento de conhecimento. Sendo assim, a contribuição dessa abordagem serviu para manter a interação constante entre todos os participantes do início até o fim da dinâmica.

Os valores pertinentes à **Diversão** revelaram que 100% dos estudantes concordaram que se divertiram durante a dinâmica de RP. Além disso, 96,6% deles expressaram concordância ao afirmar que certas situações da dinâmica provocaram risos. Referente a este item, não houve algo planejado dentro da elaboração da prática pensando em um alívio cômico propriamente dito, mas isso se deu de forma natural através da boa interação entre os participantes.

Quanto ao item **Atenção Focada** demonstraram que 90% dos membros alegaram concordar que houve elementos cativantes no início do RP que capturaram sua atenção. Além do mais, 70% deles concordaram estarem tão imersos no RP que perderam a noção do tempo, ao mesmo tempo que, 83,3% relataram concordar sobre o fato de terem esquecido o ambiente em que estavam imersos durante o RP. Alguns pontos que podem explicar esses valores podem ser esclarecidos pela quantidade de pessoas, uma vez que cada execução do RP aconteceu com uma média de 10 participantes e durante a execução havia distrações, conversas e intervenções nas práticas o que de certa forma não permite uma imersão de concentração focada, outro ponto pode ser o fato de alguns particionantes estarem fadigados dos afazeres do dia, uma vez que alguns alunos apresentaram cansaço por conta de rotina de trabalho ou jornada acadêmica.

Os critérios percebidos a respeito da **Relevância** exemplificam que 86,6% dos colaboradores afirmaram concordar que o conteúdo abordado no RP é importante. Além de que, 86,6% informaram concordar com o fato de que o conteúdo do RP tem total ligação com a disciplina. Outros 93,3% afirmaram que o RP é um método adequado para a disciplina, por fim, foi perceptível que 73,3% preferiram aprender com o RP do que de outro jeito. Os valores atingidos aqui são o reflexo do RP como metodologia ativa, o que permite aos participantes ter uma aprendizagem prática diferente de aulas expositivas que o aluno acaba tendo um papel passivo na aprendizagem. Outro ponto relatado pelos participantes durante as observações da dinâmica consiste no fato de a prática possibilitar uma experiência na qual o aluno cria um projeto, faz sua publicação, bem como monitora suas execução. Isso foi algo positivamente

muito pontuado pelos alunos.

Finalmente, sobre a **Percepção de Aprendizagem** percebeu-se que 93,3% dos participantes concordaram que o RP contribuiu para a aprendizagem. Além disso, 86,6% deles expressaram concordar ao relatarem que o RP foi eficaz na aprendizagem, além do que, 86,6% disseram concordar que o RP foi um fator direto que contribuiu para agregar conhecimentos. Por fim, 90% dos membros concordaram que o RP comparado com uma aula normal sobre DevOps foi mais fascinante. O mérito destes resultados estão atrelados a alguns fatores que ajudaram a criar um ambiente propício ao desenvolvimento cognitivo dos participantes. Aqui o principal conceito utilizado foi a aprendizagem ativa, na forma de um RP, que por meio de conceitos teóricos e práticos, cuidadosamente escolhidos, se pode solidificar os conhecimentos dos alunos. Ao utilizar esse elemento, foi possível criar um ambiente que não apenas facilitou a aquisição de conhecimento, mas também enriqueceu a percepção subjetiva da experiência de aprendizagem.

Ao utilizar o modelo MEEGA+ adaptado, buscou-se avaliar a experiencia do jogar, concentrando-se na sua perspectiva. Desta forma, os resultados revelaram que a organização do conteúdo e o acesso a recursos como uma *wiki* puderam contribuir para a confiança dos participantes. Além do mais, pontos como a interação social, colaboração e compartilhamento de conhecimento foram bastante valorizados. Isso fez com que na opinião dos alunos o RP uma forma muito mais fascinante de aprender sobre DevOps do que uma aula convencional. Por fim, esse ensino ativo pode proporcionar um ambiente adequado para o desenvolvimento cognitivo, solidificando sabres dos alunos e enriquecendo suas experiências.

6.4 Análise de feedbacks gerais sobre a experiência como um todo

A presente seção tem como foco a análise temática de conteúdo referente à questão aberta com foco na experiência do estudante como um todo em relação ao RP, realizada no após a conclusão da dinâmica. Nesse sentido, verificou-se dois temas distintos, conforme sintetizado na Quadro 2

Quadro 2 – Síntese da Análise Temática de Conteúdo referente à experiência como um todo.

Tema	Códigos	Participantes
Aspectos Positivos	Explorando Práticas e Ferramentas	P1,P2,P6,P14,P28,P30
	Aprendizagem em Pares	P8, P11
	Aceitação do RP	P6,P12,P13,P16,P15,18,P24,P28
Perspectivas de Melhorias	Gestão de Tempo	P14,P15,P16,P19
	Otimização da Dinâmica	P13,P19,P20,P28

Fonte: Autoria própria (2024).

Sobre o primeiro tema, ou seja, com foco nos **Aspectos Positivos**, observou-se três códigos distintos. O primeiro código Explorando Práticas e Ferramentas reforçou a percepção positiva dos alunos sobre o uso de ferramentas no processo de aprendizado. Sendo assim, P1 expôs: "*Gostei bastante da parte de CI e CD*"; já P2, destacou em sua fala o seguinte mérito: "*Usar o SonarQube para ver os Code Smell*"; por sua vez, P6 trouxe em sua resposta a seguinte descrição: "*Foi a integração com o Fly, uma ferramenta nova que aprendi e que posso futuramente usar em projetos, sua configuração complexa pode ser desafiadora, mas para aplicações pequenas torna-se bem fácil a utilização, também a questão de iteração com logs foi fundamental para a compreensão de como é feita a ligação de produção com outra ferramenta que pegue esses arquivos*"; o fato de haver o momento prático dos tópicos abordados foi um fator positivo, como destacado por P14: "*O fato de fazer na prática, mão na massa. Isso é essencial para fixar melhor o aprendizado*"; por sua vez, em sua resposta, P28 destacou: "*A interatividade diferente da vista em salas de aula normais*"; por fim, P30 fez a seguinte menção: "*Rodar a aplicação na web utilizando o fly*". As falas dos participantes puderam revelar um entusiasmo com a implementação de práticas modernas de desenvolvimento, como CI/CD. Os mesmos também destacaram o uso do *SonarQube* para identificar potenciais problemas no código evidenciando uma preocupação com a qualidade. A integração com o *Fly.io* foi algo positivo na ótica de utilização de projetos futuros, além disso se pode destacar a experiência interativa diferente da abordada em sala de aula.

Outro código identificado, diz respeito a Aprendizagem em Pares para promoção do aprendizado. Sabe-se que a programação em pares proporciona uma troca de conhecimentos entre seus adeptos, além de melhorar a qualidade do código, principalmente no que diz respeito a erros. Isso ficou evidenciado nas fala de P8 e P11, onde o mesmos destacaram o uso acertado do *pair programming* e colaboração constante entre a dupla. P8 disse que: "*O que mais gostei foi a escolha da programação em pares, minha conclusão foi que essa escolha foi indispensável para a boa assimilação dos conhecimentos e para forçar o aluno participante a interagir o máximo possível*"; P31, por sua vez, afirmou: "*A divisão dos papéis e a contribuição constante com minha dupla*".

Outro código que emergiu das respostas dos participantes foi Aceitação do RP, tendo em vista que a avaliação das falas revelaram uma recepção positiva em relação à experiência de aprendizado em DevOps, assim como em uma abordagem clara e acessível em relação a temas complexos, tornando assim o conteúdo compreensível. Isso ficou claro na fala de P6:

"*Bem expositivo e de fácil entendimento, esta prática poderia descomplicar temas complexos e suas peculiaridades*"; P16 ponderou a prática de RP como positiva ao lhe proporcionar novos conhecimentos uma vez que o mesmo relatou: "*Gostei bastante da prática em si, por adquirir novos conhecimentos*"; P12, P13 e P18 ressaltaram o interesse pela a área de DevOps, uma vez que disseram respectivamente: "*Achei muito interessante o conteúdo abordado, me explicou muita coisa mostrado em DevOps*"; "*A iniciativa foi muito boa, e ótima para enfatizar a importância da área*"; e "*Conhecimento de DevOps é extremamente importante*"; já P15 destacou que: "*No geral o RP foi uma experiência alternativa e divertida*". Desta maneira, a percepção geral é de uma prática valiosa, trazendo conhecimentos significativos sobre tecnologias e ferramentas importantes. Nesse sentido, P24 registrou que: "*Levar essa dinâmica para as salas de aula*"; P28, por sua vez avaliou que: "*A dinâmica apresentada bem como os assuntos abordados foram de grande ajuda e trouxeram conhecimento sobre tecnologias e ferramentas importantes*".

O segundo tema, por sua vez, derivou códigos relacionados as **Perspectivas de Melhorias**, ou seja, o que poderia ser melhorado na dinâmica do RP. Nesse contexto, o primeiro código se refere à Gestão de Tempo. Algumas respostas indicaram uma preocupação mais latente com o tempo disponibilizado para a ocorrência da prática, além disso aulas dedicadas a cada fase para uma assimilação mais aprofundada. P14 relatou: "*Mais tempo, muita informação em um momento curto*"; similar a essa fala, P16 disse que: "*Mais tempo e aulas dedicadas a cada fase*"; uma outra sugestão fez referência ao fato de omitir alguns conceitos somente teóricos a fim de deixar a prática mais rápida, como foi mencionado por P19: "*Talvez se algumas partes somente de teoria fossem tiradas para ser mais rápido*"; Houve também a sugestão de um ponto, onde o mesmo diz respeito à complexidade da segunda fase que tomou boa parte da dinâmica, o que fica claro na fala de P15: "*O tempo entre as fases. A segunda foi mais complexa e tomou boa parte do tempo*". Em resumo, todas as falas caminham para a necessidade de uma abordagem balanceada e flexível para otimizar o aprendizado durante a aplicação do RP.

Ainda no que se refere às melhorias da dinâmica, foi constatado outro código, com foco em sugestões para Otimização da Dinâmica. Aqui as colocações dos alunos destacaram possíveis aspectos relevantes para otimizar a dinâmica. Uma delas foi colocado por P20, que alegou: "*Ajustar melhor de alguma forma as etapas para tentar prever erros comuns de usuários de instalação etc*"; nesse sentido, P13, disse: "*Talvez tornar mais claro o contexto, além da prática em si, trazer exemplos funcionais e operacionais durante a implementação*"; por sua

vez, P19: "*Acredita que uma introdução relacionado às tecnologias utilizadas nos exemplos e práticas*". Paralelamente, P28 discorreu que: "*Adicionar mais colaboradores para que se pudesse de fato utilizar todos os papéis, como por exemplo o NPC*". P13 salientou necessidade de exemplos funcionais e operacionais durante a implementação, mas toda a dinâmica foi concebida para operacionalidade e funcionalidade.

7 DISCUSSÃO

Esta pesquisa teve como objetivo geral investigar as experiências e percepções de estudantes de graduação sobre a adoção de RP para o aprendizado de DevOps. Como forma de alcançar tal objetivo, definiu-se a seguinte questão de pesquisa: *Quais são as experiências e percepções de estudantes de graduação sobre a adoção de Role Play para o aprendizado de DevOps*? A motivação para essa questão está em torno de evidenciar experiências e percepções, por parte de alunos concludentes da disciplina de ES, ofertadas nos cursos de Ciências da Computação e Sistemas de Informações na Universidade Federal do Ceará no *Campus* Crateús. Desta maneira, pode-se avaliar de foma experimental as percepções dos alunos durante a aplicação do modelo de RP. Ao identificar e compreender os relatos, por meio de questionários em conjunto com observação participante, tornou-se possível prover análises práticas e consequentemente contribuir para o aprimoramento do entendimento sobre ensino de DevOps, beneficiando discentes e docentes na busca por abordagens didáticas que melhorem o aprendizados.

Os resultados dos questionamentos feitos aos 30 participantes, possibilitou a identificação de discussões pertinentes para a questão de pesquisa proposta neste trabalho. Um exemplo disso foi a Aprendizagem Cooperativa o qual perpassou entre as três fases do RP (Fase 1, Fase 2 e Fase 3), com destaque para a adoção de programação em pares. Outro ponto que pode ser observado foi o compartilhamento de informações, ideias, sugestões e experiências relevantes, de indivíduo uns com os outros (BARTOL; SRIVASTAVA, 2002). A adoção de programação em pares e compartilhamento de informações foram valiosas para a interação entre os alunos, pois auxiliaram a quebrar o gelo entre os envolvidos, contribuindo para que que os mesmos colaborassem uns com os outros em relação à construção de conhecimento, permitindo, assim, uma fluidez durante toda a dinâmica seja ela de forma prática.

O momento de dramatização de papéis foi importante, pois ao usar o RP, além de o mesmo poder proporcionar meios de aplicar cargas de conteúdos teóricos faz-se imprescindível que exista momentos de práticas condizentes com a teoria. Neste sentido, há um consenso entre os pesquisadores da área de *software*, em que abordagens práticas são relevantes para o ensino de ES (PORTELA *et al.*, 2016). Adicionalmente, alguns autores destacam que este ensino deve ser mais centrado no aluno, a fim de aumentar sua motivação, participação e, consequentemente, sua aprendizagem (PRIKLADNICKI *et al.*, 2009; CICO *et al.*, 2021). Por meio das análises realizadas, pode-se observar que a aprendizagem envolvendo ferramentas exploradas na indústria forneceu experiências e aprendizados importantes, incluindo a perspectiva de exercitar diferentes

processos, como integração contínua, controle de versão, análise estática de código, e engenharia de lançamento. Há de se destacar também a importância de práticas como padronização de nomenclatura, registro de *logs* e a automação de *pipelines*. Tais elementos contribuem para a eficiência e qualidade de processos de ES, bem como para a colaboração de equipes. **Quanto aos desafios e pontos positivos**, alguns apontamentos foram feitos em relação a complexidade de processos da dinâmica com suas nomenclaturas e ferramentas utilizadas, mas compreensíveis no geral. Por sua vez, a didática e a clareza durante a explicação foram importantes, bem como a relevância do assunto abordado.

Sobre o que poderia ser aprimorado na dinâmica do RP, pode demonstrar que a Gestão do Tempo foi um tópico importante, em particular, este ponto evidenciou algumas dificuldades apontadas pelos alunos em conseguir assimilar parte do conteúdo proposto, dentro do tempo estipulado, havendo ainda alguns relatos de sobrecarga do conteúdo. Outro fato que foi evidenciado foi o desconhecimento de alguns conceitos teórico/práticos abordados como a criação de *pipelines* de DevOps e até mesmo o *deploy* da aplicação criada por eles.

No caso desta pesquisa, se buscou elaborar e avaliar um modelo de RP que suprisse as necessidades acadêmicas e profissionais no ensino de DevOps. **Do ponto de vista de contribuições acadêmicas**, averiguou-se uma prática imersiva em conceitos teóricos e práticos sobre situações reais e desafiadoras envolvendo DevOps, além de proporcionar aos docentes um método de ensino alternativo, que prenda a atenção do aluno, podendo promover momentos valiosos, muitas vezes não contemplados em aulas de caráter expositivo. **No que corresponde as contribuições para indústria**, os alunos puderam ver como funciona minimamente uma esteira de programação desde o recebimento das demandas, seu processamento, atrelados a questões que envolvem versionamento de código, teste, automatização de processos de subida de código e o monitoramento de aplicações. Finalmente, através das discussões apresentadas se pode averiguar o ganho real para os discentes, uma vez impactados por uma metodologia que lhes proporcionaram engajamento com seus pares e o compartilhamento de conhecimento que para os docentes torna-se uma saída de aulas tradicionais expositivas.

8 LIMITAÇÕES

Este estudo considera um conjunto de limitações que podem ter influenciado os resultados e a interpretação dos dados obtidos. No que se relaciona a coleta dos dados, uma das limitações está relacionada ao tamanho da amostra de participantes, que neste estudo cobriu uma amostra de 30 estudantes de graduação em Ciência da Computação e Sistemas de Informação. Ainda que se tenha obtido resultados relevantes para a amostra, é importante aumentar a amostragem considerando a heterogeneidade de perfis de participantes, cursos e até regiões do Brasil. Todavia, reconhece-se que a quantidade pessoas atingida, sendo divida em três turmas distintas, com diferentes perfis, colaborou para o cumprimento do objetivo proposto.

Ademais, os participantes foram expostos de forma simultânea aos questionários de formulário de feedback parcial entre as fases e ao questionário final baseado no MEEGA+ (WANGENHEIM *et al.*, 2018) ao final da dinâmica de RP, proporcionando, assim, um ambiente mais controlado. Parte do tema investigado era de conhecimento dos participantes, o que ajudou a mitigar algumas ameaças, como o desalinhamento de conhecimentos básicos. Adicionalmente, buscou-se, antes de aplicar a dinâmica, explicar didaticamente sobre a a metodologia proposta, incluindo os conceitos e práticas atrelados a mesma. Adicionalmente, busco-se definir horários para as sessões que fossem mais convenientes para a participação dos discentes.

Além disso, é fundamental saber que o presente estudo se propôs a investigar a capacidade e entendimento dos estudantes de graduação, no que compreende a adoção de aprendizados ativos com o foco em DevOps. Com isso, o grau de conhecimento pode variar entre os participantes, pois apesar de um critério obrigatório ter sido a conclusão da disciplina de ES, ficou nítido que nem todos estavam no mesmo patamar de entendimento de conceitos teórico/prático. Para atenuar essa problemática foi utilizada a prática de programação em pares. Além disso, os alunos poderiam ajudar outros participantes fora da sua dupla. Por sua vez, o mestre da dinâmica também poderia intervir, sempre que houvesse dificuldades com o código ou débitos técnicos com as ferramentas utilizadas. Os alunos também dispunham de uma *wiki* de apoio. Ao analisar as problemáticas referentes ao comportamento dos alunos, é importante atentar-se a possíveis condutas, como a introversão causada por estarem em grupos ou mesmo por estarem sendo observados. Sendo assim, esses pontos podem influenciar de forma negativa na ocorrência do modelo proposto. Entretanto, dentro da dinâmica ao fortalecer a apresentação dos participantes, bem como conversas informais com o condutor da prática é possível atenuar questões comportamentais ligadas a introversão.

9 CONSIDERAÇÕES FINAIS

O uso de *Role-Play* (RP) no ensino de DevOps em uma disciplina de Engenharia de Software (ES) demonstra-se uma abordagem inovadora que promove a imersão dos alunos em cenários práticos. Através da simulação de situações reais, os estudantes podem vivenciar desafios e tomar decisões como profissionais da área. Isso não apenas fortalece a compreensão dos conceitos teóricos, mas também desenvolve habilidades de trabalho em equipe e resolução de problemas.

Diante destas circunstâncias, o presente trabalho teve como objetivo investigar as experiências e percepções de estudantes de graduação sobre a adoção de RP para o aprendizado de DevOps. Quanto aos procedimentos metodológicos, estipulou-se quatro etapas principais. Na primeira etapa, foi feito uma análise *ad-hoc* da literatura, visando compreender os conceitos principais e as definições teóricas que relacionam DevOps e RP no ensino de ES. Na segunda etapa, foi elaborado o modelo de ensino, a identificação dos seus conceitos, a elaboração da narrativa e a definição do material de apoio. Na terceira etapa, houve a coleta dos dados, onde exploro-use o modelo de RP na prática, aplicou-se os questionários entre fases e o questionário final adaptado do modelo MEEGA+. Na quarta e última etapa, foram realizadas a Análise Estatística Descritiva e Análise Temática de Conteúdo.

No que se refere os resultados obtidos, inicialmente, através da análise das fases por meio de questionário, foi possível chegar a uma avaliação quantitativa do nível de compreensão (em uma escala de 1 a 5) sobre o nível de compreensão a cada fase. Em média, considerando as três turmas, verificou-se que para as Fases 1, 2 e 3, 93,3% dos alunos alegaram um alto nível de compreensão (4 e 5). Ao se analisar a questão aberta sobre cada fase, foi possível evidenciar uma aceitação positiva dos participantes, especialmente em decorrência da aprendizagem cooperativa alinhada com processos/tecnologias explorados na indústria. Uma pessoa discente (P8) destacou: "*A dinâmica de programação em pares foi muito produtiva pois mostrou a importância da cooperação e comunicação entre os pares tanto do time, como também, de toda a empresa*". De forma complementar, também foi possível obter analises das categorias oriundas do modelo MEEGA+ com foco na experiência do jogador, entre elas a percepção de aprendizado, onde 93% dos participantes concordaram que o RP pode contribuir para o aprendizado. Por fim, ao analisar os *feedbacks* sobre a experiência como um todo, aspectos positivos também ficaram evidentes, tangenciando questões como aprendizagem em pares, porém, com perspectivas de melhorias, como gestão de tempo e otimizações da dinâmica.

Em suma, as principais contribuições desta pesquisa, pode-se destacar, sob a perspectiva acadêmica, que o modelo de ensino proposto baseado em RP oferece uma oportunidade engajadora para docentes e discentes explorarem o processo de capacitação no contexto de DevOps. No que se às contribuições para a prática/indústria, tem-se a possibilidade de contribuir para a formação de profissionais mais preparados e aptos para lidar com processos adotados em situações análogas realidade de um ambiente de trabalho em desenvolvimento de software. Tal questão estimula uma formação que contempla tanto a lapidações de habilidades técnicas, mas também interpessoais.

Finalmente, quanto aos trabalhos futuros, pode-se investigar a adaptação do modelo proposto em contextos diferentes da ES. Uma outra possibilidade trata-se de adaptar o conteúdo proposto para sua execução de forma assíncrona, na forma de videoaula, pelos alunos, como um complemento da disciplina de ES.

REFERÊNCIAS

ALKIN, M. C.; CHRISTIE, C. A. The use of role-play in teaching evaluation. **American Journal of Evaluation**, Sage Publications Sage CA: Thousand Oaks, CA, v. 23, n. 2, p. 209–218, 2002.

ALVES, I.; ROCHA, C. Qualifying software engineers undergraduates in devops-challenges of introducing technical and non-technical concepts in a project-oriented course. In: IEEE. **2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)**. [S.l.], 2021. p. 144–153.

ATLASSIAN. **What is Continuous Integration**. 2021. Acessado em: 19/11/2021. Disponível em: <https://www.atlassian.com/continuous-delivery/continuous-integration>.

ATLASSIAN. **Gerenciamento de código-fonte.** 2023. Acessado em:11/02/2023. Disponível em: <https://www.atlassian.com/br/git/tutorials/source-code-management>.

AZERI, I. **What is DevOps?** 2017. Acessado em: 24/03/2023. Disponível em: <https://dzone.com/articles/what-is-cicd>.

BARDIN, L. Análise de conteúdo. **Lisboa: Edições**, 1979.

BARTOL, K. M.; SRIVASTAVA, A. Encouraging knowledge sharing: The role of organizational reward systems. **Journal of leadership & organizational studies**, Sage Publications Sage CA: Thousand Oaks, CA, v. 9, n. 1, p. 64–76, 2002.

BECK, K. Embracing change with extreme programming. **Computer**, IEEE, v. 32, n. 10, p. 70–77, 1999.

BHATT, D. A survey of effective and efficient software testing technique and analysis. **Iconic Research and Engineering Journals (IREJOURNALS)**, v. 326, 2017.

BLOOM, B. S.; KRATHWOHL, D. R.; MASIA, B. B. *et al.* Bloom taxonomy of educational objectives. In: **Allyn and Bacon**. [S.l.]: Pearson Education London, 1984.

BONWELL, C. C.; EISON, J. A. **Active learning: Creating excitement in the classroom. 1991 ASHE-ERIC higher education reports.** [S.l.]: ERIC, 1991.

BOOCH, G. **Object oriented design with applications**. [S.l.]: Benjamin-Cummings Publishing Co., Inc., 1990.

BRAUN, V.; CLARKE, V. Using thematic analysis in psychology. qualitative research in psychology. **Qualitative Research in Psychology**, v. 3, n. 2, p. 77–101, 2006.

BUSARELLO, R. I.; ULBRICHT, V. R.; FADEL, L. M. A gamificação e a sistemática de jogo: conceitos sobre a gamificação como recurso motivacional. **Gamificação na educação. São Paulo: Pimenta Cultural**, p. 11–37, 2014.

CANOREA, E. **First steps on GitHub Actions**. 2021. Acessado em: 05/12/2021. Disponível em: <https://www.plainconcepts.com/what-is-github-actions/#Benefits_of_GitHub_Actions>.

CAPOZUCCA, A.; GUELFI, N.; RIES, B. Design of a (yet another?) devops course. In: SPRINGER. **Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1**. [S.l.], 2019. p. 1–18.

CARDOSO, A. F. M. P.; VIEIRA, M. d. F. d. S. B.; MATOS, M. E. E. d.; TOMÉ, S. A. d. M. O role play como ferramenta no desenvolvimento das competências comunicativas dos alunos do ensino básico. Porto:[Edição do Autor], 2009.

CARNEIRO, A. P. S. **Implantação do Graylog como ferramenta de centralização de logs de dados no Instituto Metrópole Digital-IMD/UFRN**. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2022.

CECHIN, M. R. Role play para graduandos e o ensino da leitura. **Linguagens & Cidadania**, 2002.

CHEN, L. Continuous delivery: Huge benefits, but challenges too. **IEEE software**, IEEE, v. 32, n. 2, p. 50–54, 2015.

CHERIF, A. H.; SOMERVILL, C. H. Maximizing learning: using role playing in the classroom. **The American Biology Teacher**, JSTOR, p. 28–33, 1995.

CHERON, M. Aplicação da técnica role-play na educação profissional. 2019.

CHRISTENSEN, H. B. Teaching devops and cloud computing using a cognitive apprenticeship and story-telling approach. In: **Proceedings of the 2016 ACM conference on innovation and technology in computer science education**. [S.l.: s.n.], 2016. p. 174–179.

CICO, O.; JACCHERI, L.; NGUYEN-DUC, A.; ZHANG, H. Exploring the intersection between software industry and software engineering education - a systematic mapping of software engineering trends. **Journal of Systems and Software**, v. 172, p. 110736, 2021. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121220301667>.

CLAPS, G. G.; SVENSSON, R. B.; AURUM, A. On the journey to continuous deployment: Technical and social challenges along the way. **Information and Software technology**, Elsevier, v. 57, p. 21–31, 2015.

CLOUD, G. **Medida de DevOps: monitoramento de sistemas para basear decisões de negócios**. 2022. Acessado em: 19/03/2022. Disponível em: <https://cloud.google.com/architecture/devops/devops-measurement-monitoring-systems?hl=pt-br>.

CLOUD, G. **Integração contínua (CI)**. 2023. Acessado em: 31/03/2023. Disponível em: <https://cloud.google.com/solutions/continuous-integration?hl=pt-br>.

CLOUD, G. **Tecnologia do DevOps: automação de implantação**. 2023. Acessado em: 21/02/2023. Disponível em: <https://cloud.google.com/architecture/devops/devops-tech-deployment-automation?hl=pt-br>.

CRUZES, D. S.; DYBA, T. Recommended steps for thematic synthesis in software engineering. In: IEEE. **2011 international symposium on empirical software engineering and measurement**. [S.l.], 2011. p. 275–284.

DECKER, A.; SIMKINS, D. Leveraging role play to explore software and game development process. In: IEEE. **2016 IEEE Frontiers in Education Conference (FIE)**. [S.l.], 2016. p. 1–5.

DEPLOY, O. **What Is CI/CD?** 2023. Acessado em: 23/03/2023. Disponível em: <https://octopus.com/devops/>.

DIXON, A.; JAGODZINSKI, A. Qualitative data analysis of thinking-aloud role-play exercises: Usefulness in software-engineering requirements analysis. **Concurrent Engineering: Advanced Design, Production and Management Systems**, v. 20, p. 107–113, 2003.

DU, M.; LI, F. Spell: Streaming parsing of system event logs. In: IEEE. **2016 IEEE 16th International Conference on Data Mining (ICDM)**. [S.l.], 2016. p. 859–864.

DUVALL, P. M.; MATYAS, S.; GLOVER, A. **Continuous integration: improving software quality and reducing risk**. [S.l.]: Pearson Education, 2007.

EGELE, M.; SCHOLTE, T.; KIRDA, E.; KRUEGEL, C. A survey on automated dynamic malware-analysis techniques and tools. **ACM computing surveys (CSUR)**, ACM New York, NY, USA, v. 44, n. 2, p. 1–42, 2008.

FARROHA, B. S.; FARROHA, D. L. A framework for managing mission needs, compliance, and trust in the devops environment. In: IEEE. **2014 IEEE Military Communications Conference**. [S.l.], 2014. p. 288–293.

FERREIRA, T. d. S. D.; VIANA, D.; SANTOS, R. P. dos. Árvore de ecos: Um jogo para ensino de conceitos de ecossistemas de software. **Revista Brasileira de Informática na Educação**, v. 29, p. 273–300, 2021.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering: A roadmap and agenda. **Journal of Systems and Software**, Elsevier, v. 123, p. 176–189, 2017.

FOWLER, M. **Continuous Integration**. 2006. Acessado em: 19/11/2021. Disponível em: <https://www.martinfowler.com/articles/continuousIntegration.html>.

GABRIELLI, S.; KIMANI, S.; CATARCI, T. The design of microlearning experiences: A research agenda (on microlearning). academia. edu, 2017.

GEDIGAMES, G. d. E.; GAMES, D. d. I. de. Relatório final: Mapeamento da indústria brasileira e global de jogos digitais. **São Paulo: Universidade de São Paulo**, 2014.

GOMES, L. T. P. S. Criação de um documento de padronização de codificação java no lcc. Universidade Federal de Minas Gerais, 2013.

HAYCRAFT, J. **An Introduction to English Language Teaching: England**. [S.l.]: Longman Group, 2010.

HENRY, T. R.; LAFRANCE, J. Integrating role-play into software engineering courses. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 22, n. 2, p. 32–38, 2006.

HILTON, M.; TUNNELL, T.; HUANG, K.; MARINOV, D.; DIG, D. Usage, costs, and benefits of continuous integration in open-source projects. In: IEEE. **2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)**. [S.l.], 2016. p. 426–437.

HOLTON, J. A. The coding process and its challenges. **The Sage handbook of grounded theory**, v. 3, p. 265–289, 2007.

HUMBLE, J. **Continuous Delivery vs Continuous Deployment**. 2010. Acessado em: 07/11/2021. Disponível em: <https://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/>.

HUMBLE, J. **What is Continuous Delivery?** 2010. Acessado em: 07/11/2021. Disponível em: <https://continuousdelivery.com/>.

HUMBLE, J.; FARLEY, D. **Continuous delivery: reliable software releases through build, test, and deployment automation**. [S.l.]: Pearson Education, 2010.

JABBARI, R.; ALI, N. bin; PETERSEN, K.; TANVEER, B. What is devops? a systematic mapping study on definitions and practices. In: **Proceedings of the Scientific Workshop Proceedings of XP2016**. [S.l.: s.n.], 2016. p. 1–11.

KHAN, M. O.; JUMANI, A. K.; FARHAN, W. A. Fast delivery, continuously build, testing and deployment with devops pipeline techniques on cloud. **Indian Journal of Science and Technology**, v. 13, n. 05, p. 552–575, 2020.

KIM, G.; BEHR, K.; SPAFFORD, K. **The phoenix project: A novel about IT, DevOps, and helping your business win**. [S.l.]: IT Revolution, 2014.

KIM, G.; HUMBLE, J.; DEBOIS, P.; WILLIS, J.; FORSGREN, N. **The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations**. [S.l.]: IT Revolution, 2021.

KINSMAN, T.; WESSEL, M.; GEROSA, M. A.; TREUDE, C. How do software developers use github actions to automate their workflows? **arXiv preprint arXiv:2103.12224**, 2021.

KRUSCHE, S.; ALPEROWITZ, L. Introduction of continuous delivery in multi-customer project courses. In: **Companion Proceedings of the 36th International Conference on Software Engineering**. [S.l.: s.n.], 2014. p. 335–343.

KUŚNIEREK, A. Developing students' speaking skills through role-play. **World Scientific News**, n. 1, p. 73–111, 2015.

KUUSINEN, K.; ALBERTSEN, S. Industry-academy collaboration in teaching devops and continuous delivery to software engineering students: towards improved industrial relevance in higher education. In: IEEE. **2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)**. [S.l.], 2019. p. 23–27.

LADOUSSE, G. P. **Role play**. [S.l.]: Oxford University Press, 1987. v. 3.

LEITE, L.; ROCHA, C.; KON, F.; MILOJICIC, D.; MEIRELLES, P. A survey of devops concepts and challenges. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 52, n. 6, p. 1–35, 2019.

LEPPÄNEN, M.; MÄKINEN, S.; PAGELS, M.; ELORANTA, V.-P.; ITKONEN, J.; MÄNTYLÄ, M. V.; MÄNNISTÖ, T. The highways and country roads to continuous deployment. **Ieee software**, IEEE, v. 32, n. 2, p. 64–72, 2015.

LIMA, A.; ROSSI, L.; MUSOLESI, M. Coding together at scale: Github as a collaborative social network. In: **Eighth international AAAI conference on weblogs and social media**. [S.l.: s.n.], 2014.

LIMA, J. V. V.; SILVA, C. A. D.; ALENCAR, F. M. R. de; SANTOS, W. B. Metodologias ativas como forma de reduzir os desafios do ensino em engenharia de software: diagnóstico de um survey. In: SBC. **Anais do XXXI Simpósio Brasileiro de Informática na Educaçao**. [S.l.], 2020. p. 172–181.

MAGAZINE, R. D. **Automatizando o processo de build - Revista .Net Magazine 101**. 2012. Acessado em: 13/03/2022. Disponível em: <https://www.devmedia.com.br/revista-net-magazine-101/26554>.

MAILEWA, A.; HERATH, J.; HERATH, S. A survey of effective and efficient software testing. In: **The Midwest Instruction and Computing Symposium.(MICS), Grand Forks, ND**. [S.l.: s.n.], 2015.

MARINS, E. S. **O uso de Role-Playing Game (RPG) no ensino de Ciências: uma atividade voluntária e complementar às aulas no Ensino Fundamental II**. Tese (Doutorado) — Universidade de São Paulo, 2017.

MATTAR, J. **Games em educação**. [S.l.]: Pearson Educación, 2010.

MAXIM, B. R.; BRUNVAND, S.; DECKER, A. Use of role-play and gamification in a software project course. In: IEEE. **2017 IEEE frontiers in education conference (FIE)**. [S.l.], 2017. p. 1–5.

MELLADO, R. P.; MONTINI, D. Á.; DIAS, L. A. V.; CUNHA, A. M. da *et al.* Software product measurement and analysis in a continuous integration environment. In: IEEE. **2010 Seventh International Conference on Information Technology: New Generations**. [S.l.], 2010. p. 1177–1182.

MICROSOFT. **O que é o DevOps?** 2022. Acessado em: 19/03/2022. Disponível em: <https://azure.microsoft.com/pt-pt/overview/what-is-devops/#devops-overview>.

MICROSOFT. **O que é controle de versão?** 2023. Acessado em: 11/02/2023. Disponível em: <https://learn.microsoft.com/pt-br/devops/develop/git/what-is-version-control>.

MILES, M. B.; HUBERMAN, A. M.; HUBERMAN, M. A.; HUBERMAN, M. **Qualitative data analysis: An expanded sourcebook**. [S.l.]: sage, 1994.

MILLS, E. E. **Software metrics**. [S.l.], 1988.

MORENO, J. L.; MORENO, Z. Psicodrama: Terapia de ação e princípios da prática. **Daimon.(Obra original publicada em 1969)**, 2006.

MOTA, L. L. **Repositório GitHub contendo Slide para a dinâmica RP**. 2023. Acessado em: 06/05/2023. Disponível em: <https://github.com/lucaslmota/DeO-DevOps>.

NAKAMURA, T.; MARUYAMA, H.; TAKASHIMA, A.; SAMBE, Y. Role-play exercises for project management education that incorporate a software agent. In: IEEE. **Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012**. [S.l.], 2012. p. W2A–8.

NASCIMENTO, C. I. d. C. **Role-plays: promovendo a produção oral nas aulas de língua inglesa**. Dissertação (Mestrado) — Brasil, 2019.

NESTEL, D.; TIERNEY, T. Role-play for medical students learning about communication: guidelines for maximising benefits. **BMC medical education**, Springer, v. 7, p. 1–9, 2007.

NETO, J. G. d. R. **Entendendo a relação entre integração contínua e cobertura de testes: um estudo empírico**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2021.

NICOLLETTI, P. S. **O Processo de Desenvolvimento de Software**. 2002. Acessado em: 16/01/2022. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/intro/processo.htm>.

NOVAK, J. **Desenvolvimento de games**. [S.l.]: Cengage Learning, 2011.

O'HANLON, C. A conversation with werner vogels: Learning from the amazon technology platform: Many think of amazon as' that hugely successful online bookstore.'you would expect amazon cto werner vogels to embrace this distinction, but in fact it causes him some concern. **Queue**, ACM New York, NY, USA, v. 4, n. 4, p. 14–22, 2006.

PAIR, A. **Continuous Deployment For Practical People**. 2023. Acessado em: 21/02/2023. Disponível em: <https://www.airpair.com/continuous-deployment/posts/continuous-deployment-for-practical-people>.

PERERA, P.; SILVA, R.; PERERA, I. Improve software quality through practicing devops. In: IEEE. **2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)**. [S.l.], 2017. p. 1–6.

PETRI, G.; WANGENHEIM, C. G. von; BORGATTO, A. F. Evolução de um modelo de avaliação de jogos para o ensino de computação. In: SBC. **Anais do XXV Workshop sobre Educação em Computação**. [S.l.], 2017.

PETRI, G.; WANGENHEIM, C. Gresse von; BORGATTO, A. F. Meega+: Um modelo para a avaliação de jogos educacionais para o ensino de computação. **Revista Brasileira de Informática na Educação**, v. 27, n. 3, 2019.

PINNA-DÉRY, A.-M. Teaching devops at the graduate level. In: SPRINGER. **Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers**. [S.l.], 2019. v. 11350, p. 60.

PITTET, S. **Integração contínua vs. entrega contínua vs. implantação contínua**. 2021. Acessado em: 19/11/2021. Disponível em: <https://www.atlassian.com/br/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. R. B. Frames: Uma proposta de framework para o ensino de tópicos da engenharia de software. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2016. v. 27, n. 1, p. 1361.

PORTER, G. L. **Role play**. [S.l.]: Oxford University Press, 1989.

PRIKLADNICKI, R.; ALBUQUERQUE, A. B.; WANGENHEIM, C. G. von; CABRAL, R. Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas. **FEES-Fórum de Educação em Engenharia de Software**, p. 1–8, 2009.

PRINCE, S. **The Product Managers' Guide to Continuous Delivery and DevOps**. 2016. Acessado em: 06/11/2021. Disponível em: <https://www.mindtheproduct.com/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/>.

PRIYADARSINI, K.; RAJ, E. F. I.; BEGUM, A. Y.; SHANMUGASUNDARAM, V. Comparing devops procedures from the context of a systems engineer. **Materials Today: Proceedings**, Elsevier, 2020.

RABELO, L.; GARCIA, V. L. Role-play para o desenvolvimento de habilidades de comunicação e relacionais. **Revista Brasileira de Educação Médica**, v. 39, n. 4, p. 586–596, 2015.

REDONDO, A. M. F.; CÁRDENAS, F. d. J. N. *et al.* Devops: un vistazo rápido. **Ciencia Huasteca Boletín Científico de la Escuela Superior de Huejutla**, v. 10, n. 19, p. 35–40, 2022.

REDONDO, R. P. D.; VILAS, A. F.; ARIAS, J. J. P.; SOLLA, A. G. Collaborative and role-play strategies in software engineering learning with web 2.0 tools. **Computer applications in engineering education**, Wiley Online Library, v. 22, n. 4, p. 658–668, 2014.

RONG, G.; GU, S.; ZHANG, H.; SHAO, D. Devopsenvy: an education support system for devops. In: IEEE. **2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)**. [S.l.], 2017. p. 37–46.

SANYAL, R. **O que é Monitoramento Contínuo em DevOps?** 2022. Acessado em: 15/05/2023. Disponível em: <https://dtrends.com.br/o-que-e-monitoramento-continuo-em-devops/>.

SCHOTS, M.; SANTOS, R.; MENDONÇA, A.; WERNER, C. Elaboração de um survey para a caracterização do cenário de educação em engenharia de software no brasil. **Anais do II Fórum de Educação em Engenharia de Software, XXIII Simpósio Brasileiro de Engenharia de Software**, p. 57–60, 2009.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. **IEEE Access**, IEEE, v. 5, p. 3909–3943, 2017.

SILVA, M. V. *et al.* O jogo de papéis (rpg) como tecnologia educacional e o processo de aprendizagem no ensino médio. UNIVERSIDADE TUIUTI DO PARANÁ, 2009.

SILVA, R. R.; RIVERO, L.; SANTOS, R. P. D. Programse: Um jogo para aprendizagem de conceitos de lógica de programação. **Revista Brasileira de Informática na Educação**, v. 29, p. 301–330, 2021.

SKELTON, M.; O'DELL, C. **Continuous delivery with windows and. NET**. [S.l.]: O'Reilly Media, 2016.

SOMMERVILLE, I. Software engineering 9th edition. **ISBN-10**, v. 137035152, 2011.

SOMMERVILLE, I. Engenharia de software/ian sommerville. **Tradução Ivan Bosnic e karlinka G. de O. Gonçalves**, 2013.

STÅHL, D.; BOSCH, J. Cinders: The continuous integration and delivery architecture framework. **Information and Software Technology**, Elsevier, v. 83, p. 76–93, 2017.

SUNDMAN, Y. **Continuous Delivery vs Continuous Deployment**. 2013. Acessado em: 06/11/2021. Disponível em: <https://blog.crisp.se/2013/02/05/yassalsundman/continuous-delivery-vs-continuous-deployment>.

SWARTOUT, P. **Continuous delivery and DevOps: a quickstart guide**. [S.l.]: Packt Publishing Birmingham, 2012.

TANAKA, S.; VIANNA, M.; VIANNA, Y.; MEDINA, B. Gamification, inc.: como reinventar empresas a partir de jogos. mjv Press, 2013.

UNITY. **O que é CI/CD?** 2023. Acessado em: 04/03/2023. Disponível em: <https://unity.com/pt/solutions/what-is-ci-cd>.

VALENTE, M. T. Engenharia de software moderna (livro digital). 2020.

VASILESCU, B.; YU, Y.; WANG, H.; DEVANBU, P.; FILKOV, V. Quality and productivity outcomes relating to continuous integration in github. In: **Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering**. [S.l.: s.n.], 2015. p. 805–816.

WANGENHEIM, C.; HAUCK, G.; BOIGATTO, J.; ADRIANO, F.; PACHECO, L. **MEEGA+ A model for evaluating educational games**. 2018. Acessado em: 24/05/2023. Disponível em: <http://www.gqs.ufsc.br/quality-evaluation/meega-plus/>.

WEBER, I.; NEPAL, S.; ZHU, L. Developing dependable and secure cloud applications. **IEEE Internet Computing**, IEEE, v. 20, n. 3, p. 74–79, 2016.

WICHMANN, B. A.; CANNING, A.; CLUTTERBUCK, D.; WINSBORROW, L.; WARD, N.; MARSH, D. W. R. Industrial perspective on static analysis. **Software Engineering Journal**, London: published by the Institution of Electrical Engineers for the British . . . , v. 10, n. 2, p. 69, 1995.

YANG, Y.; LI, Z.; HE, L.; ZHAO, R. A systematic study of reward for reinforcement learning based continuous integration testing. **Journal of Systems and Software**, Elsevier, v. 170, p. 110787, 2020.

ZOWGHI, D.; PARYANI, S. Teaching requirements engineering through role playing: Lessons learnt. In: IEEE. **Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.** [S.l.], 2003. p. 233–241.

ZUPPIROLI, S.; CIANCARINI, P.; GABBRIELLI, M. A role-playing game for a software engineering lab: Developing a product line. In: IEEE. **2012 IEEE 25th Conference on Software Engineering Education and Training**. [S.l.], 2012. p. 13–22.

APÊNDICE A – DESCRIÇÃO TEXTUAL DETALHADA SOBRE AS FASES DO MODELO DE *ROLE-PLAY*

- **Fase #1**: tem como objetivo explorar a categoria de Compartilhamento de Conhecimento. Bartol e Srivastava (2002) definem compartilhamento de conhecimento como sendo, o compartilhamento de informações, ideias, sugestões e experiências organizacionalmente relevantes, do indivíduo com outros, e afirmam que o compartilhamento de conhecimento é um componente chave dos sistemas de gestão do conhecimento. Nesse caso, a orientação de conceitos se manifesta a partir das explicações sobre Cultura de colaboração e Compartilhando conhecimento. Na Cultura de colaboração faz-se o uso de quatro pontos: priorizar relacionamentos, comunicação, alinhamento de responsabilidades/incentivos e respeito. Para o Compartilhando conhecimento tem-se a aplicabilidade do conhecimento explícito e tácito aliados a programação em pares. A respectiva prática de conceitos se dará através da divão dos grupos de alunos, onde já dentro da dinâmica os mesmo devem primeiro se apresentar, falando seus nomes, cargos e formação. Após a essa breve apresentação será feita uma *stand-up meetings* a fim de que se possa discutir sobres os itens iniciais pedidos pelo PO. Isso fará com que os participantes tenham suas responsabilidades alinhadas e mantenham o respeito sobre questões pessoais e técnicas. Em seguida, relativo ao conhecimento explícito e tácito, as informações pertinentes a API como o *fork*, devem ser fornecidas nos slides e pelo *tech lead*. Para o conhecimento tácito deve ser feito o uso da programação em pares. Como objetivos para essa fase, espera-se que os participantes tenham entendido os pontos solicitados pelo PO, terem feito as devidas apresentações para seus pares sobre quem é você, enquanto colaborador, efetuado as divisões para a programação em pares e ter efetuado o *fork* do projeto e analisado o código base. Dando continuidade a fase um, O Gerenciamento de Código-fonte (SCM), é usado para rastrear modificações em um repositório de código-fonte. O SCM rastreia um histórico em execução de alterações em uma base de código e ajuda a resolver conflitos ao mesclar atualizações de vários colaboradores. SCM também é sinônimo de controle de versão(VCS) (ATLASSIAN, 2023). A categoria supracitado, possuí em sua orientação dos conceitos o controle de versão,o VCS por sua vez apresenta os seguintes benefícios: criar fluxos de trabalho, trabalhar com versões, codificar juntos, manter um histórico e automatizar tarefas (MICROSOFT, 2023). A prática aderente ao VCS se dará da seguinte forma, com o projeto devidamente baixado, os participante, se valendo da programação em pares,

devem agora baixar dois pacotes, o primeiro é *Nodemon* que é uma biblioteca que ajudará no desenvolvimento de sistemas com *NodeJs*, possibilitando reiniciar automaticamente o servidor e o segundo é o *Express* que é um *framework* que fornece recursos mínimos para a construção de servidores web. Dando prosseguimento a prática os participantes devem concluir as demandas solicitadas pelo PO, mas após a conclusão de cada método da API, duas convenções devem ser seguidas, a primeira é o Versionamento semântico(SemVer) e a outra é a Convenção de *commits*. Por exemplo, após a conclusão do método *Get* a equipe tem de atualizar a versão do projeto no *package.json* de acordo com os respectivos *Major*, *Minor* e *Patch* do SemVer e no momento de mandar a alteração para o repositório tem-se de seguir o padrão de *commit* adequado podendo este ser indicado pelo *tech lead*, caso haja dúvida de qual deva se utilizar. E esse mesmo processo se repete para cada método implementado. Espera-se para essa fase atingir os seguintes objetivos, o compartilhamento de conhecimento entre os pares na criação dos métodos da API, no arquivo *package.json* gerado, há uma *"version"*, adequá-lo ao padrão do SemVer e ter feito todos os *commit* e efetuar o *push* para o repositório *forkado* no *GitHub*.

- **A Fase #2**: se propõe a abordar a categoria Processo de construção (PC). O PC é o processo de criação de um sistema completo, executável por meio da construção e ligação dos componentes de sistema, bibliotecas externas, arquivos de configuração, etc (SOMMERVILLE, 2011). Consideramos não apenas ferramentas que geram pacotes implantáveis, também chamados de *builds*, mas também ferramentas que geram artefatos essenciais e *feedback* usando o código-fonte como entrada (LEITE *et al.*, 2019). A orientação dos conceitos da fase atual agrupa as explicações sobre três conceitos: Engenharia de lançamento(EL), Automação de testes(AT) e Análise estática(AEs). Na devida ordem, a EL possui em suas bases teóricas exigências em alguns temas como entre eles o gerenciamento de pacotes. Na AT são usadas ferramentas e *scripts* de execução (MAILEWA *et al.*, 2015). Estes testes têm como objetivo repetir ações predefinidas comparando os requisitos com o resultado real do teste (BHATT, 2017). Por último, a AEs de um programa é aquela realizada sem executá-lo (WICHMANN *et al.*, 1995; EGELE *et al.*, 2008). As técnicas de análise estática podem ser usadas para verificar os modelos de especificação, e de projeto de um sistema, para encontrar erros antes que uma versão executável do sistema esteja disponível (SOMMERVILLE, 2011). A pratica pensada para essa fase começa com a instalação de alguns pacotes através do *npm*, gerenciador de pacotes para *NodeJs*, entre eles o mo-

cha, que é uma estrutura de testes *JavaScript*, após a instalação será necessário que os participantes elaborem os casos de testes necessários e possam executá-los recebendo uma resposta positiva ou negativa. Após isso, deve-se instalar mais dois pacotes, agora referentes a AEs, o primeiro é o *Prettier*, que é um formatador de código e ou outro é o *ESLint* que é uma ferramenta de *linting* desenvolvida especificamente para *JavaScript*, de imediato será possível ver uma serie de erros no código e para solucioná-los os alunos devem editar alguns arquivos passados pelo condutor do RP. Para os objetivos da terceira fase espera-se que os alunos tenham conseguido, ter instalado todos os pacotes informados para construção dos casos de testes, ter construído o arquivo dos testes automatizados e ter obtido sucesso em todos os casos e ter instalado e configurado o *Prettier* e *ESLint*. Agora se debruçando sobre a categoria de Integração Contínua (CI). Fowler (2006) relata que CI é uma prática de desenvolvimento de software onde membros de uma equipe integram seu trabalho com frequência. Cada integração é verificada por uma compilação automatizada (incluindo testes) para detectar erros de integração o mais rápido possível. A orientação dos conceitos relativa a fase de CI trata sobre o processo de liberação frequente e confiável. Humble e Farley (2010) estabelecem alguns pontos que garantem um processo de liberação frequente e confiável, entre eles criar um processo repetível e confiável para entrega de software, automatizar tudo que for possível, manter tudo em um sistema de controle de versões e se um passo causa dor, execute-o com mais frequência e o quanto antes. A parte prática do CI será implementada com o auxilio da plataforma *GitHub Actions*, onde a mesma auxilia na construção de *pipelines* de CI/CD. Dentro do repositório que foi *forkado* pelos participantes existe uma aba chamada **Actions**, apos selecionar a mesma os participantes serão direcionados para uma outra pagina contendo *Workflows*, onde um respectivo a *NodeJs* tem de ser escolhido, posteriormente será mostrado um código inicial *YAML* que deve ser explicado pelo *Tech Lead* e complementado para atender as necessidades exigidas. Fornecidas as explicações e as devidas alterações os participantes devem somente executar um *commit* que o *pipelines* de CI que foi configurado será disparado. Para os objetivos dessa fase basta que os participantes tenham consegui construir o pipeline de CI e ao rodá-lo ter obtido cem porcento de sucesso. Chegando ao fim dessa fase, a Automação de Implantação(AIm), é o que permite implantar software em ambientes de teste e produção com apenas um clique. A automação é essencial para reduzir o risco de implantações de produção. Também é essencial para fornecer *feedback* rápido

sobre a qualidade do software, permitindo que as equipes realizem testes abrangentes o mais rápido possível após as alterações (CLOUD, 2023b). Falar de Automação de Implantação é falar de Implantação Contínua(CD), CD é a próxima etapa após a entrega contínua, em que você não está apenas criando constantemente um pacote implantável, mas na verdade está implantando-o de forma constante (PAIR, 2023). Para Valente (2020) as vantagens do CD são reduzir o tempo de entrega de novas funcionalidades, tornar novas *releases* (ou implantações) um "não-evento", reduzir o stress causado por *deadlines* e novas funcionalidades entram rapidamente em produção. Para a prática desta fase será utilizada o *Fly.io* que é uma plataforma global de distribuição de aplicativos. O primeiro passo será instalar a solução através do terminal, o segundo passo é se inscrever na plataforma, concluído esses dois passos o *tech lead* irá instruir os participantes pois algumas etapas devem ser feitos por linha de comando através do terminal. Após concluídas as instruções fornecidas, é necessário a criação de uma *secret key* que ira permitir a inclusão de mais um trecho de código *yml* que tem como responsabilidade concluir o *pipeline* de CD. Os objetivos que devem ser atingidos nesta fase são relativos a ter seguido todos os passos para criar a conta no *Fly.io* até a implantação da API no mesmo, ter adicionado os *scripts* necessários no nosso arquivo *yml* para as devidas configurações do CD e e efetuar mais um *commit* para acionar o pipeline de CI/CD e ver a implantação da API no *Fly.io*.

- **Fase #3**: é discorrido sobre a categoria de Monitoramento e Registro. O monitoramento geralmente é feito através de ferramentas que rastreiam as propriedades não funcionais dos aplicativos, como desempenho, disponibilidade, escalabilidade, resiliência e confiabilidade e entre as várias tarefas executadas pelas ferramentas de monitoramento há o Você construiu, você executa (*You build it, you run it*), Monitoramento contínuo (*Continuous monitoring*), Métricas (*Metrics*) e gerenciamento de *Log*, (LEITE *et al.*, 2019). A orientação dos conceitos presentes na fase de Monitoramento e Registro aborda a definição de Você construiu, você executa esse termo surgiu em 2006 cunhado por Werner Vogels ,CTO da Amazon, isso coloca os desenvolvedores em contato com a operação diária de seus Programas. Também os coloca em contato diário com o cliente. Esse ciclo de *feedback* do cliente é essencial para melhorar a qualidade do serviço (O'HANLON, 2006). Outro ponto abordado na orientação é o de Monitoramento Contínuo (CM), onde o CM em DevOps é o processo de identificação de ameaças às regras de segurança e conformidade de um ciclo e arquitetura de desenvolvimento de software (SANYAL, 2022). Além da

CM tem-se as Métricas, que como um conjunto de medidas de um processo ou produto de software, onde um produto de software pode ser visto como um objeto abstrato que evolui de uma instrução inicial para o sistema de software finalizado, incluindo o código-fonte e variadas formas de documentação produzidas durante o desenvolvimento (MILLS, 1988).Por ultimo nas orientações há os*logs*, que para Du e Li (2016) trata-se de registro de eventos, em ambiente computacional, é a documentação produzida de forma automática ou iniciada a partir de um sistema ou *host*, com data e hora de eventos relevantes para um sistema em particular. No trabalho de (CARNEIRO, 2022) o mesmo relata que ainda podem ser incluídos nos *logs* o nome do *host*, tipo de *log*, endereço IP e endereço MAC. A prática relacionada a presente fase começa com a instalação da imagem do *Sonarqube*, após a instalação deve-se executar o contêiner e acessar o localhost:9000 para que se possa fazer o primeiro acesso e configurar o *admin*. Posterior a isso deve-se baixar e instalar o *SonarScanner*, que nada mais é do que uma foram genérica de avaliar o das aplicações, voltando ao Sonar deve-se criar um projeto manual, nomeá-lo, escolher que tipo de código vai ser analisado, após isso irá ser gerado um *token* que deve ser usado no terminal indiciando o caminho relativo do projeto e quando executado fará com que o *Scanner* analise os arquivos e suba para o *Sonarqube*. A outra prática se trata da manipulação dos pacotes *body-parser e morgan-body*, em seguida os participantes devem aplicar as constantes no código *NodeJs* responsável pelas requisições da API, posteriormente a isso deve-se criar uma pasta na raiz do projeto intitulada de *logs* onde o arquivo de *log* possibilitando a visualização de algumas informações que foram gravadas no momento da chamada dos *endpoints*. Os objetivos atribuídos a sexta fase são, a corretude na implantação dos pacotes, bem como a obtenção de um arquivo de *log* contendo informações iniciais a respeito do tipo da requisição e os dados trafegados através dos mesmos, além de eventuais erros de chamadas contidos na aplicação.

APÊNDICE B – TABELA CONTENDO INFORMAÇÕES SOBRE AS CARACTERIZAÇÕES DOS PARTICIPANTES E EXPERIMENTOS

Quadro 3 – Caracterização dos participantes e experimentos.

ID do Experimento	ID do Participante	Gênero dos Participantes	Curso dos Participantes	Nível de conhecimento sobre DevOps
E1	P1	Homem	CC	7
	P2	Homem	SI	7
	P3	Homem	SI	7
	P4	Mulher	SI	7
	P5	Homem	CC	8
	P6	Homem	SI	8
	P7	Homem	SI	9
	P8	Homem	SI	8
	P9	Homem	CC	2
	P10	Homem	CC	7
E2	P11	Homem	CC	8
	P12	Homem	SI	8
	P13	Mulher	SI	5
	P14	Homem	CC	2
	P15	Mulher	SI	6
	P16	Mulher	SI	4
	P17	Mulher	SI	7
	P18	Mulher	SI	7
	P19	Homem	SI	5
E3	P20	Homem	CC	5
	P21	Homem	CC	4
	P22	Homem	CC	5
	P23	Homem	CC	1
	P24	Mulher	SI	1
	P25	Homem	SI	1
	P26	Homem	CC	1
	P27	Mulher	SI	1
	P28	Homem	SI	3
	P29	Mulher	SI	2
	P30	Homem	CC	6

Fonte: Autoria própria (2022).

APÊNDICE C – *CODEBOOK* UTILIZADO PARA ANÁLISE DOS DADOS

Temas	Subtemas	Participantes
Experiências vivenciadas em cada fase	Aprendizagem Cooperativa	P8,P10,P18,P22, P23,P26,P27,P29
	Aprendizagem envolvendo ferramentas exploradas na indústria	P4,P5,P8,P19, P12,P28,P29
	Desafios e pontos positivos	P4,P15,P14,P23,P27, P20,P22,P27
Aspectos Positivos	Explorando Práticas e Ferramentas	P1,P2,P6,P14, P28,P30
	Aprendizagem em pares	P8,P11
	Aceitação do RP.	P6,P12,P13,P16,P15,18,P24,P28
Perspectivas de Melhorias	Gestão de Tempo	P14,P15,P16,P19
	Otimização da Dinâmica	P13,P19,P20,P28

Fonte: Autoria própria (2024).