comunidade TESTER GLOBAL

Esquema visual das aulas

v 2.0
viniciuspessoni.com

#CTG aula 1.1

Programa

é uma sequência de instruções que instrui o computador como realizar uma tarefa

Software

é composto pelo programa, os dados para executar ele e a documentação de como ele funciona

Sistema

é um conjunto de diversas partes interagentes e interdependentes que colaboram para realizar uma ação

Apesar de usarmos esses termos de forma intercambiável, eles são diferentes.
O teste de software é aplicado em cada um deles de forma distinta.

Estamos fazendo o produto certo?

Estamos fazendo certo o produto?

V&V= validação e verificação

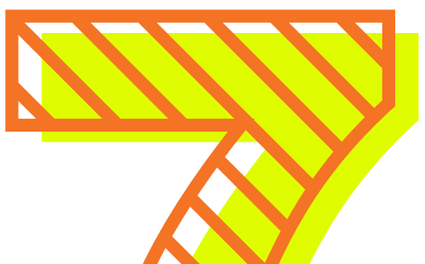
TODO SOFTWARE TEM DEFEITO

e por isso TESTAMOS:
- pra ter um produto com mais confiança
- diminuir problemas de produção
- reduzir custos

QUAIS ÁREAS DE ATUAÇÃO

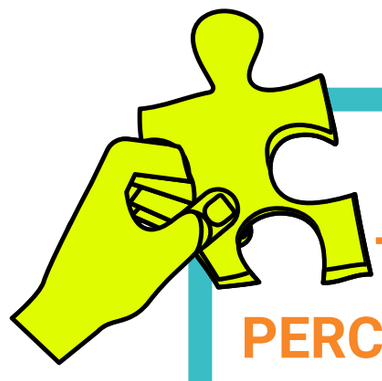
- FRONT END
- BACK END
- DADOS
- MOBILE
- FULL STACK

[foque em experimentar todas pra saber qual você se identifica. Quanto mais variedades saber melhores oportunidades terá]



Princípios do Teste

1. Mostra presença de defeitos
2. Teste exaustivo é impossível
3. Teste antecipado
4. Agrupamento de defeitos
5. Paradoxo do pesticida
6. Teste depende do contexto
7. A ilusão da ausência de erro



DESAFIO 01

TREINAR O SEU OLHAR E PERCEPÇÃO PARA OS ERROS E BUGS COTIDIANOS

Atentem-se no seu dia a dia, nos programas, sites, app que você usa. Quando ocorre problema, erro ou se comporta diferente do que deveria.

Dê print de pelo menos 3 deles essa semana, poste no insta e me marque

processos de software

elementos da qualidade

versionadores

mínimo uma linguagem de programação

arquitetura de sistemas

+ inglês

O QUE EU PRECISO SABER?

#CTG aula 1.2

Bugs em produção: de quem é a responsabilidade

Em equipe ágeis a responsabilidade é de TODOS:

- P.O (product owner)
- Developer
- Tester

Pois todo sotware tem defeito e erros. Toda equipe trabalha junto para que sejam menores possíveis

DO TESTE

Não há tempo para o teste exaustivo;as vezes nem para o teste em si

Muitas combinações de entrada;dificuldade de cobertura do software todo

Dificuldade em determinar os resultados esperados para cada caso de teste

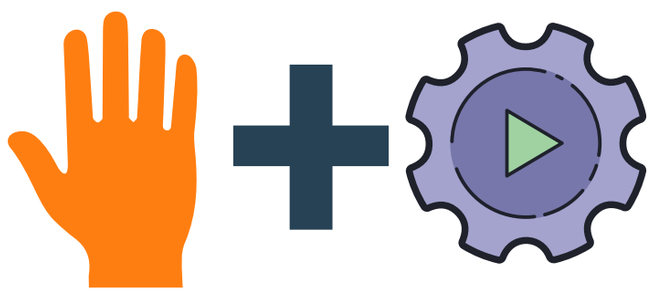
DESAFIOS

Requisitos inexistentes ou mudam rapidamente.

Não há treinamento no processo de teste OU falta ferramentas de apoio

Gerentes desconhecem teste ou não preocupam com qualidade

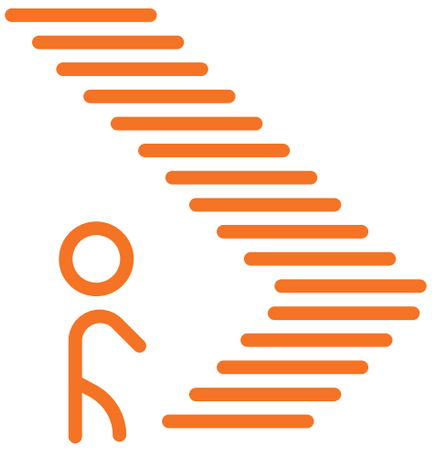
Como o tester atual trabalha?



MANUAL + AUTOMAÇÃO
em todas as fases do processo:

- REQUISITOS:
ajuda a revisar e descrever os requisitos
- PROJETO:
ajuda a discutir e construir a arquitetura do software
-DESENVOLVIMENTO:
desenvolde os testes, faz programação em pares e code reviews
-TESTE:
realiza os testes
-DEPLOY EM PROD:
coloca ou ajuda a oclocar o código em produção e monitora defeitos

DESAFIO 02



Criar seu plano de desenvolvimento

Identificar pontos de melhoria e definir metas com atividades para cada meta

Sugiro o Uso do método SMART, você pode saber mais sobre ele no meu livro " Carreira Internacional em TI"

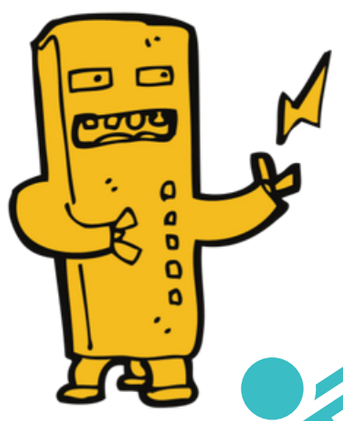
#CTG aula 2.1

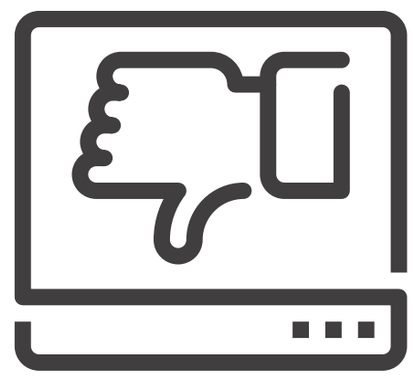
A Engenharia de Software surgiu como uma tentativa de solução para a Crise do Software

Imprecisão nas estimativas de prazo e de custo

Insatisfação do cliente com o sistema entregue

não atende os requisitos





baixa qualidade

Difícil gerenciar, manter e evoluir

FERRAMENTAS + MÉTODOS + PROCESSO = permitem produzir software de alta QUALIDADE

é uma área e que visa garantir bons produtos a partir de bons processos

ISO/IEC 25010:2011 SquaRE

Adequação funcional
Integridade Funcional
Correção Funcional
Pertinência Funcional

Eficiência de Desempenho
Comportamento Temporal
Utilização de Recursos
Capacidade

Compatibilidade
Coêxistência
Interoperabilidade

Usabilidade
dequação no reconhecimento
Proteção contra erros | Aprendizagem
Operacionalidade | Acessibilidade
Estética da Interface

Confiabilidade
Maturidade | Disponibilidade | Tolerância a Falhas | Capacidade de Recuperação

Segurança
Confidencialidade | Integridade
Não repúdio | Reponsabilização

Manutenibilidade
Modularidade | Reutilização
Facilidade de Análise
Mutabilidade | Testabilidade

Portabilidade
Adaptabilidade | Facilidade de Instalação
Capacidade de Substituição

Qual o principal objetivo quando desenvolvemos software?

Atender os requisitos do usuário!

MITOS DE SOFTWARE

são crenças errôneas sobre o software e sobre os processos de software que existem há vários anos e por algum motivo ainda são acreditados; são afirmações que parecem ser verdadeiras mas não são, apesar de conter alguns elementos verdadeiros

DESAFIO 03

Escolher um software que você usa e analisar as características da norma SQUARE nele

#CTG aula 2.2

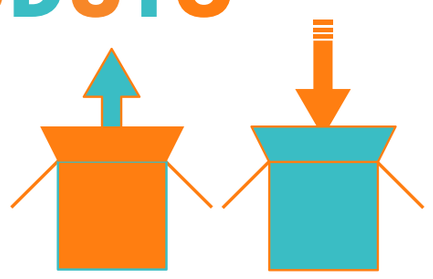
PROCESSO

permitem um desenvolvimento racional e organizado do software

ATIVIDADE + RESULTADOS = PRODUTO

interdependente

com responsáveis



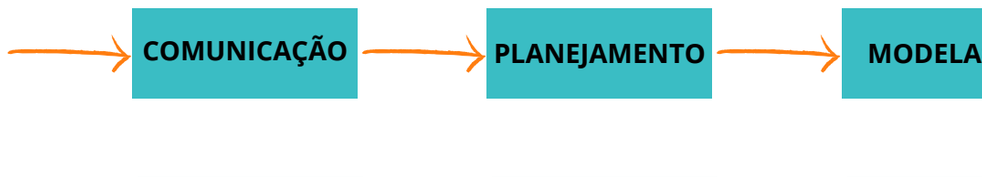
com entradas e saídas definidas

com um fluxo a ser seguido

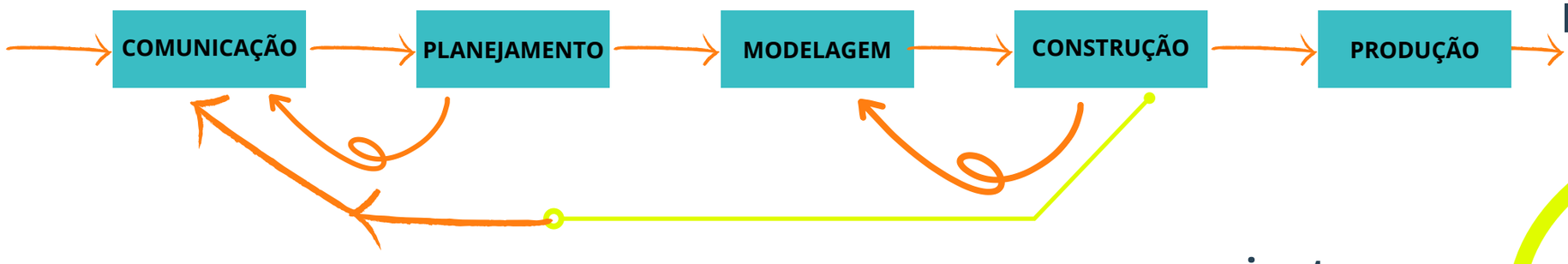
Modelo de processo = ciclo de vida do software

fases do ciclo de vida

a forma que as atividades são executadas (ou seja, seu fluxo) pode ser:

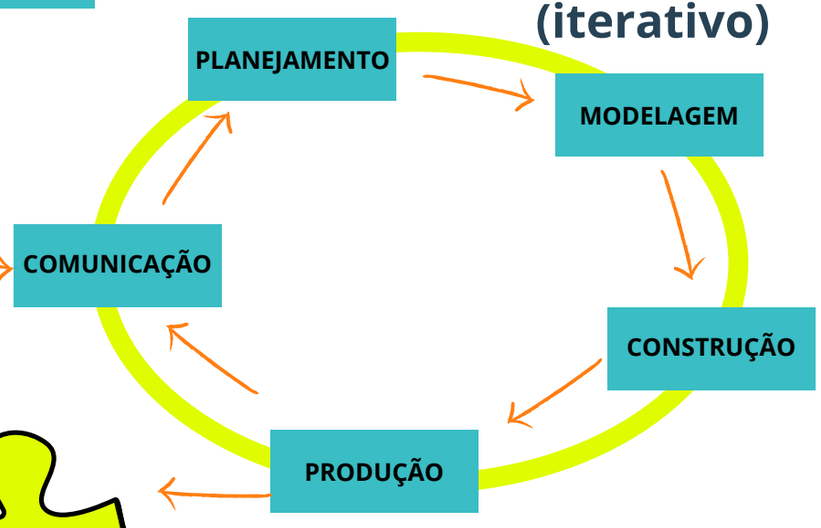


Sequencial (linear)



Incremental

Evolucionário (iterativo)

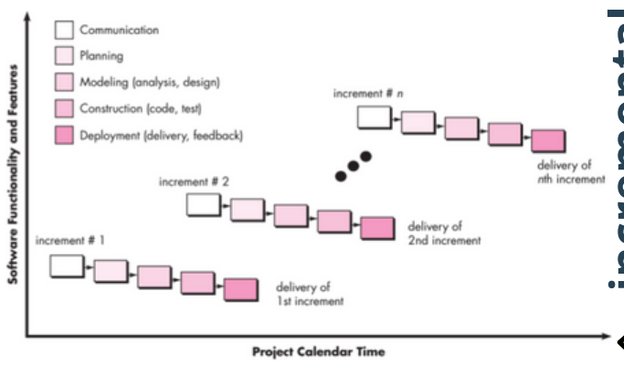


- Modelos Prescritivos

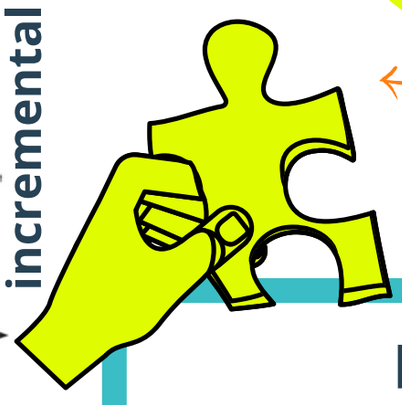
porque prescrevem um conjunto de elementos de processo e também descrevem o fluxo

cascata

modelo em V

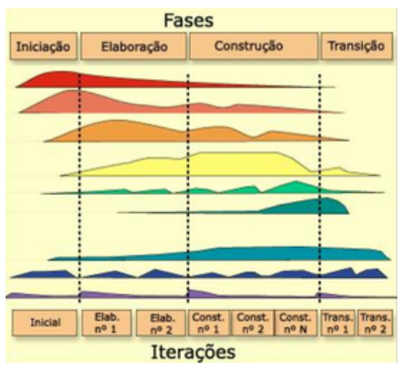


incremental



prototipação

espiral



RUP

Para documentar um processo, (qualquer um dos modelos acima) precisamos de uma linguagem. A mais comum é a

UML

DESAFIO 04

Ler sobre UML, entender os documentos e onde são usados.

Criar um diagrama de sequência de uma função de um APP ou programa que gostar.

#CTG aula 2.3

manifesto

INDIVÍDUOS E INTERAÇÕES
mais que processos e ferramentas
SOFTWARE EM FUNCIONAMENTO
mais que documentação abrangente
COLABORAÇÃO COM O CLIENTE
mais que negociação de contratos
RESPONDER A MUDANÇAS
mais que seguir um plano

+12 AGÉIS princípios

OBJETIVOS

Entregar versões funcionais em prazos curtos

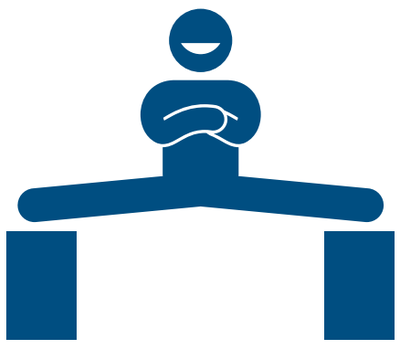
Estar preparado para requisitos que mudam com frequência

Negócios e desenvolvedores junto

Cliente é considerado parte da equipe

Prima troca de informações através de conversas diretas

Plano de projeto deve ser flexível



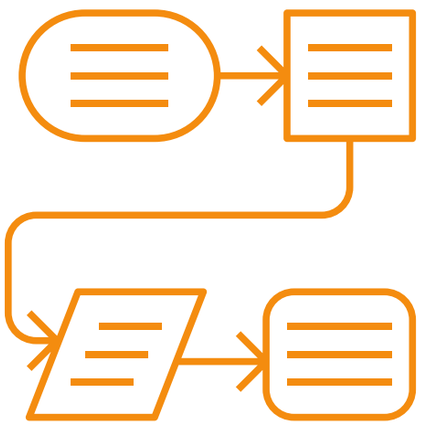
PROCESSO AGILIDADE

- O Processo deve ser capaz de administrar a **imprevisibilidade**;
- Processo facilmente **adaptável**;
- Adaptar **incrementalmente**;
- Equipe precisa de **feedback** do cliente para adaptações incrementais;
- Um protótipo operacional ou parte de um sistema operacional entregues em **curtos períodos** de tempo auxiliam no feedback do cliente.

métodos agéis

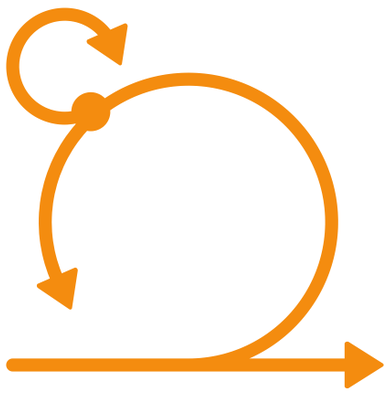
XP

Conjunto de práticas



SCRUM

Foco mais gerencial



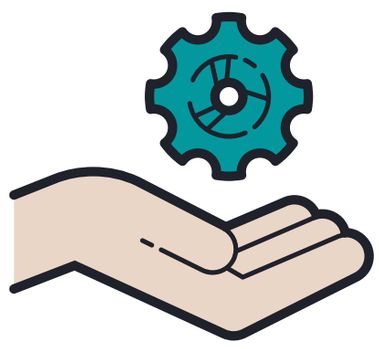
Muito provavelmente se usa uma mistura mistura dos dois

O que realmente importa com processos e métodos:
- resolve o problema?
- é adaptado a realidade onde está implantado?
-como é a comunicação?
- estamos realmente entregando o que o cliente quer?

#CTG aula 2.4

REQUISITOS
de um sistema são as descrições:

do quê o sistema deve fazer



dos serviços que ele oferece

das restrições do seu funcionamento

Podemos descrever esses requisitos de diversas formas:
- um texto explicando o que o usuário quer;
- usando templates dos modelos tradicionais de software como RUP ou outros;
-histórias de usuário (user stories); etc.

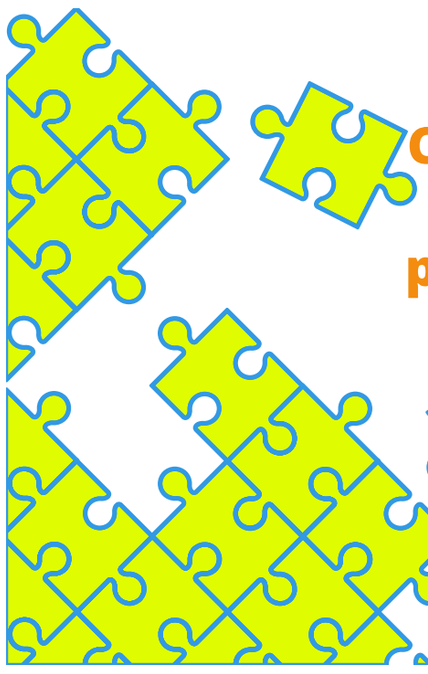
E é aí que o BDD entra como uma ferramenta que vai agregar valor em como colhemos e descrevemos esses requisitos

USER STORY

descrição simples concisa e clara de uma função que traz valor para usuários reais

=

descrição dos REQUISITOS



Cada estória adiciona um pouco de valor; normalmente precisamos de várias delas para conseguir implementar uma função completa

Minimal Marketable Feature
menor porção que podemos construir de forma que isso chegue ao mercado e possa ser usado por nossos usuários

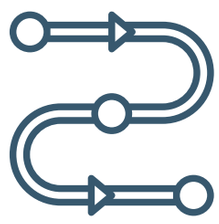
PARA QUEBRAR STORIES USA-SE O

Template da User Story com :

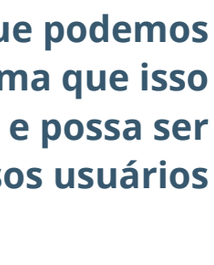
COMO
(papel do usuário) [As a..]
EU QUERO
(atividade) [I want]
PARA QUE
(valor de negócio) [So that]

possui 3 elementos

PROCESSO



LINGUAGEM



ARTEFATO

Behaviour Driven Development

BDD

ajuda a evitar Mini Waterfalls no processo de desenvolvimento

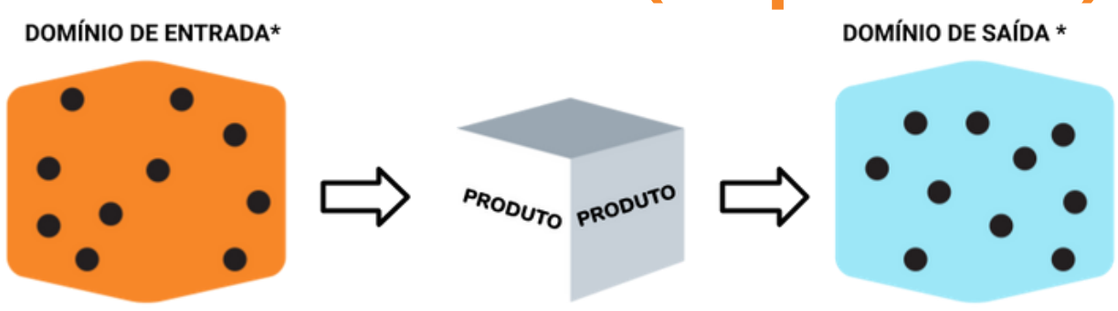
CENÁRIO 1
DADO (pré condições)
QUANDO (ação)
ENTÃO (resultado)

CENÁRIO N
DADO (pré condições)
QUANDO (ação)
ENTÃO (resultado)

DESAFIO 05
Escolher um aplicativo ou sistema que goste e descrever 3 user stories com o formato do BDD

#CTG aula 3.1

Teste exaustivo (impossível)



*pode ser infinito (∞)

GESTÃO DE DEFEITOS

Ao encontrar erros enquanto testamos precisamos REGISTRAR de alguma forma:

- **Registro de incidente ou de defeito**
- **Preferencialmente, em um sistema de rastreamento de defeitos (bugtracker)ex: Jira, GitLab, Bugzilla, Mantis, Trello, etc.**

O sucesso do teste está no projeto dos CASOS DE TESTE

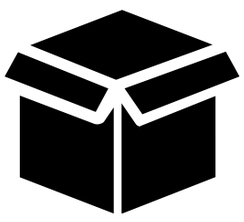
Um caso de teste é um DOCUMENTO que DETALHA OS PASSOS de um teste.
Possui no mínimo três partes:
ENTRADAS
SAÍDAS
ORDEM DE EXECUÇÃO

COMO ESCOLHER A TÉCNICA?

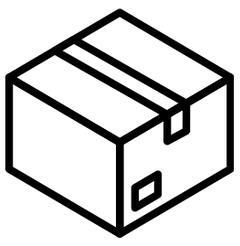
Cada técnica nos diz de onde obteremos as informações para criar os requisitos e casos de teste
EX: especificação de requisitos - TÉCNICA CAIXA PRETA
código fonte - TÉCNICA CAIXA BRANCA
histórico de defeitos - BASEADO EM DEFEITOS

Técnicas de Teste (Tipos de Teste)

Caixa-preta (ou Funcional)

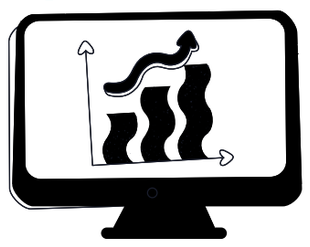


Caixa-branca (ou Estrutural)

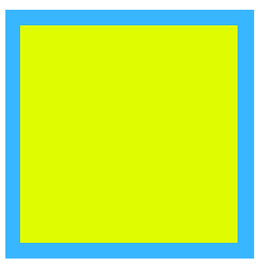


Baseado em defeito

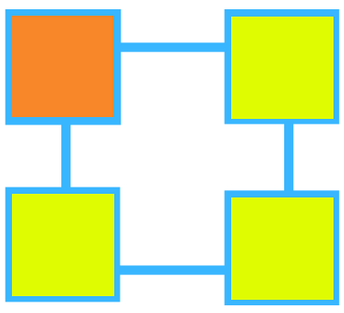
Baseado em experiência



Exploratório



Teste de Unidade



Teste de Integração

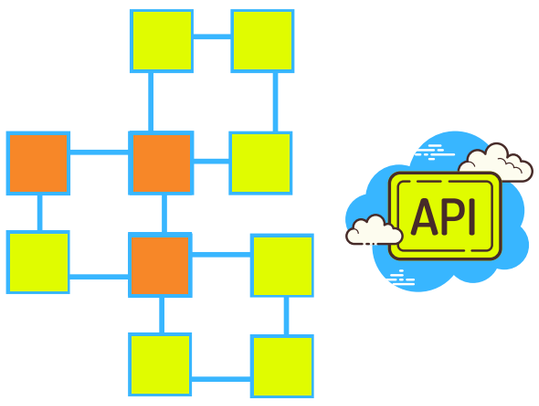
Testes Funcionais

X

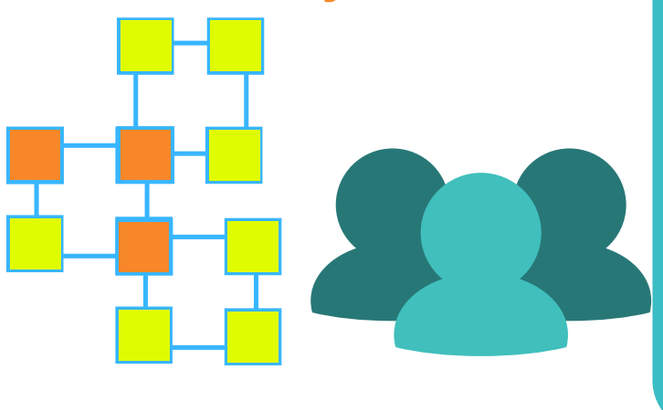
Testes Não Funcionais

Fases de Teste (níveis de Teste)

Teste de Sistema



Teste de Aceitação



DESAFIO 06

Escolher um sistema em conjunto no telegram e fazer casos de teste usando as técnicas de teste:

1. **Análise do Valor Limite**
2. **Classes de Equivalência**
3. **Tabela de Decisão**
   a. **OBS: escrever os testes no formado de BDD**

#CTG aula 3.2

Por quê planejar os testes?

Estimar esforço
Organizar trabalho
Definir métricas
Melhor qualidade

Forma de priorizar o que deve ser testado (processo)
X
Forma de documentar esse plano (documento)

Responder as perguntas:

- **O que (escopo/cobertura): front, back, mobile ...**
- **Como: técnicas, níveis e tipos ...**
- **Por quem: dev, tester ...**
- **Quando: cronograma, fases, níveis ...**
- **Resultados: reports, dados ...**
- **Automatizado x Manual**

COMO PLANEJAR OS TESTES ?

- **Risco!**
- **O que é mais importante?**
- **Quais testes preciso fazer para cobrir o mais importante?**
- **Quais os tipos de teste preciso fazer?**
- **Funcionais x Não Funcionais**

QUAIS FERRAMENTAS USAR

wiki, jira, plugins, testlink, mantis...

MindMaps

Bom e velho .txt
O que funcionar melhor na empresa e pra você!

ISO 29119-1
(substituiu a ISO 829)

#CTG aula 3.3

PIRÂMIDE DE TESTE

QUANTO MAIS ALTO MAIS:

- CARO
- LENTO
- FRÁGIL
- DEPENDENTE

Google sugere:

- 10% são E2E
- 20% dos testes são de integração
- 70% dos testes são de unidade

EQUILÍBRIO

O SEGREDO

EXPECTATIVA X REALIDADE

Framework é "um jeito" de se fazer algo usando algumas ferramentas;
Devemos selecionar programas para nos ajudar a realizar os testes;
As regras sobre como escreveremos os testes dependem dessas ferramentas

FERRAMENTA: determina o formato do teste

Frameworks de Teste Automatizado

é o conjunto dos programas necessários para realizaros testes juntamente com as regras de como essestestes serão escritos

Para uma framework de teste, precisamos de:

forma de executar os testes

jeito de verificar os resultados

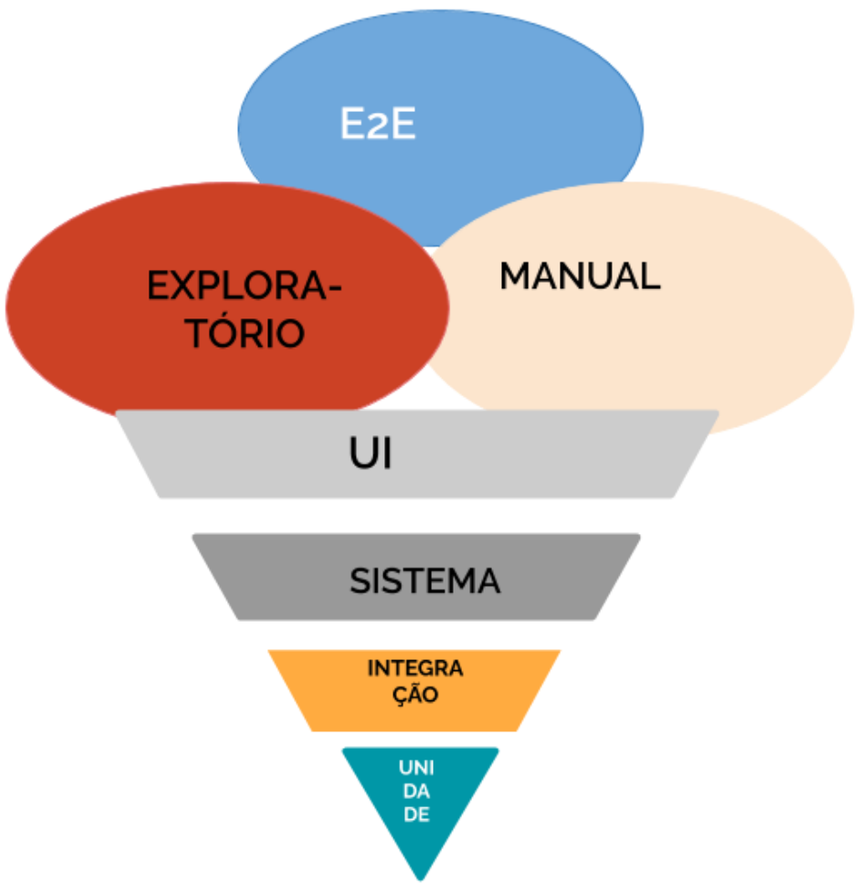
ferramentas adicionais

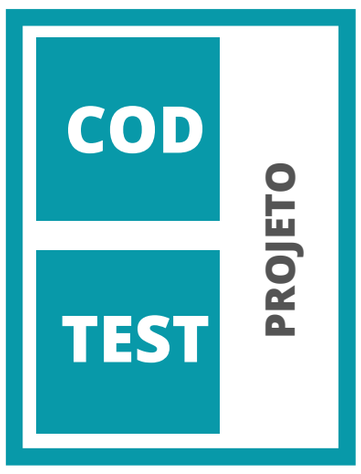
linguagem de programação

EX: JAVA e KOTLIN

Forma de executar os testes: **Junit/Cucumber**
Jeito de verificar os resultados: **Junit/Hamcrest**
Linguagem de programação: **Java/Kotlin**
Ferramentas adicionais: **restAssured cucumber**

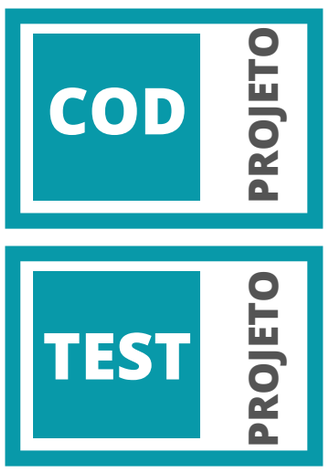
Ice Cream Cone: anti padrão





Organização dos testes

juntos OU separados com o projeto do código de produção



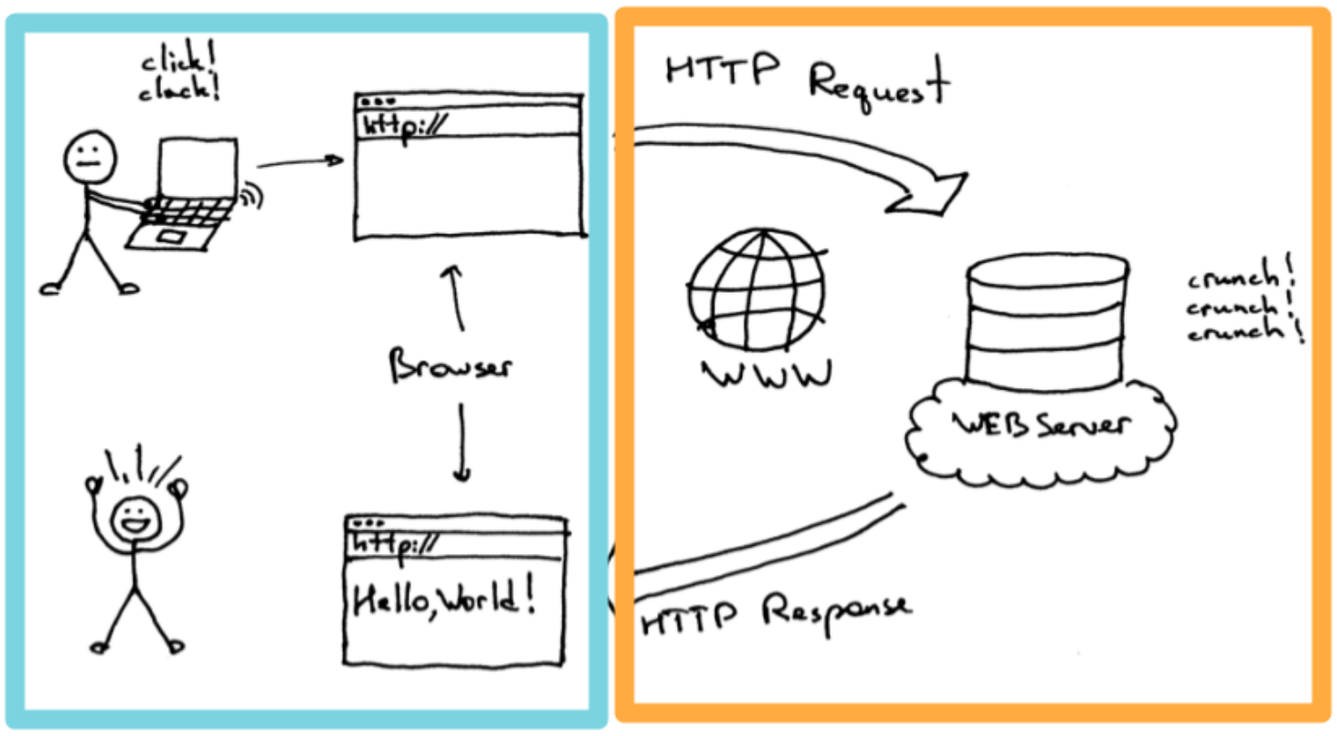
DESAFIO 07

Definir uma framework pra sua realidade

#CTG aula 3.4

SISTEMA WEB

digitamos um endereço (URL) no navegador, esse pedido viaja pela internet até o servidor em que o site está armazenado (hospedado). O servidor processa o pedido e envia os arquivos para o navegador, que também os processa e os exibe pra gente;



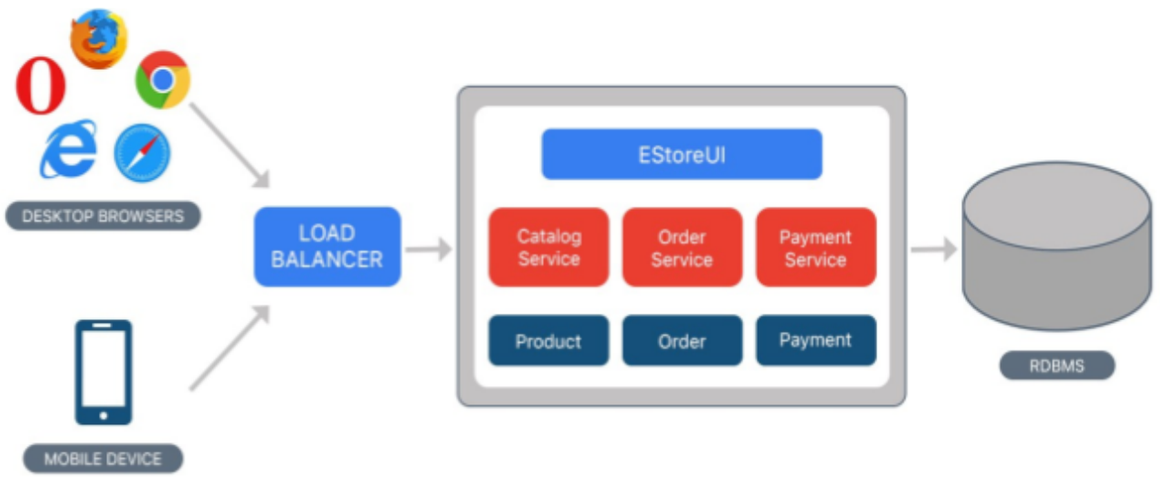
A arquitetura de um sistema WEB é a definição dos seu componentes e das interações entre eles;

- **aplicações, middleware, banco de dados, etc**

Ou seja, como eles funcionam juntos para realizar as ações esperadas.

FRONTEND **BACKEND**

Arquitetura Monolítica



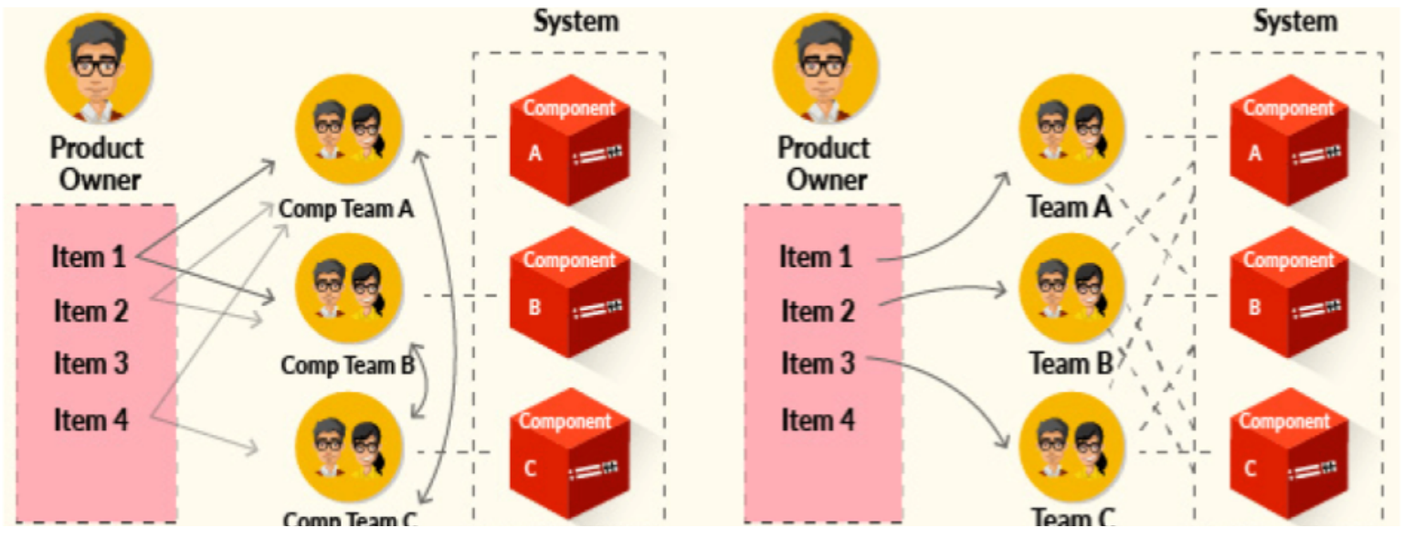
Arquitetura Micro Serviços

A arquitetura influencia em como vamos fazer os testes principalmente PERFORMANCE|SEGURANÇA |ESTABILIDADE

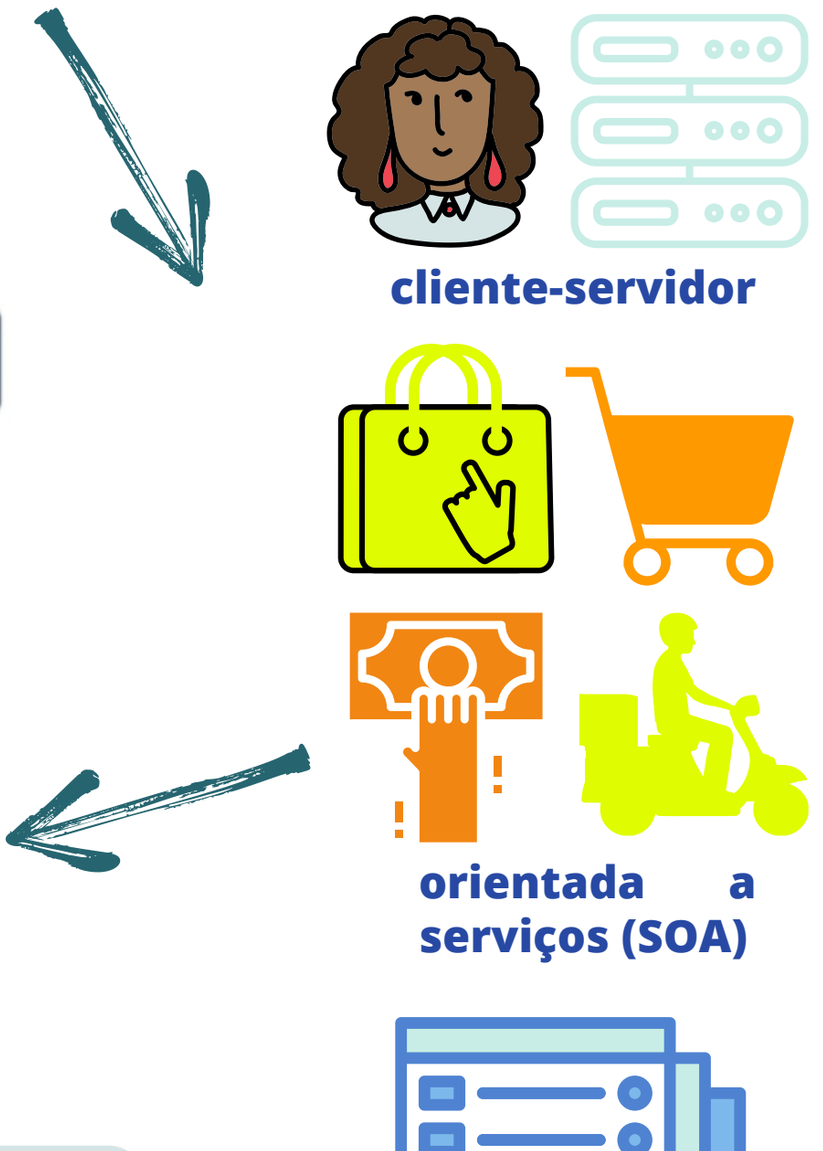
Any organization that designs a system (broadly) will produce a design whose structure is a copy of the organization's communication structure.

Lei de Conway

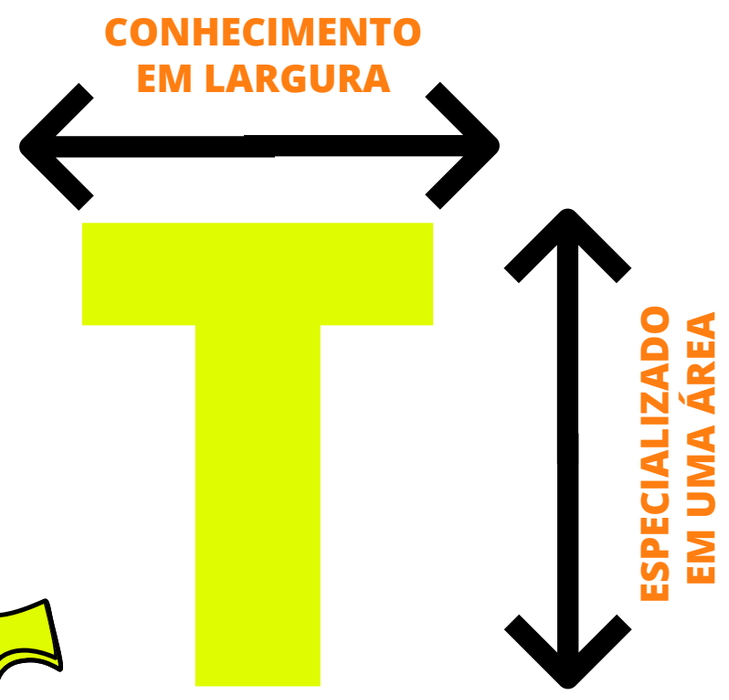
Component Teams | Feature Teams



TIPOS DE ARQUITETURA



Profissionais T-Shaped



DESAFIO 08

Descrever arquitetura de um sistema escolhido

#CTG aula 3.5

sistemas de controle de versão

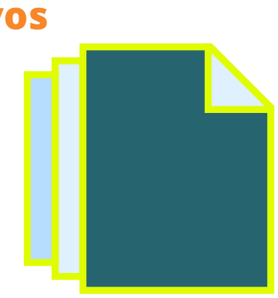
É um software/servidor “central” que guarda os arquivos

Ajuda a gerenciar mudanças no código fonte através do tempo;

Gerenciam a junção de partes do código trabalhadas por diferentes pessoas

Possibilita o trabalho em paralelo

Protege o código de erros humanos e catástrofes

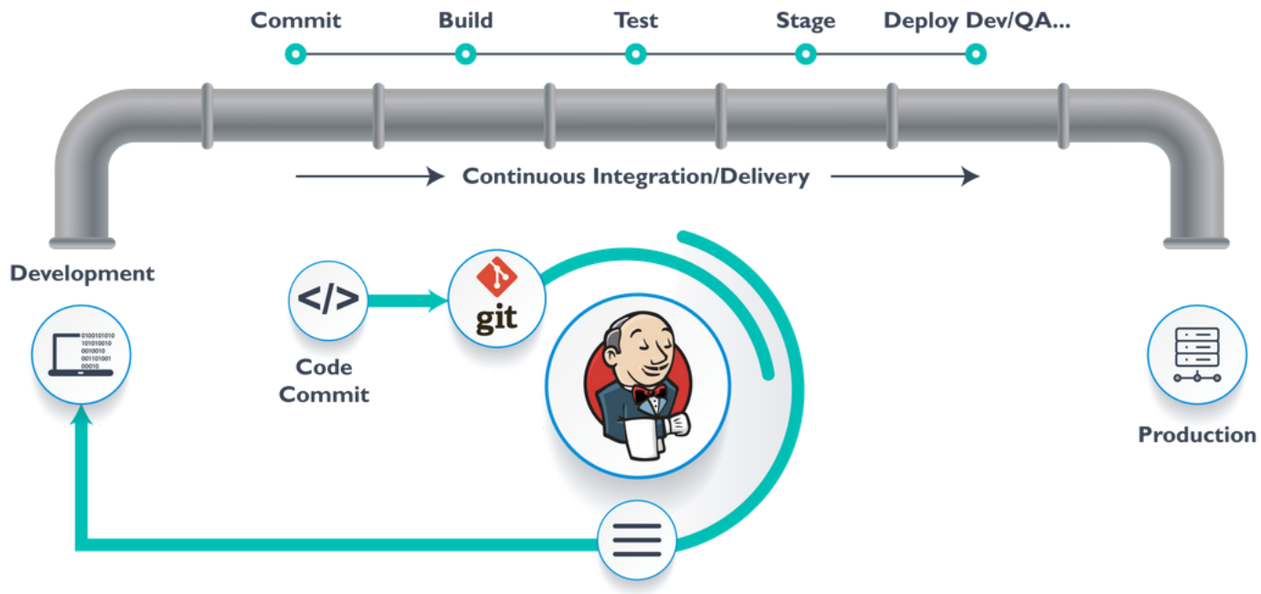


Ajuda a ver as diferenças entre versões, e quem fez o quê; adição, remoção e alteração

Nos ajudam na automação da **PIPELINE**

Pipeline ou linha de produção, são etapas que o software passa desde a máquina do desenvolvedor até chegar na produção.

- Integração contínua (CI)
- Entrega contínua (CD)



Branches

Forma de trabalho em que cada pessoa pode trabalhar paralelamente, e depois integrar de volta na versão principal.

Quando o desenvolvedor termina de desenvolver algo na sua branch, ele abre um pull request;
Que pode ser revisada (code review) por outros membros da equipe e só então integrada a branch de desenvolvimento principal

git

É um SCV open source, ativamente mantido

Implementações

Github
Gitlab
Bitbucket
SourceForge

Comandos Essenciais

Clone - clonar arquivos do remoto
Add - adicionar ao commit
Commit - salvar as mudanças localmente
Status - ver os arquivos que mudamos localmente
Push - enviar da nossa máquina pro servidor
Pull - baixar a versão mais atual do servidor pra nossa máquina
Fetch - pegar a lista mais atual de branches do servidor
Checkout - mudar para a branch específica ou criar uma nova

Conflict

Quando partes iguais do código possuem modificações concorrentes;

Para cada modificação conflitante precisamos resolver manualmente

Uma boa IDE facilita nossa vida pois nos mostra onde está o conflito e nos ajuda a resolver ele;

Escolhemos qual parte queremos que fique e qual sairá

Após a resolução do conflito precisamos fazer um novo commit e depois tentar dar o push de novo;

O git nos ajuda com mensagens de erro e sugestivas

Por fim, após resolução de conflito, o push é sucesso..

Checkout

retornar arquivos específicos para a versão igual a do servidor,
criar uma nova branch local;
alternar entre branches

END

DESAFIO 09

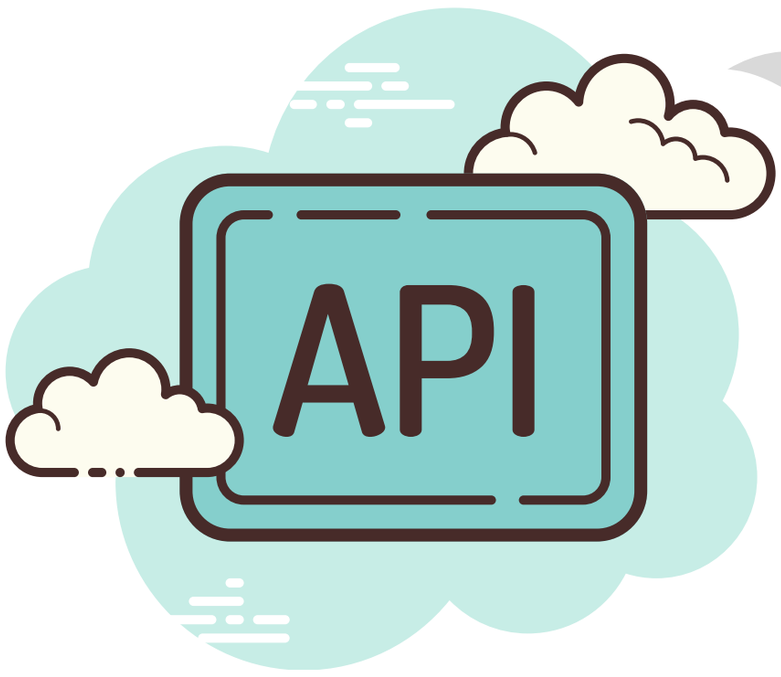
Criar sua conta no github, clonar o repositório, criar uma branch, incluir sua citação, criar o pull request para ela.

https://github.com/vinnypessoni

#CTG aula 4.1

Testes de API

Application **P**rogramming **I**nterface
Interface de Programação de Aplicação



“é um conjunto de rotinas, protocolos e ferramentas para construir aplicações de software”

o papel das APIs é possibilitar o transporte de dados entre aplicações, bancos de dados e dispositivos

Web Services são softwares que permitem comunicação por meio da internet;
As APIs deles permitem isso

Como o uso de APIs WEB se popularizou muito, foi preciso criar padrões de comunicação;
Padronização facilita com que aplicativos feitos em diferentes linguagens e rodando em diferentes ambientes possam se comunicar;
Assim surgiu o SOAP e REST

Simple Object Access Protocol
Representational State Transfe

SOAP e REST

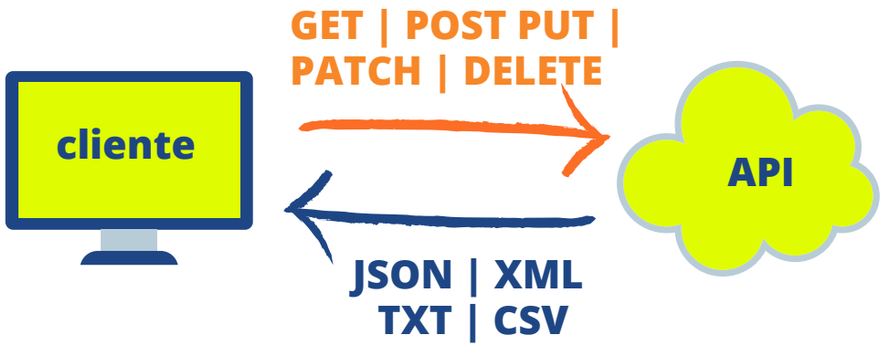
é um protocolo de comunicação entre aplicações
Independente de linguagem, plataforma e transporte (http, FTP ...)
Padronizado
Sempre XML para codificar as mensagens (enviar e receber) Lida com erro de forma integrada
Envia um envelope, mais complexo e precisa de mais dados
Todos os métodos via POST

é um estilo arquitetural
Tenta solucionar alguns problemas do SOAP e prover uma forma mais facil de acessar web services
Pode usar XML, JSON, csv, txt, etc para codificar as mensagens
Mensagens mais simples, menos dados
Usa get, post, put, delete do HTTP

O princípio da Interface Uniforme é o coração das APIS REST e possui 4 frentes:
- Identificação do recurso nas requisições;
- Manipulação do recurso por meio de representações;
- Mensagens auto descritivas;
- Hypermedia como o meio de definir o estado da aplicação.

REQUISIÇÕES HTTP

O HTTP é um protocolo para comunicação da internet

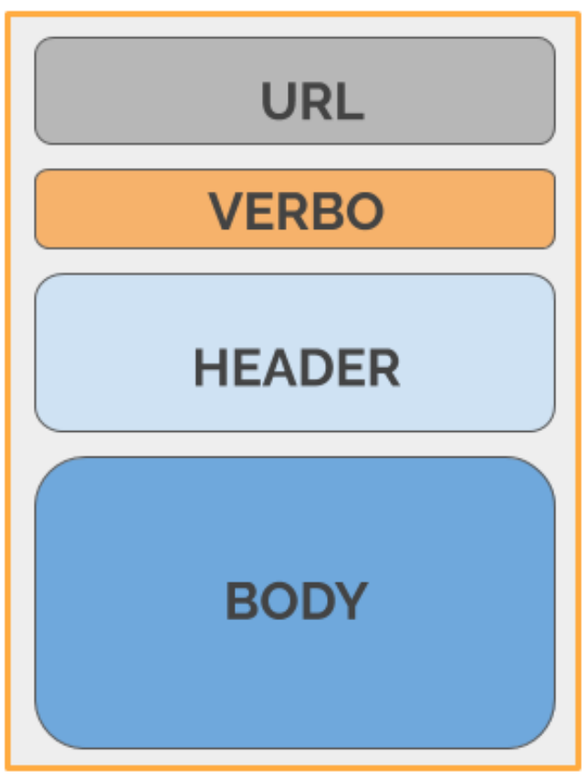


Um caminho para o recurso (endpoint) http://exemplo.com

Um verbo HTTP, que define qual ação realizar : GET | POST | PUT PATCH | DELETE

Um cabeçalho (header), que permite ao cliente passar informações da requisição Basic HTTP, OAuth, noneHTTP Headers

Um corpo (body) opcional da mensagem com dados;normalmente em Json ou XML



O que tem na requisição HTTP

Para usar as APIS precisamos fazer requisições para elas;
E para testá-las também!

FERRAMENTAS = ajudam a testar APIs

POSTMAN

APACHE JMeter™

RSPEC

REST-assured

A AULA 4.2 FOI UMA AULA COMPLETAMENTE PRÁTICA, PORTANTO NÃO HÁ ESQUEMA VISUAL

#CTG aula 4.3

O HTTP é um protocolo para comunicação da internet

Códigos HTTP

cada RESPOSTA de uma requisição HTTP tem

CÓDIGO

MENSAGEM

2xx sucesso
4xx erro do cliente
5xx erro do servidor

2xx
200 OK
201 Created
202 Accepted
204 No Content

4xx
400 Bad Request
401 Unauthorized
403 Forbidden
404 Not Found
405 Method Not Allowed

5xx
500 Internal Server Error
502 Bad Gateway
503 Service Unavailable

O que testar nas APIs ?

Testes Funcionais

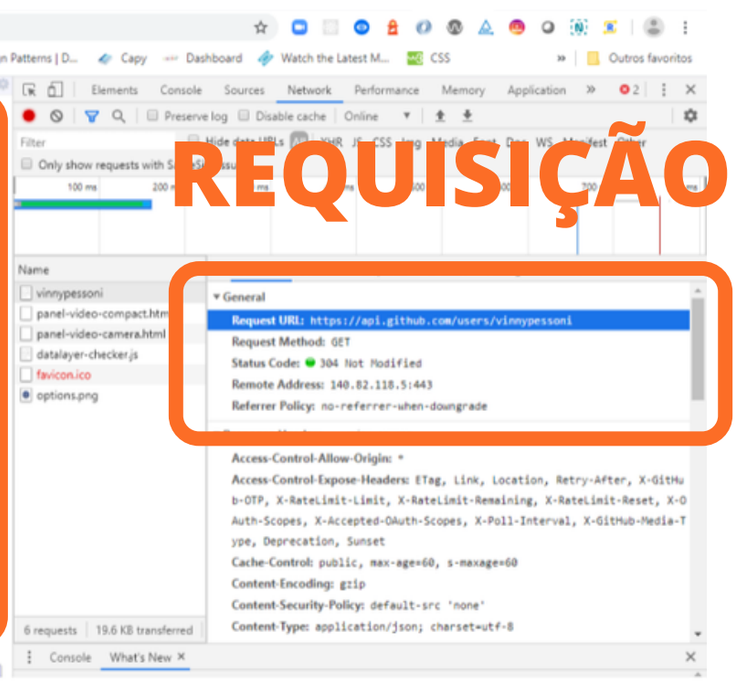
Testes Não Funcionais

Testes de API com POSTMAN

Usamos o Postman para:
- Realizar requisições para APIs
- Fazer testes automatizados

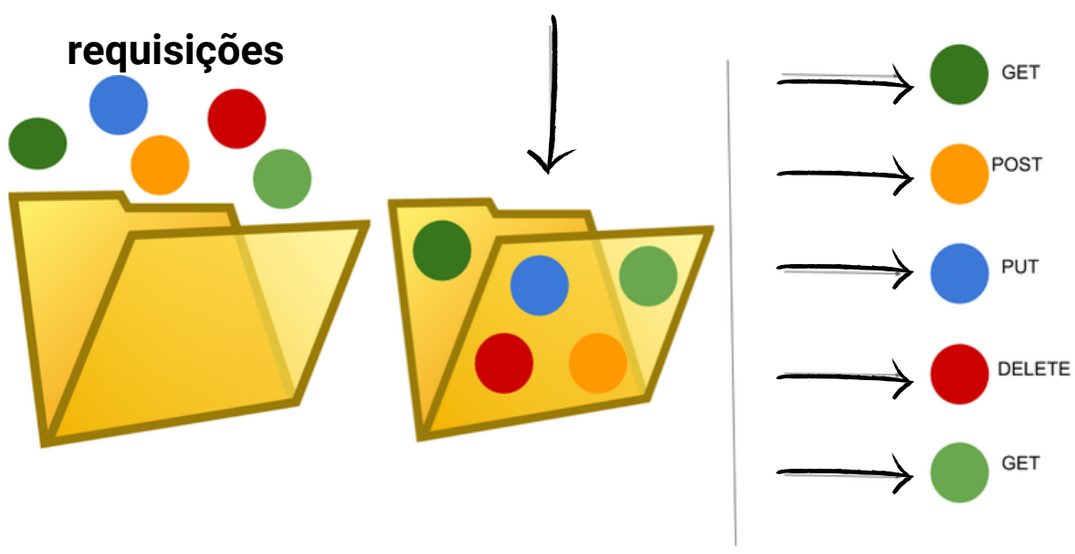
RESPOSTA

REQUISIÇÃO

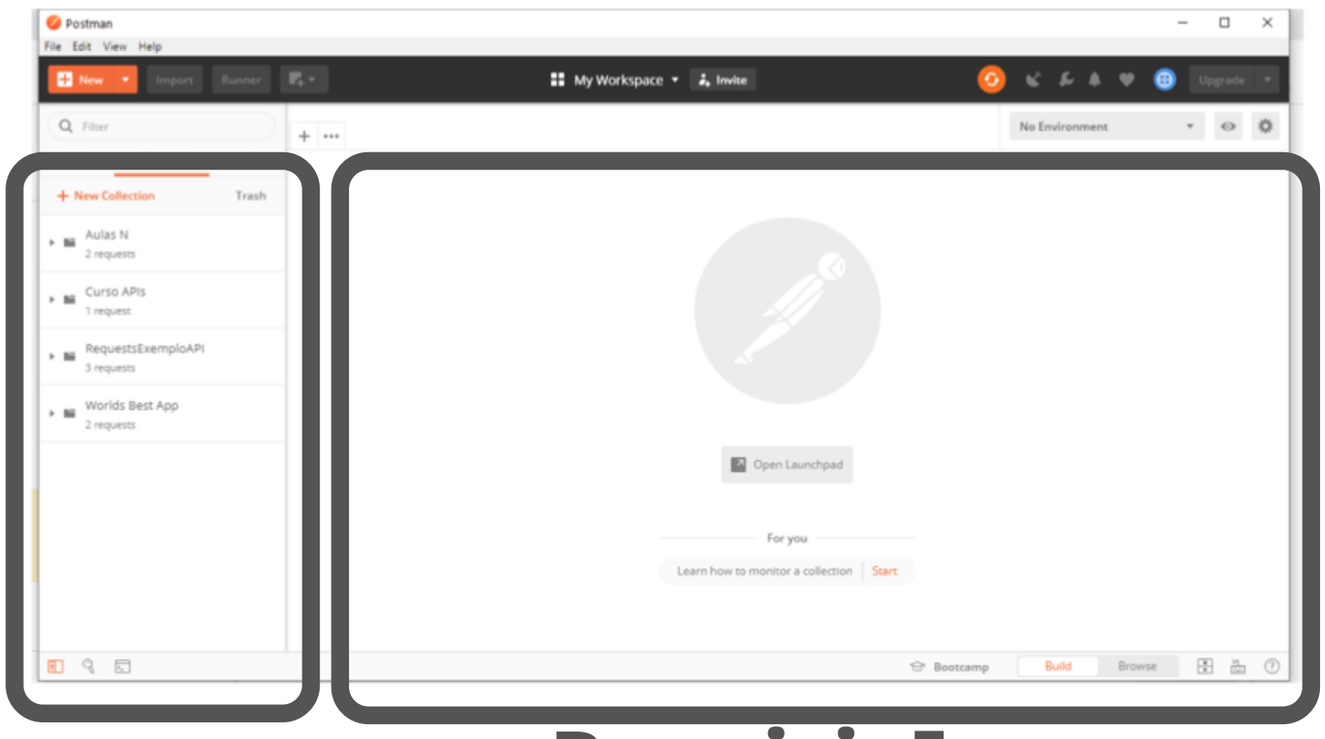


coleções

Cada coleção agrupa as requisições (como se fossem pastas)



Visão geral da ferramenta

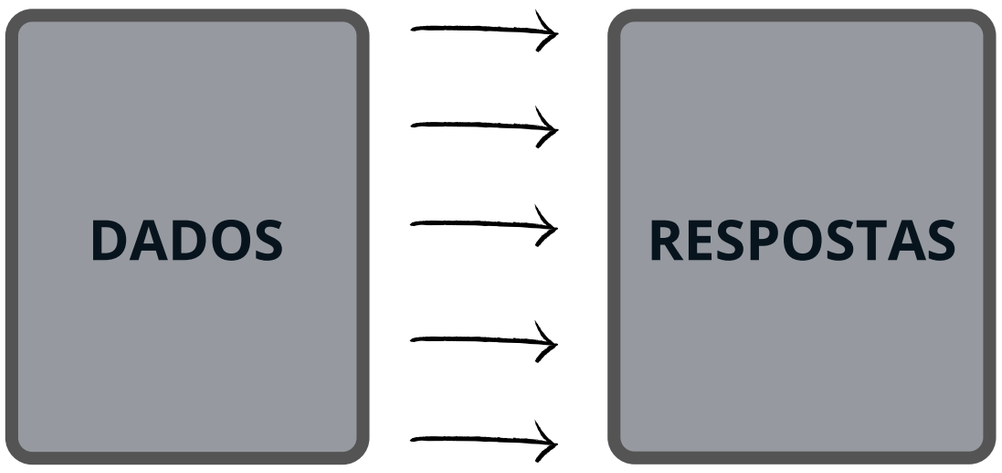


coleções

Requisições

requisições

Cada requisição tem os dados da requisição e a tela de resposta



DESAFIO 10

Escolher uma API pública e exercitar os métodos HTTP (GET, POST, PUT e DELETE)

#CTG aula 4.4

PARA UMA

framework de teste

PRECISAMOS DE :

forma de executar os testes

jeito de verificar os resultados

linguagem de programação

ferramentas adicionais

podem ser **juntos** OU **separados** com o projeto do código de produção

ESSAS FERRAMENTAS NOS AJUDAM A:
- simplificar requisições
- tirar screenshots
- produzir relatórios
- verificar resultados

Montando a sua do Zero

Forma de executar os testes:
Junit

Jeito de verificar os resultados:
Junit/Hamcrest

Linguagem de programação:
Java

Ferramentas adicionais:

Simplificar requisições:
restAssured REST-assured

1. Abrir o intellij
   criar novo projeto > Gradle > java -> Finalizar
2. Abrir o arquivo build.gradle e colocar as dependências que precisamos nele
   Junit5, RestAssured e outras que escolher
3. Criar uma classe java para conter os testes
   src -> test -> Java
4. Importar as bibliotecas do rest assured
5. Criar um método para testar um endpoint

TESTES DE API

FUNCIONAIS
- -cobertura dos endpoints
- -código da resposta da requisição
- -corpo da resposta da requisição
- -integração com outras apis (serviços/microserviços)

NÃO FUNCIONAIS
- -performance (6 tipos)
- - segurança
- -escalabilidade

Dependências Usadas

```
testImplementation 'io.rest-assured:rest-assured:4.3.0'
testImplementation 'org.junit.jupiter:junit-jupiter:5.6.0'

test {
  useJUnitPlatform()
}
```

Senior Dicas
- Teste manualmente primeiro para entender o fluxo, as requisições e respostas
- Use breakpoints enquanto desenvolve
- Leia a documentação da biblioteca que está usando
- Veja exemplos de códigos! da documentação oficial e também de blogs, sites, etc.
  - Se o gradlew ficar dando uns erros esquisitos, force a versão dele a ser a mais atualgradle -> wrapper -> gradle-wrapper.properties]
  - https\://services.gradle.org/distributions/gradle-6.4-bin.zip

Anotações Junit 5

@test
- todos os nossos testes precisam ter essa anotação pro Junit saber que ele é um teste

@DisplayName
- melhores nomes pros testes

Hooks
- @BeforeEach: executar ação antes de cada teste
- @AfterEach: executar ação depois de cada teste
- @BeforeAll: executar ação antes de todos os testes
- @AfterAll: executar ação depois de todos teste

DESAFIO 11

Automatizar testes Put e Delete da nossa API com restAssured

#CTG aula 4.5

executa os testes de uma forma determinística mas em **uma ordem não previsível**

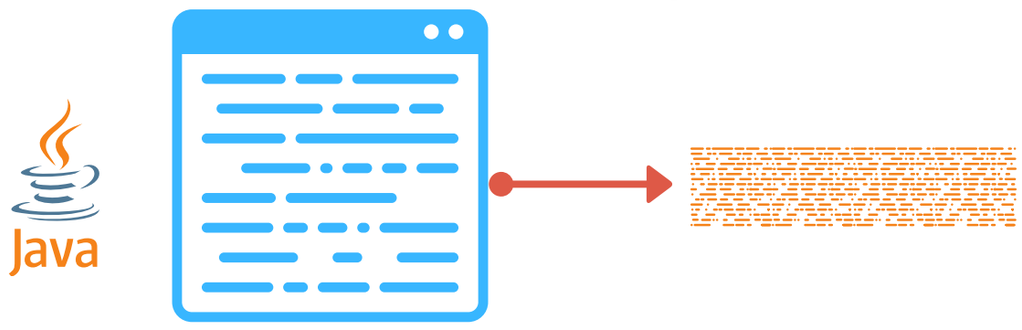
podemos “forçar” uma ordem específica de execução mas isso **não é uma boa prática**

Cada teste deve ser auto contido ou seja **Testes atômicos**

ter tudo o que é necessário para que ele seja executado

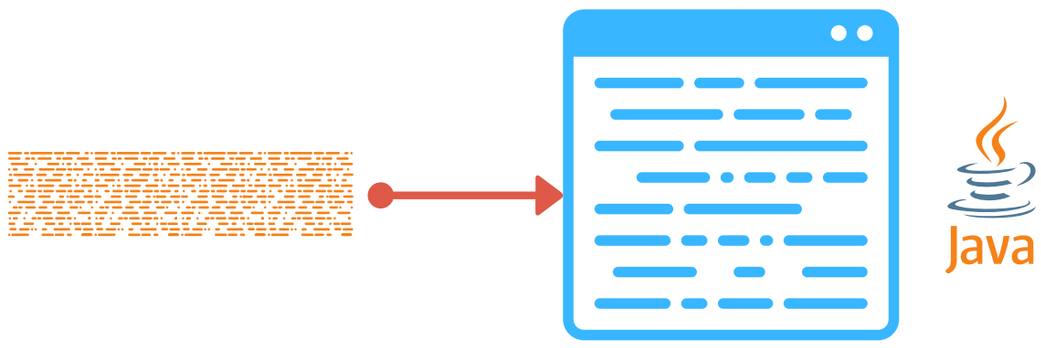
MÍNIMO DE DEPENDÊNCIAS POSSÍVEIS

serialização



transformar seu objeto em algum outro formato para transporte ou armazenamento

deserialização



pegar seu objeto de volta a partir de um formato para transporte ou armazenamento.

No caso das apis e serviços web, é transformar de/para Json, Xml, etc

SOLID

- **S**ingle responsibility principle
- **O**pen closed principle
- **L**iskov substitution principle
- **I**nterface segregation principle
- **D**ependency inversion principle

Clean Code

- Código legível
- Evitar repetições
- Nada de números mágicos no meio do código (por em constantes)
- Métodos com uma única função
- Código auto explicativo

Temos 3 branches nos exemplos de testeshttps://github.com/vinnypessoni/teste-api-clientes-restassured-java-gradle-junit5

DESAFIO 12

Refatorar o seu código atomicamente

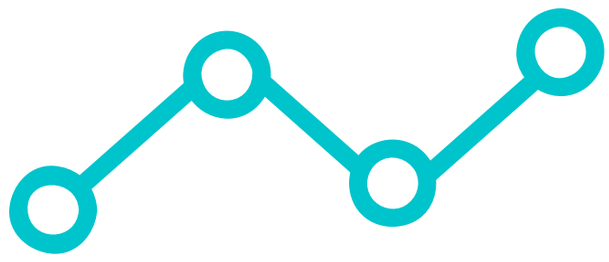
Master: nível júnior, para quem está aprendendo

Codigo-refatorado-mid-range: nível esperado de um **pleno**

Codigo-refatorado-senior: nível esperado de um **senior**

#CTG aula 5.1

Testes partindo da interface são os das camadas mais altas da pirâmide de teste. Então devem ser MINORIA!

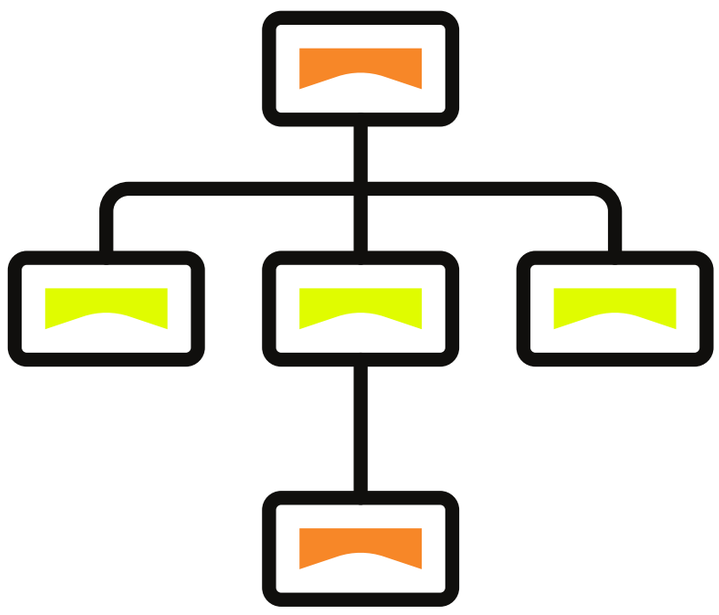


Tipos de Teste que podemos fazer a partir da interface

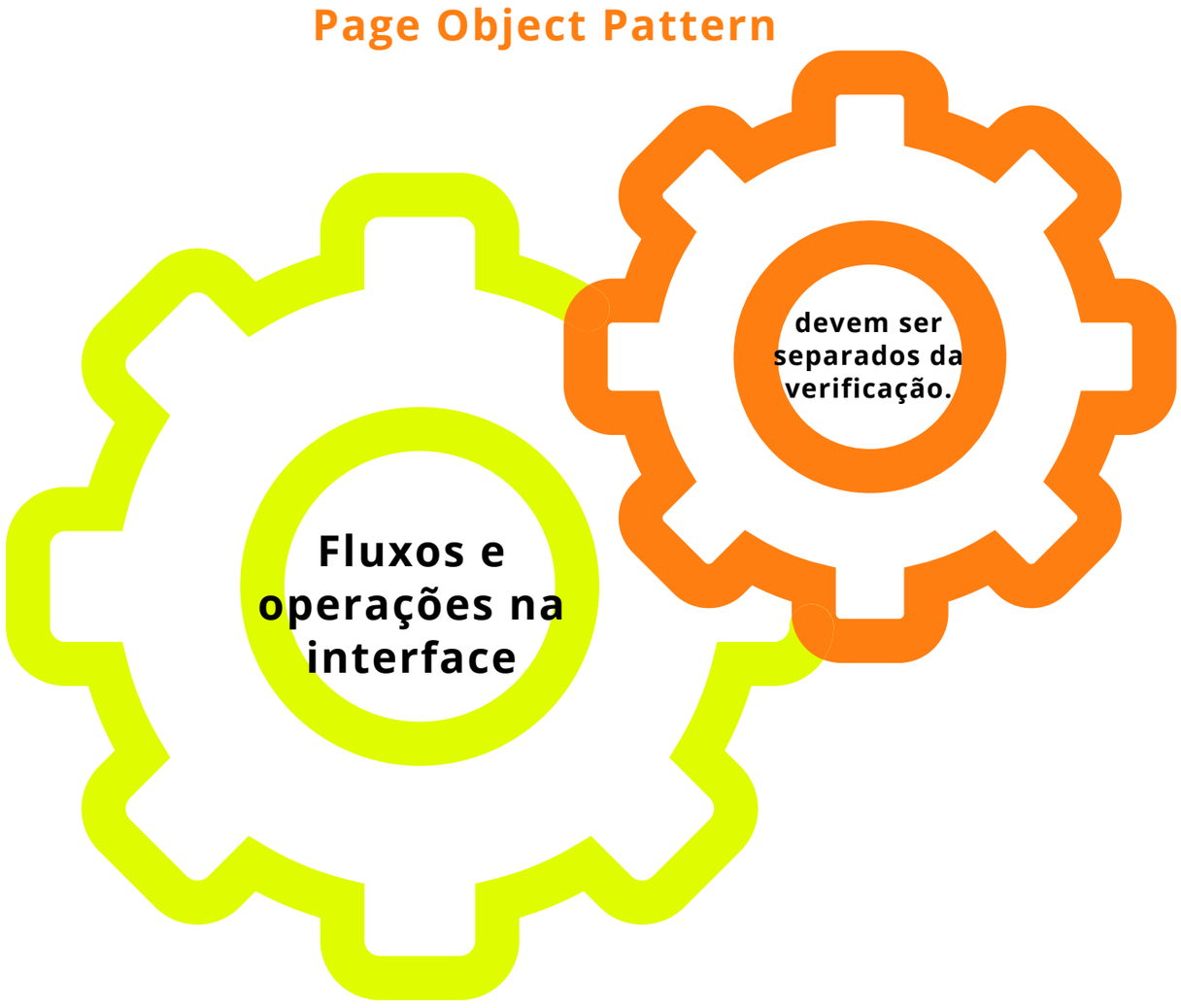
- Fluxos do produto e jornadas de usuário (e2e)
- Se um componente renderizou corretamente e está no lugar correto (SnapShot Testing), muito comum em react apps
- Responsividade (funciona bem em diferentes telas e dispositivos?)
- Usabilidade (quantidades de clicks, cores, contrastes, fontes, etc.)
- Acessibilidade
- Performance (carga, desempenho, ...)

DOM: Document Object Model

É uma árvore de objetos que representa a página para que possamos manipular
Quando uma página é carregada, o navegador cria a DOM da página
Disponível para html, xml, javascript



Page Object Pattern



Nossa Framework de Testes

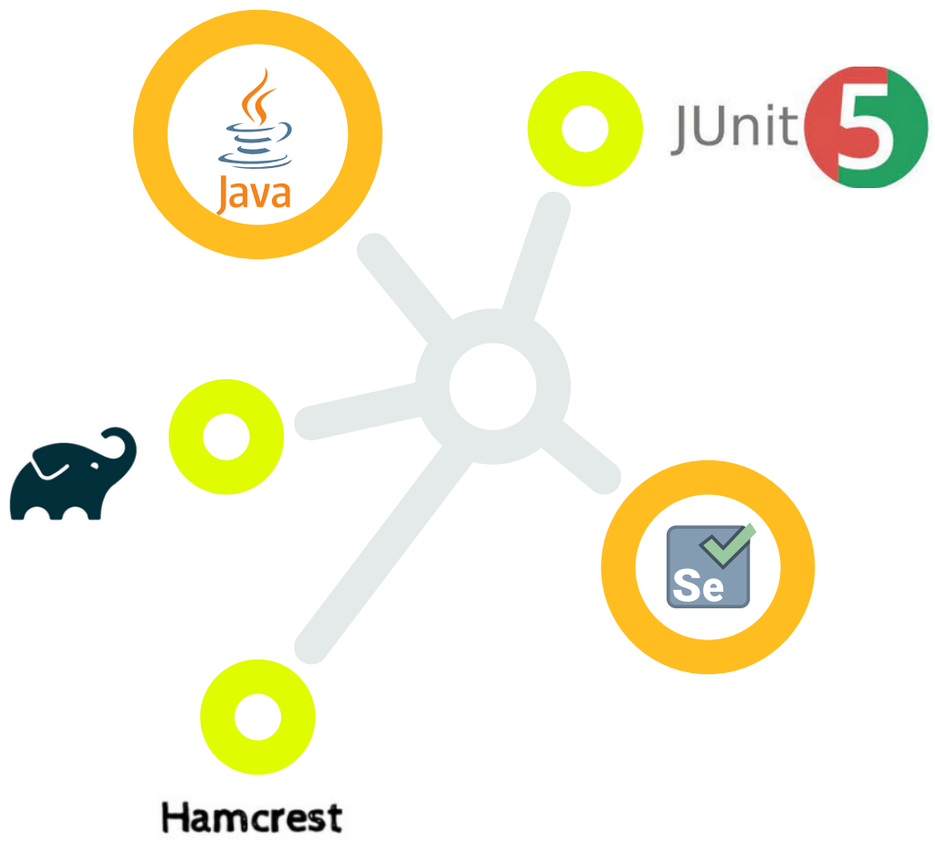
Forma de executar os testes:
Junit 5

Jeito de verificar os resultados:
Junit 5/Hamcrest

Linguagem de programação:
Java

Manipulação interface:
selenium web driver

Ferramentas adicionais:
Gradle: gerenciar dependências e gerar relatórios.



DESAFIO 13

Escolher um site público e automatizar 2 fluxos com page objects

#CTG aula 5.2

Nossa Framework de Testes

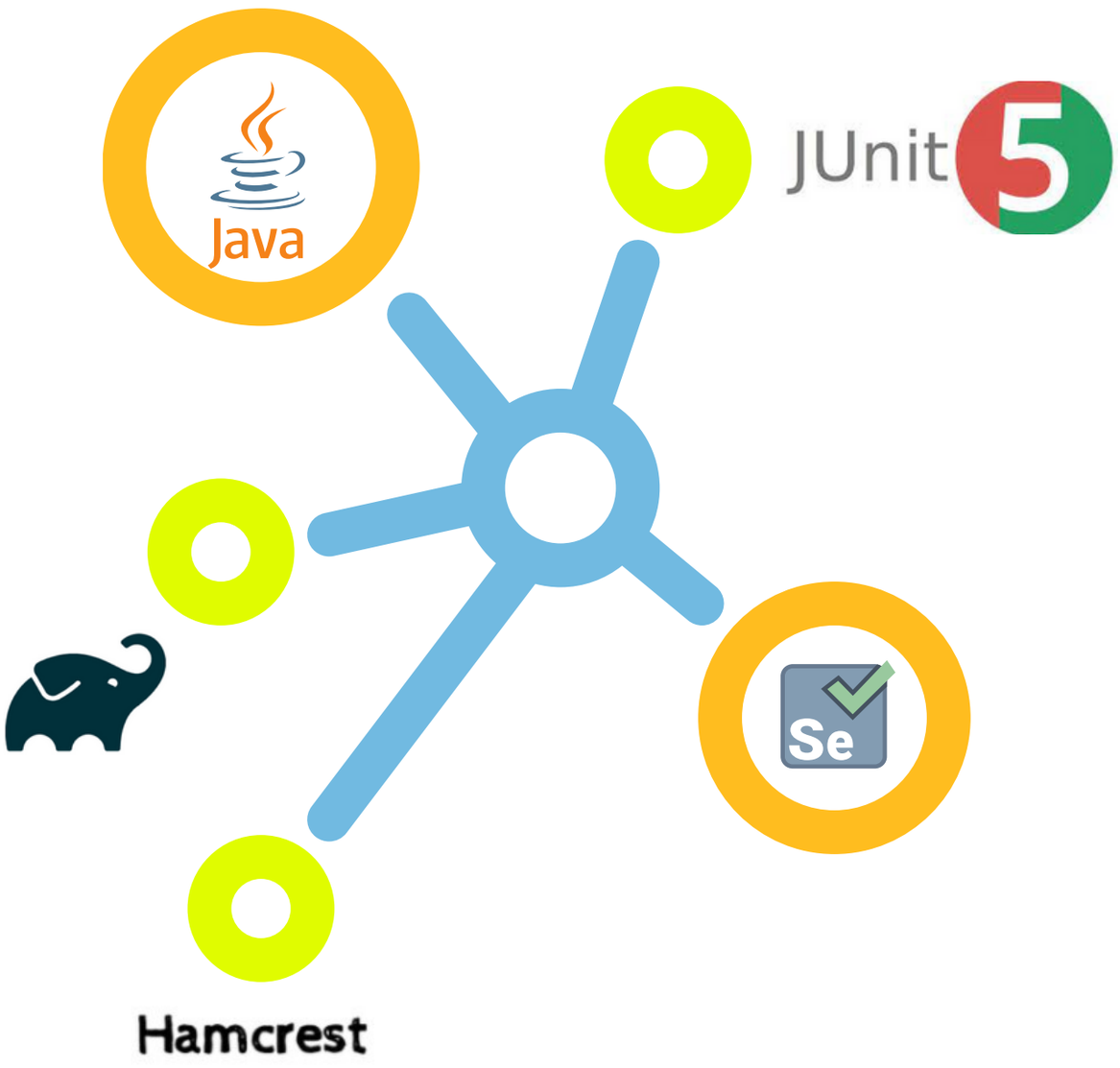
Forma de executar os testes:
Junit 5

Jeito de verificar os resultados:
Junit 5/Hamcrest

Linguagem de programação:
Java

Manipulação interface:
selenium web driver

Ferramentas adicionais:
Gradle: gerenciar dependências e gerar relatórios.



Precisamos localizar os elementos para manipular com selenium e fazer nossos testes

O selenium possui 8 localizadores de elementos

CSS ID: find_element_by_id
CSS class name: find_element_by_class_name
name attribute: find_element_by_name
DOM structure or xpath: find_element_by_xpath
link text: find_element_by_link_text
partial link text: find_element_by_partial_link_text
HTML tag name: find_element_by_tag_name

Ou podemos encontrar vários elementos usar .find_elements()

Ordem de preferência para localizadores

1. ID
2. CSS
3. Título
4. XPATH (evite ao máximo)

#CTG aula 6.1

CURRÍCULO

2 PÁGINAS IDEAL
(MÁXIMO 3)

ORDEM DE INFORMAÇÕES REVERSA CRONOLÓGICA
(do mais recente pro mais antigo)

DETALHAR EXPERIÊNCIAS

- qual era o problema?
- o que fez?
- quais ferramentas usou?
- qual o resultado gerou pra empresa?

ATS ROBOTS
Faça um currículo para passar pelos robôs e NÃO SER DESCARTADO inicialmente.

PALAVRAS CHAVES
use as palavras que coincidam com as vagas de trabalho que tem interesse

FORMATOS E DESIGN SIMPLES
- Docx ou PDF
-nada de coisas muito rebuscadas, gráficos, etc

TÍTULOS COINCIDENTES
Use termos, nomes de vagas similares quando se referir a experiências passadas

CERTIFICAÇÕES
cite as abreviações certas de certificados: CTFL, MSc, BACh, PMP

+ LinkedIN

RECOMENDAÇÕES
Peça recomendações dos seus colegas de trabalho. Mas antes disso, dê suas recomendações à eles também como gentileza

DESTACAR SUAS HABILIDADES
Colocar habilidades mais pedidas das vagas (que coincidem com as suas) e pedir pros colegas de trabalho votar nelas

GitHub

Um portifólio de projetos de programação, onde se pode hospedar e compartilhar códigos. Com ele pode-se gerenciar projetos, atualizar e acompanhar atualizações dos colaboradores

CRIE O SEU
Siga o passo a passo

https://www.aboutmonica.com/blog/how-to-create-a-github-profile-readme

inspire

Templates legais do readme pessoal

https://github.com/cyrisxd
https://github.com/anmol098/anmol098
https://github.com/martonlederer/martonlederer
https://github.com/Thaiane/Thaiane
https://github.com/monkindey/monkindey
https://github.com/anuraghazra/anuraghazra
https://github.com/anuraghazra/github-readme-stats

DESAFIO 14

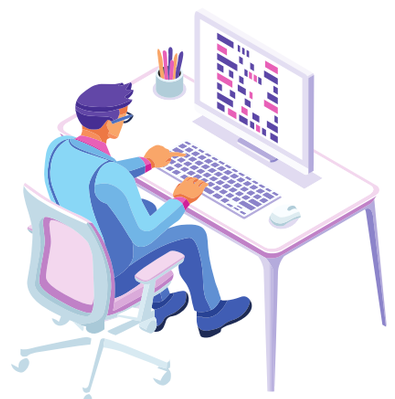
Criar/Atualizar Linkedin e Readme gitHub

#CTG aula 6.2

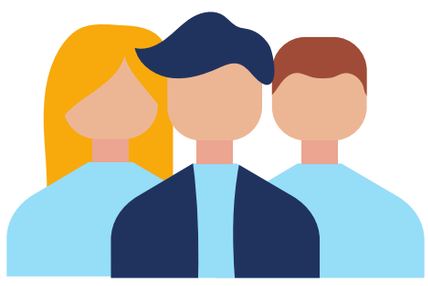
FASES ENTREVISTA DE EMPREGO EM TI

1. Recrutador

2. Entrevista Preliminar



3. Desafio Técnico



4. Entrevista com Alto Escalão

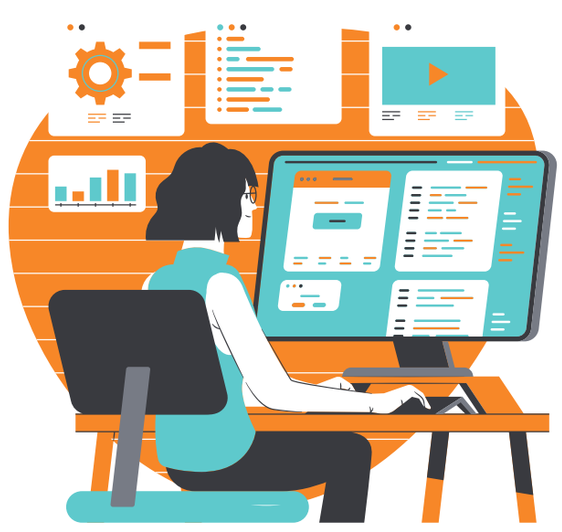
5. Negociação Salarial

6. Oferta Formal

COMO ME PREPARAR PARA ENTREVISTAS?

1. Conhecer a empresa
O que ela faz?
qual o negócio?
quero trabalhar com isso?

2. Olhar a vaga, entender o que é pedido, o que você tem e o que precisa desenvolver



3. Treinar partes técnicas e postar elas no github (quando permitido)

DICAS GERAIS

O QUE ESTOU VESTINDO IMPORTA?
NÃO TENHO TODOS OS REQUISITOS DA VAGA, DEVO ME APLICAR?
O QUÃO SINCERO DEVO SER?
MEU INGLÊS NÃO É PERFEITO, O QUE FAZER?
TREINAR RESPONDER PERGUNTAS PESSOAIS E DE EXPERIÊNCIA
EMPRESA PEQUENA VERSUS EMPRESA GRANDE
COMO SEREI CONTATADO E NÚMERO SKYPE
PONTUALIDADE
ENTREVISTAS DE VÍDEO OU ÁUDIO
VENCENDO O MEDO
EXISTE VIDA APÓS PANDEMIA?
ENERGIA!!!!!!!

SITUATION
TASK
ACTION
RESULT

DESAFIO 15

Aplicar pra 1 entrevista que te intimide

#CTG aula 6.3

CERTIFICAÇÕES

declaração formal de comprovação

credibilidade + autoridade

Prova > Certificado

QUAIS POSSO FAZER?

Teste (CTFL, TMMi ...)
Scrum master (SCM) PO
ITIL/COBIT
Desenvolvimento (Oracle)
DevOps (AWS...)

Brazilian Software Testing Qualifications Board

International Software Testing Qualifications Board

AS MAIS PEDIDAS

COMO ESTUDAR PARA ELAS ?

SYLLABUS | LIVROS

https://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html

+ MATERIAL GRATUITO NA INTERNET

#CTG aula 6.4

PROCESSO SELETIVO English Interviews

Recrutador

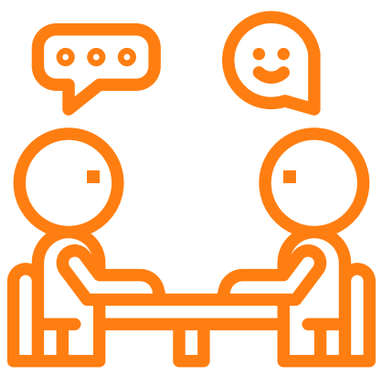
PASSO 01

Você manda seu currículo para uma vaga e o recrutador é quem entra em contato contigo antes de passar para a empresa.

Entrevista por Telefone

PASSO 02

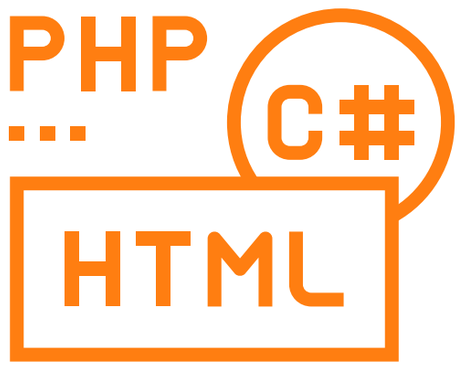
Sua entrevista preliminar com a primeira pessoa da empresa para te conhecer melhor e explicar a vaga



Desafio Técnico

PASSO 03

Seja para programador ou testador, nesse momento você vai demonstrar suas habilidades de forma prática. Membros da equipe que você vai trabalhar normalmente estão presentes nesses desafios.



Entrevista com Alto Escalão

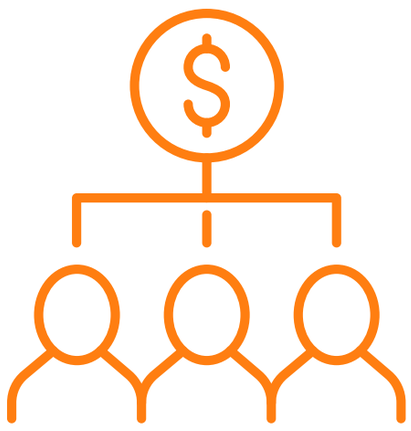
PASSO 04

Após passar pela fase técnica, nesse momento você entrevistado por diretores ou outros cargos acima da sua equipe

Negociar Salário

PASSO 05

Negociar o salário é a última etapa da seleção e após isso você receberá a oferta formal



GERAIS — exemplos

- What are your strengths?
- What are your weakness?
- Why should we hire you?
- Explain why you look for a new job.
- What can you offer us that someone else can not ?
- Explain your tools, knowlege and hard skills.
- Tell me about your qualifications.
- Discuss your professional experiences
- Explain your main duties and results

TIPS

Practice to answer interview questions with a friend/teacher. If you don´t have anyone, record yourself then watch later to see improvement opportunities

ESPECIFICAS DA ÁREA — exemplos

- Write a program to swap 2 numbers
- Explain and design components of the automation framework for a web application?
- Design an Automation framework for testing REST APIs?
- What defines and decides a bug's priority and severity?
- What is Equivalence Partitioning? Illustrate with an example.
- How to decide what is going to be automated and what has to be done manually?
- How is flow from requirements from business till it gets to production?
- How will You overcome the Challenges if the Proper Documentation for testing does not exist?