FAST

FORMAÇÃO ACELERADA EM SOLUÇÕES DE TECHDESIGN

C.E.S.A.R

Metodologias Ágeis e BDD

Todos os direitos reservados.
Este material, ou qualquer parte dele, não pode ser reproduzido, divulgado ou usado de forma alguma sem autorização escrita.

© CESAR 2025 | Todos os Direitos Reservados

O que veremos hoje?

- Conteúdo 01 - Entrega Contínua e o Papel do QA
- Conteúdo 02 - Critérios de Aceite e Definição de Pronto
- Conteúdo 03 - Introdução ao BDD e Gherkin
- Conteúdo 04 - Diferenças entre BDD, TDD e ATDD

© CESAR 2025 / Todos os Direitos Reservados

Bora revisar?

1. Papel do QA no ágil: QA atua desde o início do ciclo, definindo critérios de aceitação e garantindo qualidade contínua.

2. Colaboração e melhoria contínua: Qualidade é responsabilidade de todo o time. QA facilita feeback e antecipa riscos.

3. Técnicas de teste: Uso de valor limite e particionamento de equivalência.

Entrega Contínua e o Papel do QA

Metodologias Ágeis e BDD

Aula 2

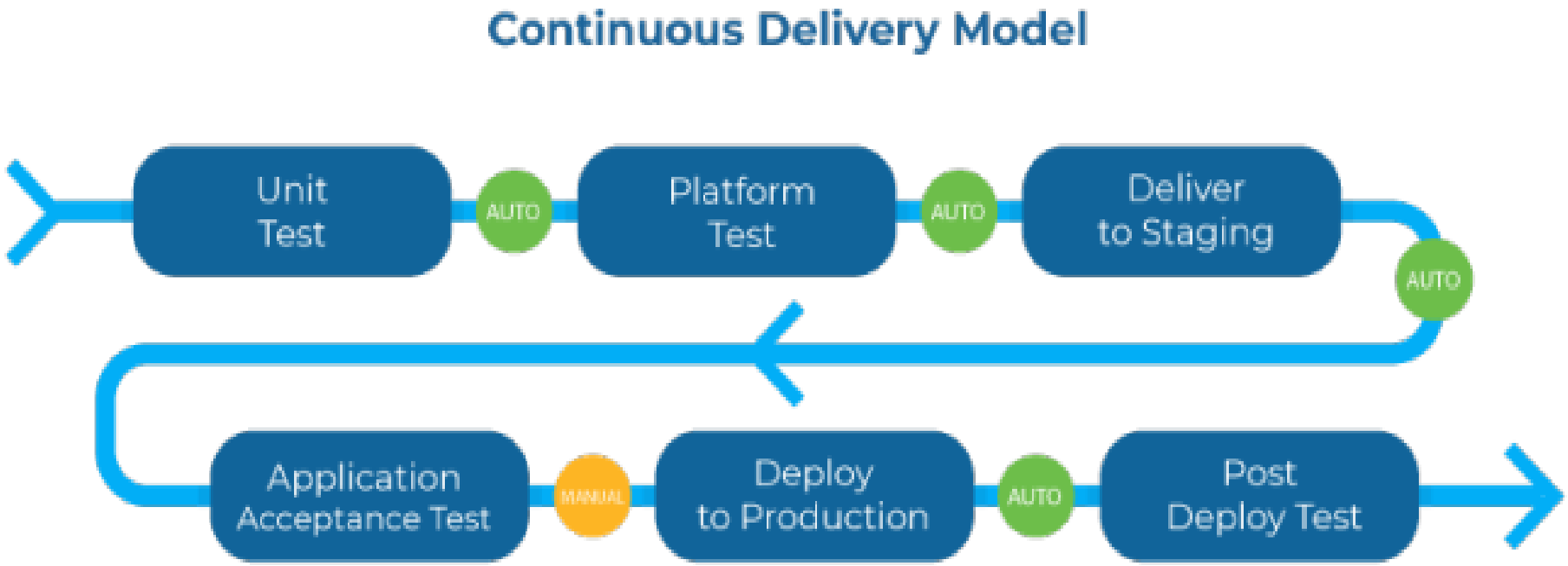
© CESAR 2025 / Todos os Direitos Reservados

O que é Entrega Contínua?

- **Entrega contínua** (Continuous Delivery - CD) é uma prática de desenvolvimento de software que visa **automatizar o processo de lançamento de software**, garantindo que as alterações de código sejam preparadas para produção e possam ser liberadas a qualquer momento.
- A entrega contínua permite que as equipes de desenvolvimento automatizam o processo que move o software ao longo do ciclo de vida de desenvolvimento de software.
- As equipes, de desenvolvimento de software, produzem um entregável em ciclos curtos, garantindo que o software possa ser lançado com segurança a **qualquer momento**.

© CESAR 2025 | Todos os Direitos Reservados

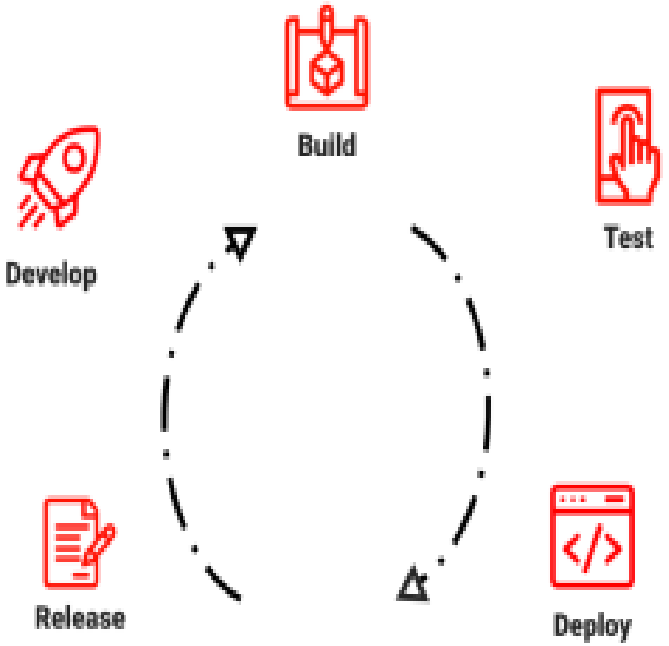
O que é Entrega Contínua?



Entrega Contínua

A entrega contínua pode oferecer muitos benefícios:

- Reduzir o tempo de implementação por meio de testes contínuos e desenvolvimento.
- Detectar erros mais rapidamente.
- Diminuir os custos associados ao desenvolvimento de software tradicional.
- Dimensionar o desenvolvimento de software com base no tamanho do projeto.
- Implementar o código automaticamente em cada fase do ciclo de desenvolvimento.
- Entregar atualizações mais rapidamente.



O papel do QA (Quality Assurance) na entrega contínua é essencial para garantir que tudo esteja funcionando corretamente. O QA cria testes que são executados na esteira de CI/CD e colabora com os desenvolvedores para identificar e resolver problemas. Isso ajuda o time a entregar um software de alta qualidade e confiável.

Entrega Contínua e QA

- O Continuous Delivery (CD) tem demonstrado benefícios significativos na prática, especialmente em termos de qualidade do software, produtividade e redução de riscos.

- O relatório *"State of Continuous Delivery 2023"* destaca que **84%** dos desenvolvedores agora participam de atividades relacionadas ao DevOps, o que inclui a adoção de **práticas de CD**.

© CESAR 2025 | Todos os Direitos Reservados

Papel do QA na Entrega Contínua

1 Definição de critérios de qualidade

- Participar do refinamento das histórias.
- Ajudar a definir critérios de aceite claros e testáveis.

2 Construção de testes automatizados

- Criar e manter testes funcionais e de regressão.
- Automatizar testes de integração com pipelines.

3 Validação de pipeline de testes

- Garantir que o pipeline execute testes de forma confiável.
- Validar resultados em ambientes de pré-produção.

4 Prevenção e detecção precoce de falhas

- Acompanhar novas features desde o início.
- Colaborar com Devs para antecipar riscos e bugs.

5 Monitoramento e feedback contínuo

- Acompanhar métricas de qualidade (ex.: coverage, flakiness).
- Alimentar ciclos de feedback e melhoria contínua no time.

6 Fomento à cultura de qualidade no time

- Promover práticas de qualidade como BDD, TDD e Pair Testing
- Estimular o mindset de qualidade compartilhada entre Devs, QA, PO e demais papéis

Fonte: ISQTB | Certified Tester Syllabus

Impactos da Entrega Contínua no papel do QA

1. **QA não "fecha" uma fase — ele atua durante todo o fluxo**
   → Deixa de ser um "gatekeeper" que testa no final.

2. **Prevenção em vez de detecção**
   → QA ajuda a antecipar problemas e riscos antes mesmo da codificação (ex.: revisão de critérios, análise de impacto).

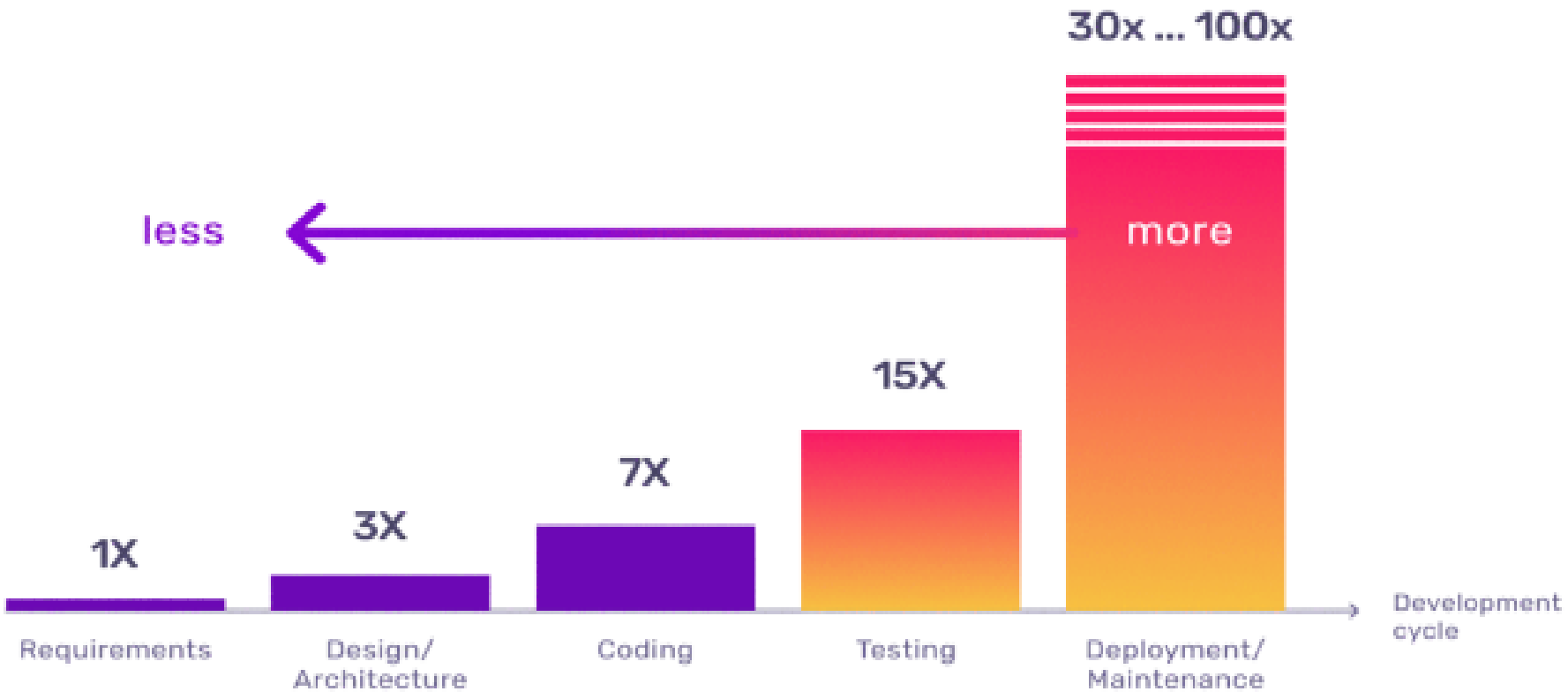
3. **Automação é essencial**
   → Testes automatizados precisam acompanhar a cadência de deploys.
   → QA colabora com Devs para garantir cobertura automatizada eficaz.

4. **Feedback rápido e contínuo**
   → O QA gera informações frequentes sobre qualidade, ajudando o time a melhorar a cada entrega.

5. **Qualidade é do time, não só do QA**
   → QA assume papel de facilitador da qualidade no time.

© CESAR 2025 | Todos os Direitos Reservados

Cost of Defects



The more time we save your team, the more time they have to find bugs sooner.

That Saves Money

CESAR 2025 | Todos os Direitos Reservados

Vamos conversar!

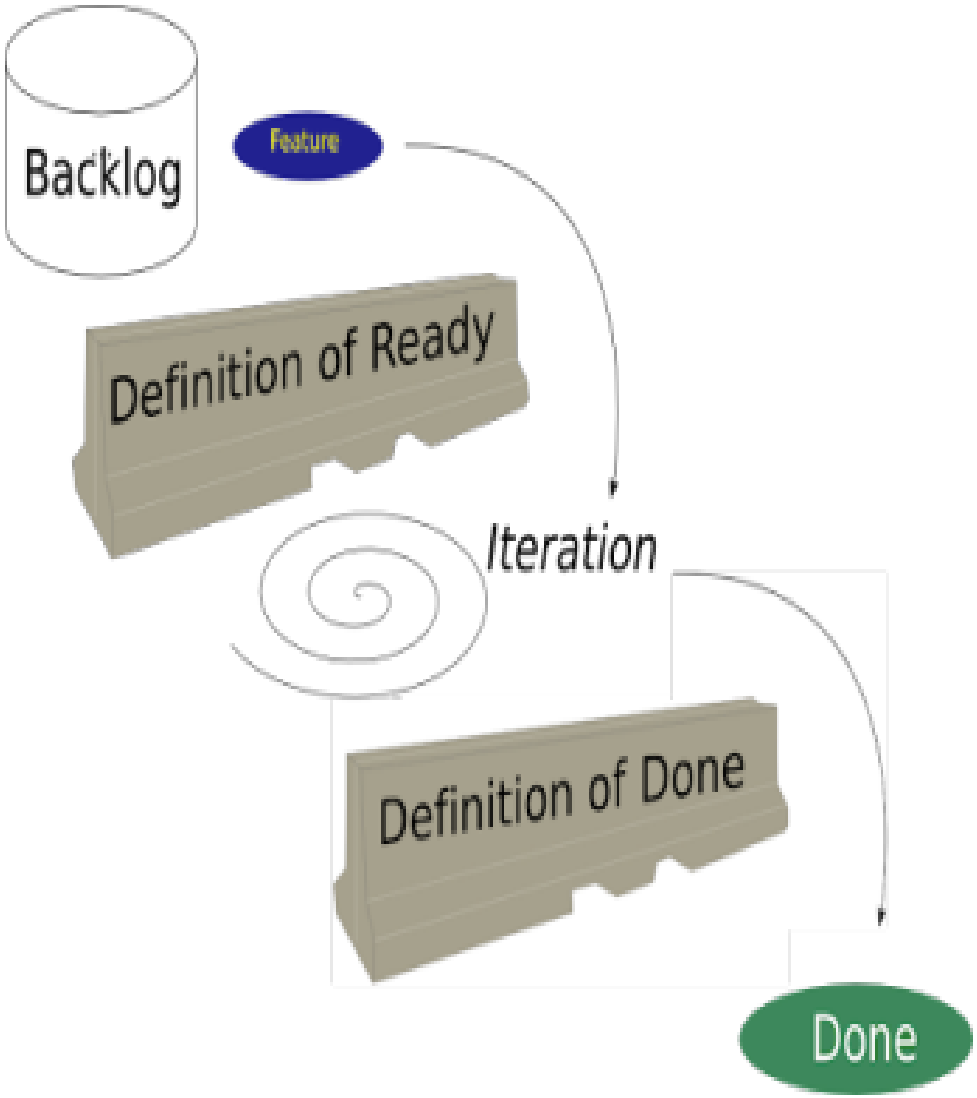
1. O que você acha que muda no trabalho do QA quando as entregas de software acontecem toda semana ou até todo dia?
2. Você imagina que é fácil ou difícil garantir a qualidade quando o time entrega mais rápido? Por quê?
3. Como você acha que o QA pode ajudar o time a encontrar problemas o mais cedo possível?
4. Se o time entrega com frequência, o que é mais importante: ter muitos testes manuais ou automatizados? Por quê?
5. Como o QA pode ajudar a evitar que bugs cheguem até o cliente mesmo com entregas rápidas?

© CESAR 2025 / Todos os Direitos Reservados

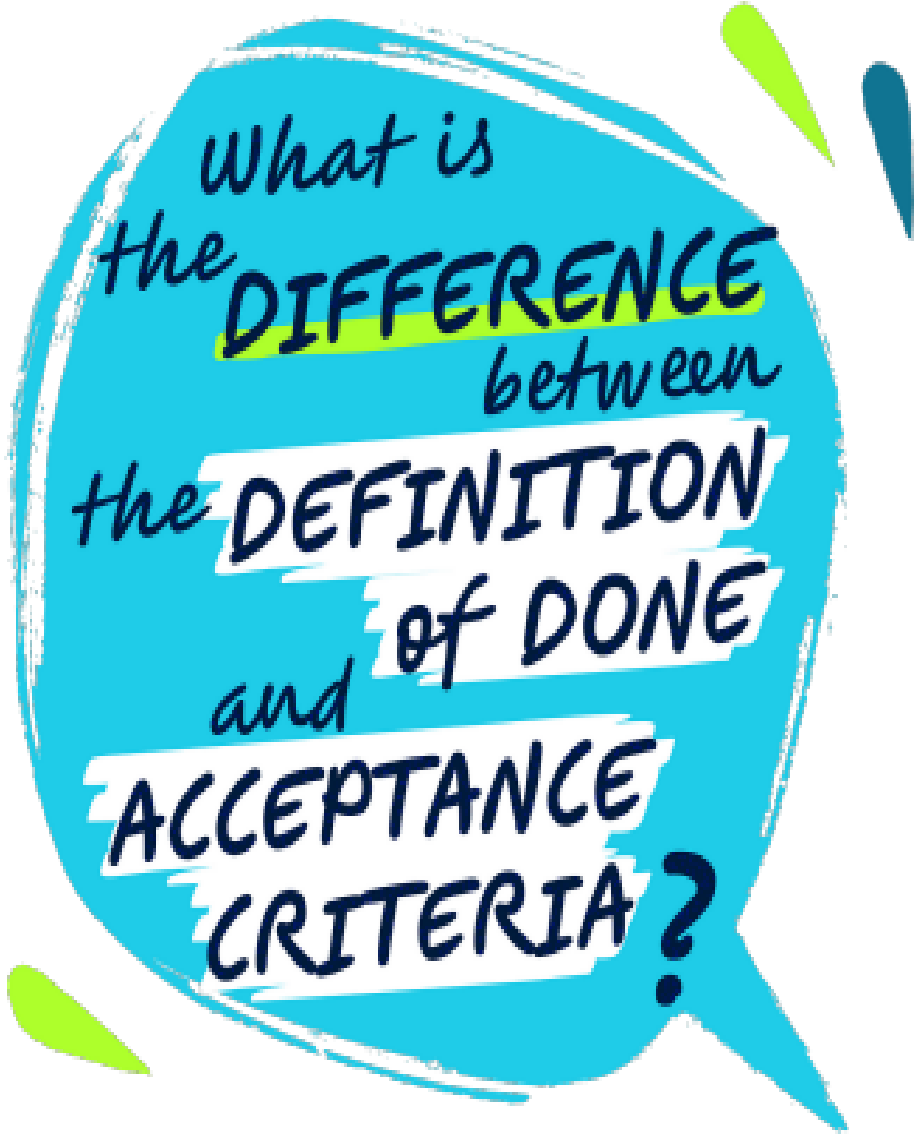
Definição de "pronto" (DoR)

- É um conjunto de critérios que determinam se um item do backlog está pronto para ser trabalhado pelo time de desenvolvimento.
- A DoR ajuda a garantir que o time tenha todas as informações e recursos necessários para iniciar o trabalho em um item, evitando retrabalho e atrasos.
- A DoR garante que o time possa iniciar o trabalho em um item com confiança e eficiência.
- Sem que todos os critérios estabelecidos pelo DoR sejam cumpridos, o recomendado é que a equipe não comece uma determinada tarefa.

O Definition of Ready (DoR), também chamado de "Definição de Pronto", é um elemento importante da metodologia Scrum para definir se uma tarefa está ou não pronta para ser executada pelo time de desenvolvimento.



© CESAR 2025 | Todos os Direitos Reservados



© CESAR 2025 | Todos os Direitos Reservados

© CESAR 2025 | Todos os Direitos Reservados

Definição de "feito" (DoD)

É uma lista de critérios gerais e padronizados que se aplicam a todas as histórias ou itens de backlog do time.

- Garante que tudo que o time chama de “pronto” tenha um nível mínimo de qualidade.
- Inclui práticas como: código revisado, testes automatizados passando, documentação atualizada, deploy em ambiente de homologação etc.

- Exemplo de DoD:
  - Código revisado por outro desenvolvedor.
  - Cobertura mínima de 80% em testes automatizados.
  - A história foi validada em ambiente de testes.

© CESAR 2025 | Todos os Direitos Reservados

Definição de "AC"

São específicos para cada item do backlog, geralmente definidos junto com a user story.

- Descrevem **as condições que a funcionalidade precisa cumprir para que o Product Owner aceite a entrega**.
- Funcionam como "regras de aceite" para aquela história ou tarefa específica.

- Exemplo:
  - Para a história "Usuário faz login", ACs podem ser:
  - O sistema exibe mensagem de erro para senha incorreta.
  - O login é bloqueado após 5 tentativas falhas.

Introdução ao BDD e Gherkin

Metodologias Ágeis e BDD

Aula 2

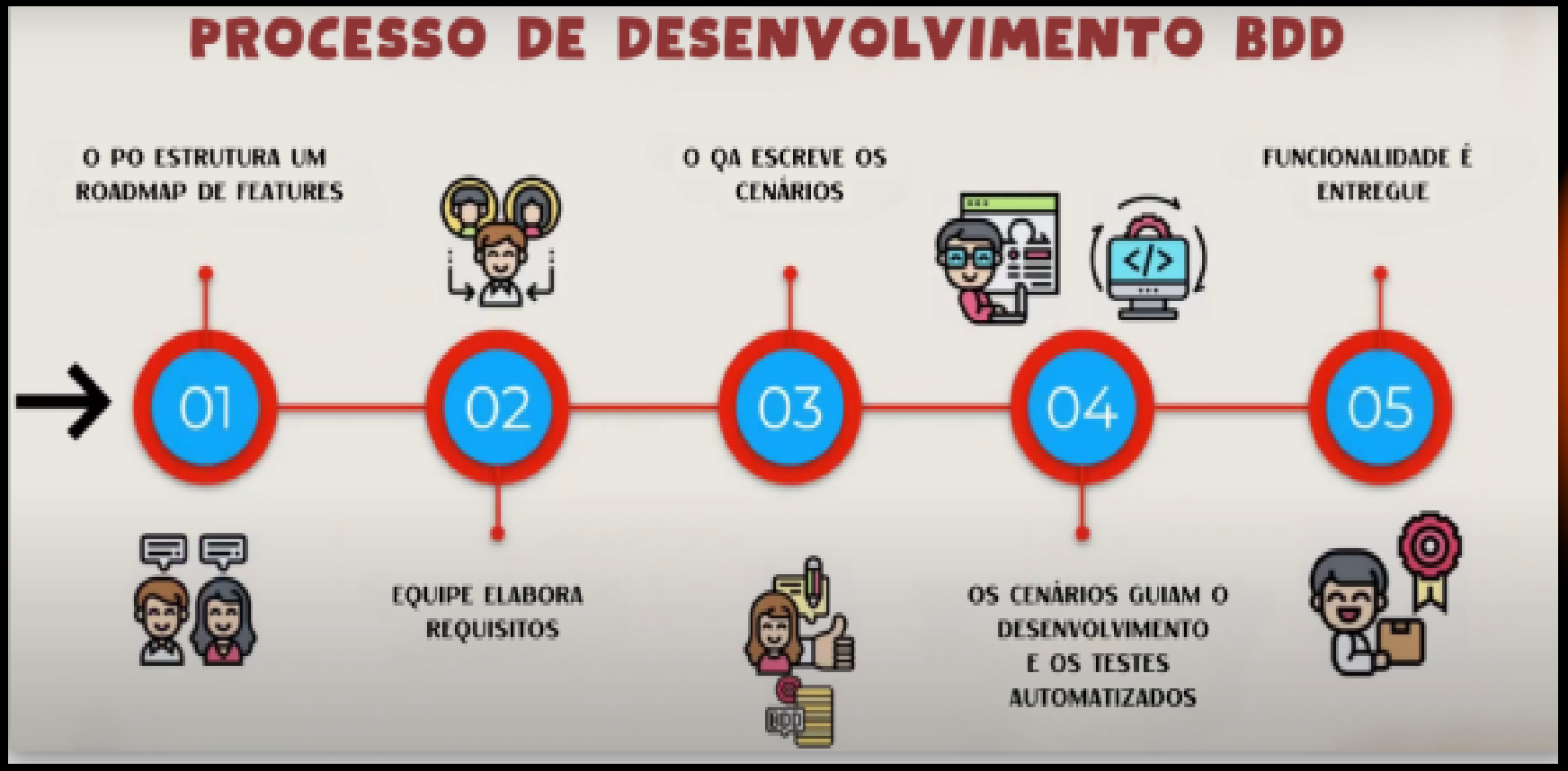
© CESAR 2025 | Todos os Direitos Reservados

O que é BDD?

- BDD (Behavior Driven Development) é uma **forma de escrever casos de uso** em linguagem simples, que todos no time (Dev, QA, PO) conseguem entender.
- Enfatiza a **colaboração entre equipes** de desenvolvimento, testes e negócios, focando na definição do comportamento esperado de um sistema através de exemplos concretos.
- Utiliza uma **linguagem natural** para descrever o comportamento desejado de uma funcionalidade.
- Em vez de focar em partes individuais, o BDD aborda o comportamento do sistema como um todo, utilizando cenários baseados em exemplos para ilustrar como o sistema deve funcionar em diferentes situações.

Valor do BDD como ferramenta de colaboração

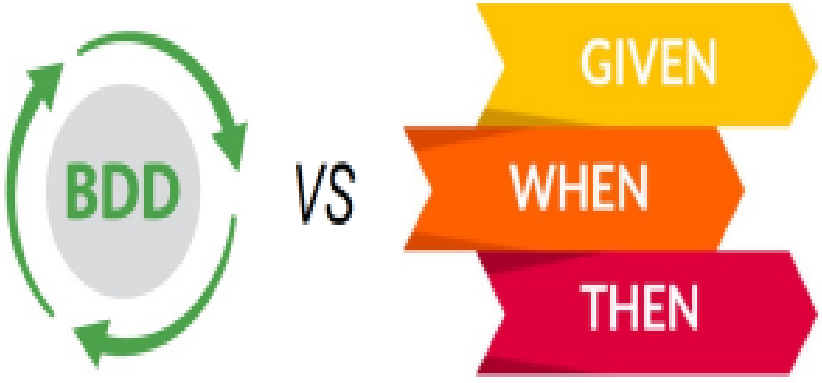
- Comunicação clara e concisa.
- Entendimento compartilhado.
- Foco no valor para o usuário.
- Redução de erros e retrabalho
- Feedback rápido e contínuo.
- Maior envolvimento das partes interessadas.
- Documentação viva



© CESAR 2025 | Todos os Direitos Reservados

BDD + Gherkin

- Gherkin é uma linguagem de especificação legível por humanos usada no Desenvolvimento Orientado a Comportamento (BDD) para descrever casos de teste de forma estruturada e clara.
- Sua função é padronizar a forma de descrever especificações de cenários, baseado na regra de negócio.
- Ele é escrito em forma de “steps” (ou “passos”), os quais especificam cada etapa de interação do usuário com o sistema a ser testado.
- No Gherkin existem “keywords” (ou “palavras-chave”) a serem utilizados para especificar a forma como cada step interage com o sistema.



© CESAR 2025 | Todos os Direitos Reservados

Padrão Gherkin

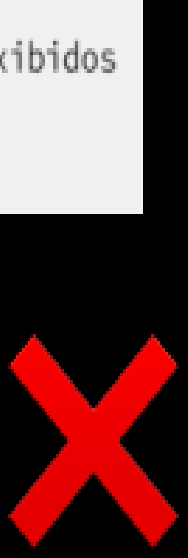
Essas palavras-chave ajudam a organizar os testes em formato de narrativa, facilitando a compreensão por todos os membros da equipe, incluindo desenvolvedores, testadores e stakeholders.

- **Given** (pt: Dado)
  - Utilizado para especificar uma pré condição.
  - Normalmente vem escrito no passado.
- **When** (pt: Quando)
  - Utilizado para definir quando será executada uma ação que se espera do sistema.
  - Este passo vem escrito no presente.
- **Then** (pt: Então)
  - Valida se o esperado aconteceu.
  - Normalmente vem escrito na forma de futuro próximo.
- **And** (pt: E)
  - Caso seja necessário mais uma interação com o sistema para complementar um fluxo.
- **But** (pt: Mas)
  - No geral serve a mesma funcionalidade do “And”, porém é normalmente utilizado após uma validação negativa depois do “Then”.

Exemplos:
Dado que o usuário tem acesso ao sistema
Dado que o usuário tem acesso à área administrativa
Dado que o usuário está logado

Exemplos:
Quando o usuário realiza login
Quando o usuário cadastra um novo produto
Quando o usuário altera uma publicação

Exemplos:
Então os dados do usuário logado devem ser exibidos
Então o produto é alterado com sucesso
Então a publicação é cadastrada com sucesso



Dado que o usuário está logado
Quando o usuário exclui uma postagem no blog
Então a postagem é excluída com sucesso

Dado que acesso ao site
Quando estou logado
E busco um produto na página principal
E seleciono um produto
Então deve ser direcionado para a página do produto
E deve ter o navigation
E deve ter a opção de zoom
E deve ter as imagens do produto
E deve ter a descrição do produto
E deve ter o título do produto
E deve ter a marca do produto
E deve ter a avaliação do produto
E deve ter o preço do produto
E deve ter Botão “Adicionar no Carrinho”

© CESAR 2025 | Todos os Direitos Reservados

1. Procure sempre escrever os cenários em terceira

✅
```
Forma recomendada:
Dado que o usuário está logado
Quando o usuário exclui uma postagem no blog
Então a postagem é excluída com sucesso
```

❌
```
Forma não recomendada:
Dado eu estou logado
Quando eu excluo uma postagem no blog
Então devo ver que a postagem é excluída com sucesso
```

2. Escreva cenários declarativos, não imperativos

✅
```
Forma recomendada:
Dado que o usuário possui credenciais válidas
Quando o usuário faz o login
Então ele tem acesso aos seus dados
```

❌
```
Forma não recomendada:
Dado que o usuário está na tela de login
Quando ele digita o usuário "admin"
E digita a senha "admin"
E clica em "Logar"
Então o usuário deve visualizar a sua tela inicial com os seus dados
E deve ver a mensagem "Bem-vindo Admin"
```

3. Insira uma narrativa

```
Exemplos:
Cenário: Exibir dados pessoais do usuário
Cenário: Cadastrar um novo produto com sucesso
Cenário: Realizar login do usuário com sucesso
```

```
Forma recomendada:
Cenário: Cadastrar um novo produto com sucesso
  Dado que o usuário tem permissão para cadastrar produtos
  Quando ele preenche as informações corretamente
  E envia os dados
  Então o produto deve ser cadastrado com sucesso

Forma não recomendada:
Cenário: Cadastrar um novo produto com sucesso
  Dado que o usuário tem permissão para cadastrar produtos
  Quando ele preenche as informações corretamente e envia os dados
  Então o produto deve ser cadastrado com sucesso
```

Diferenças entre BDD, TDD e ATDD

Metodologias Ágeis e BDD

Aula 2

© CESAR 2025 | Todos os Direitos Reservados

TDD

- O TDD, ou Desenvolvimento Orientado a Testes, é uma técnica em que os testes são escritos antes da implementação do código.
- A ideia principal é garantir que o código atenda aos requisitos definidos pelo teste desde o início.
- O TDD ajuda a criar um código mais limpo e a evitar a implementação de funcionalidades desnecessárias.

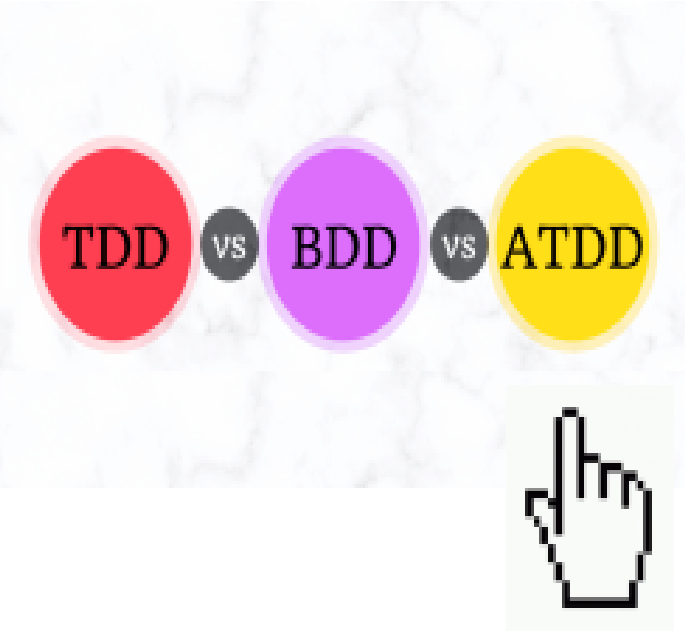
Como funciona:

1. Escreva um teste que falhe (porque a funcionalidade ainda não existe).
2. Implemente o código mínimo necessário para passar no teste.
3. Refatore o código mantendo o teste verde.

© CESAR 2025 | Todos os Direitos Reservados

ATDD

- O ATDD é uma variação do TDD que foca nos testes de aceitação, garantindo que as funcionalidades implementadas estejam de acordo com as expectativas do cliente.
- Nesta abordagem, os critérios de aceitação são discutidos e definidos antes do desenvolvimento, e esses critérios são transformados em testes automatizados.
- O ATDD garante que o código esteja alinhado com os objetivos do negócio e expectativas dos usuários.



Como funciona:

1. Stakeholders (como clientes, desenvolvedores e testadores) colaboram para definir os critérios de aceitação.
2. Os testes de aceitação são escritos antes do código.
3. O código é desenvolvido para passar nesses testes.

Característica	TDD (Test-Driven Development)	ATDD (Acceptance Test-Driven Development)	BDD (Behavior-Driven Development)
Foco	Código funcionando conforme testes	Testes de aceitação definidos pelo cliente	Comportamento do sistema (linguagem comum)
Colaboração	Principalmente entre desenvolvedores	Entre desenvolvedores, testadores e clientes	Entre toda a equipe, incluindo stakeholders
Ferramentas	JUnit, NUnit, etc.	JUnit, Selenium, etc.	Cucumber, SpecFlow, etc.
Exemplo	Escrever teste para uma função	Escrever teste de aceitação para uma funcionalidade	Escrever cenários de comportamento (Given-When-Then)

© CESAR 2025 / Todos os Direitos Reservados

Vamos praticar!

Você foi convidado(a) a colaborar como QA em um time que está desenvolvendo uma nova funcionalidade de recuperação de senha para um sistema de e-commerce.

Requisitos básicos:

✅ O usuário deve poder solicitar a recuperação de senha por e-mail.

✅ Um e-mail com link de redefinição deve ser enviado.

✅ O link expira em 24h.

✅ Se o e-mail não existir, uma mensagem de erro deve ser exibida.

📝 **Tarefa:**

1️⃣ Liste **3 critérios de aceite** para essa funcionalidade.

2️⃣ Escreva **2 cenários em Gherkin** (Given–When–Then), representando comportamentos esperados.

© CESAR 2025 | Todos os Direitos Reservados

Possível solução...

Critérios de Aceite

1. **Solicitação de redefinição de senha**: O usuário, ao informar um e-mail cadastrado, deve receber uma mensagem de sucesso na interface, indicando que o link de redefinição foi enviado.

2. **Envio do e-mail**: Um e-mail contendo um link único e seguro para a página de redefinição de senha deve ser enviado para o endereço de e-mail informado.

3. **Expiração do link**: O link de redefinição de senha deve ser válido por um período de 24 horas a partir do momento em que foi gerado. Se o usuário tentar usá-lo após esse período, o link deve ser considerado expirado e uma mensagem de erro apropriada deve ser exibida.

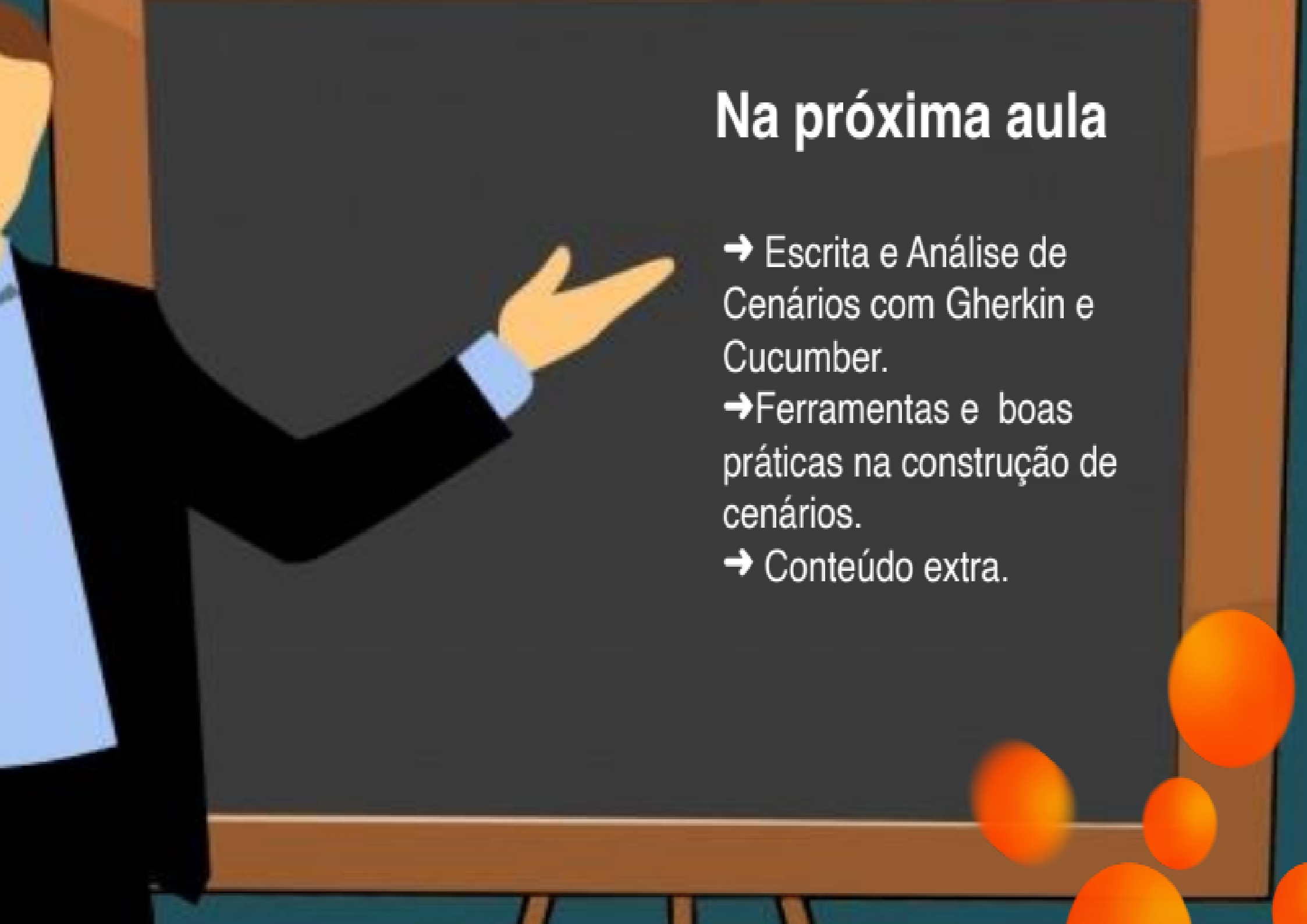
© CESAR 2025 | Todos os Direitos Reservados

Possível solução...

Cenários (Gherkin)

```
Cenário: Redefinição de senha com sucesso
Dado que estou na página de login
E desejo redefinir minha senha
Quando informo um e-mail de usuário válido
E recebo o e-mail com o link de redefinição
E acesso o link em menos de 24 horas
E preencho e confirmo a nova senha
Então a senha é atualizada com sucesso
E sou redirecionado para a página de login
E consigo entrar com a nova senha
```

```
Cenário: E-mail não cadastrado
Dado que estou na página de redefinição de senha
Quando informo um e-mail que não existe no sistema
Então é exibida a mensagem "E-mail não encontrado"
E nenhum e-mail de redefinição é enviado
```


Na próxima aula
➜ Escrita e Análise de Cenários com Gherkin e Cucumber.
➜Ferramentas e boas práticas na construção de cenários.
➜ Conteúdo extra.

DÚVIDAS?!

:)

C.E.S.A.R

MUITO OBRIGADO!

© CESAR | Todos os direitos reservados