**Tema:** Criando repositório e explicando o roadmap

**Notas:**

GIT é um sistema de ***controle de versão***. Ele permite também o trabalho em equipe num mesmo projeto.

Exemplos: Github, Bitbucket, Gitlab…

Para acessar um repositório, você precisa de credenciais. Entre elas: SSH Key, HTTPs, entre outros.

O processo de ssh key é feito apenas uma vez por maquina.

git clone é apenas uma vez também.

Abri o git bash (terminal de comando)

comandos úteis:

ls - listar tudo o que tem dentro de uma pasta

cd - entrar dentro de uma pasta

Conceitos:

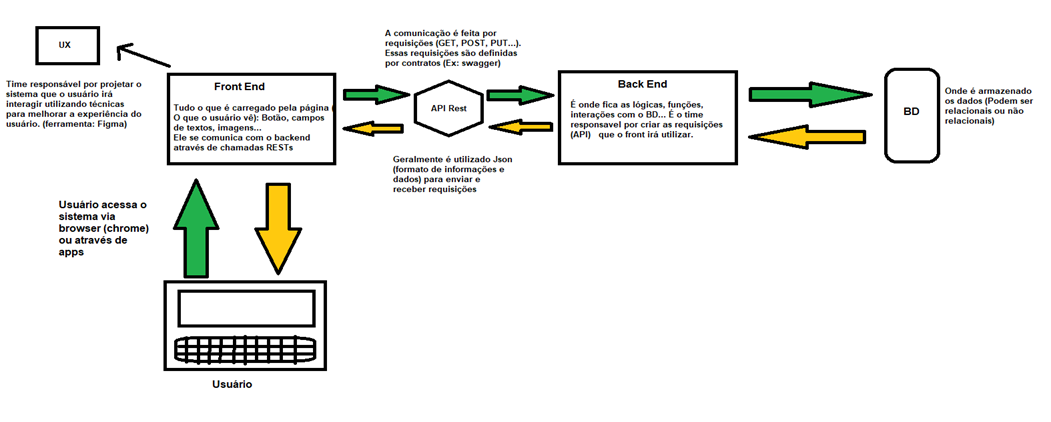
Branch representa uma parte ou todo de um projeto main ou master representa o sistema principal( pai) onde contem todos as funcionalidades

Sempre utilizar esses comandos:

1. git pull (traz tudo de novo para sua maquina)
2. git status - lista o que tem de novo, removido ou alterado
3. git add <file> - indica qual arquivo será enviado para o repositorio
4. git commit representa um conjunto de mudanças (git commit -m “mensagem do commit”)
5. git log (verifica o commit feito)
6. git push (envia para o repositorio suas mudanças)

:Q -> exit

***Tema: Arquitetura básica de software e conceitos QA 1***

******

***UX-*** Responsável por projetar a parte visual das aplicações considerando a experiencia do usuário. Ferramenta de prototipagem de sites, apps… Figma

**FrontEnd** - Responsável pela codificação das telas sugeridas pelo UX e conexão com o Backend.

Software: React e Angular

**BackEnd** - Responsavel pela condificação das logicas, funções, conexões com o banco, etc. Também desenvolve API (Application Programming Interface) para disponibilizacao dos servicos.

Linguagens: Java, Go-lang, Javascript, Node…

**Banco de dados** - Responsavel pelo armazenamento de dados

***Tema: Requisitos, Metodologias (sdlc) e Conceitos QA.***

***Requisitos:***

1- O que são:

1. Desejo/Pedido de um cliente
2. Algo obrigatorio a ser feito

2-Divisoes:

1. Requisito Funcional (RF) - **O QUE um sistema deve fazer** (login do facebook)
   1. Teste funcional - O que o sistema deve fazer (Ex: Teste de valor limite1- No brasil é obrigatorio votar de 18 a 70 anos. O nosso sistema deve validar essa idade.

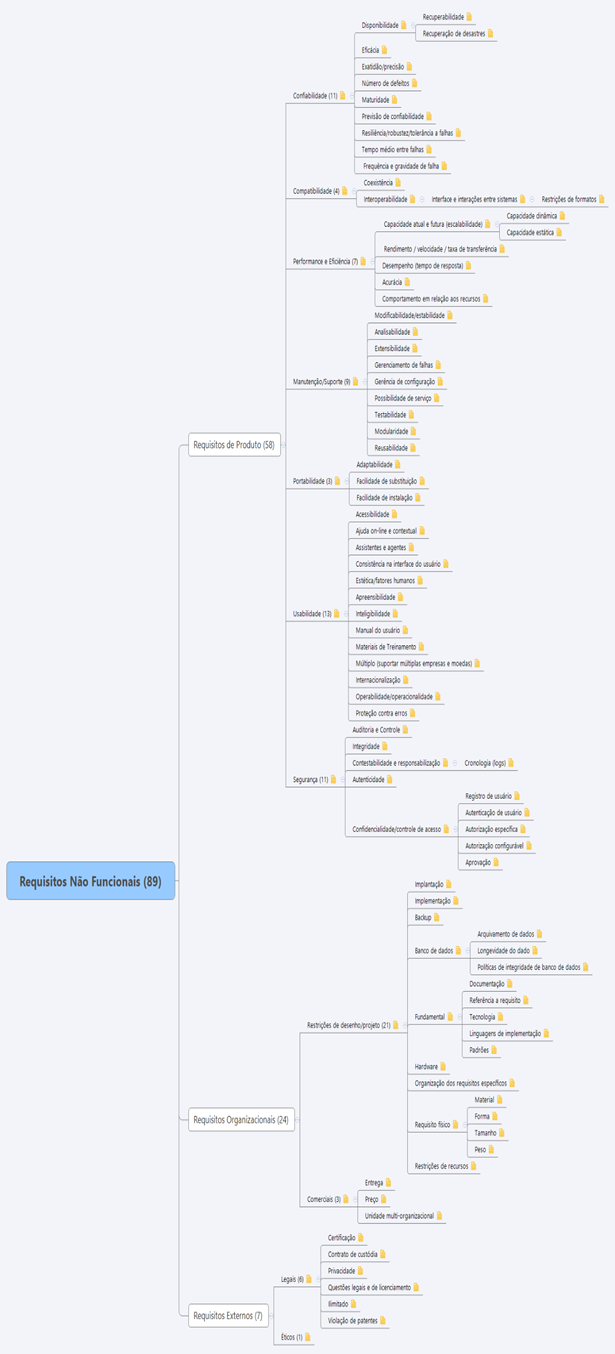
Idade: \_\_

\_\_\_\_\_\_\_\_\_1718\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7071\_\_\_\_\_\_\_\_\_\_\_

1. Requisito Não Funcional (RNF) - **COMO/QUÃO O sistema deve funcionar** (login do facebook deve durar 3s)
   1. Testes não funcionais- Como ou quão bom o sistema se comporta (https://retraining.inf.ufsc.br/taxonomiaRNF/printableRNF.html)
      1. Usabilidade
      2. Segurança
      3. Performance
      4. Compatibilidade
      5. Manutenção
      6. Responsividade
2. Requisito de Regra de Negocio (RNF) - Define um cenário de ocorrencia

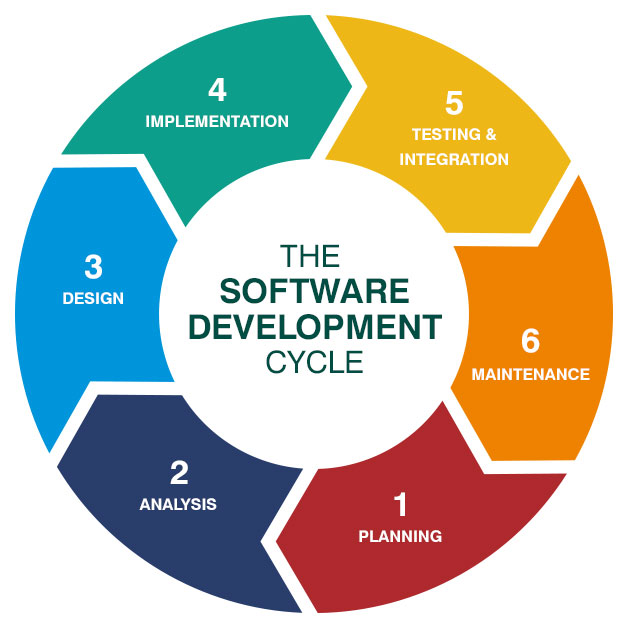
Um transponder tem que enviar uma taxa de 1.2GB em uma fibra 1mm e funcionamento de 24h por dia

O facebook deve efetuar o login por usuario e senha via browsers: Chrome e Mozilla 24h por dia



**SDLC (software development live-cycle)**

Processo constituindo de fases para o desenvolvimento de um software ou produto. Desde a concepção ate a manutenção.



1. Planning- É uma fase no qual se planeja o que sera entregue ou produzido. Já contém os requisitos.
   1. EX: Login do facebook
      1. RF- O login deve ser feito por usuario e senha
      2. RF- O sistema deve informar o usuário quando as credenciais estiverem erradas
2. Analysis- É uma fase no qual iremos analisar os requisitos e as demais documentações e ou contratos.Serve para identificar falhas ou falta de informações nos requisitos ou documentos. Também serve para definir as tecnologias que serao utilizadas. QA analisa por exemplo o que testar, quando, qual ferramenta irá usar…
   1. EX: Requisito 1: Não é claro qual tipo de login ele usará (pode gerar confusão na hora do desenvolvimento e testes)

3- Design - Como será feito, definindo prototipos. (figma). Também define o fluxo da aplicação. Como QA, também pode começar a planejar os cenarios de testes (sobre os requisitos, docs, prototipos).

4- Implemention - Codificação. O ideal é que todos os requisitos estavam prontos e não existe nenhuma duvida.

5- Testing & integração - Desenvolve (continua criando), executa e da manutenção nos casos de testes. Identificar os defeitos (bugs). Executar testes adicionais (performance, segurança)

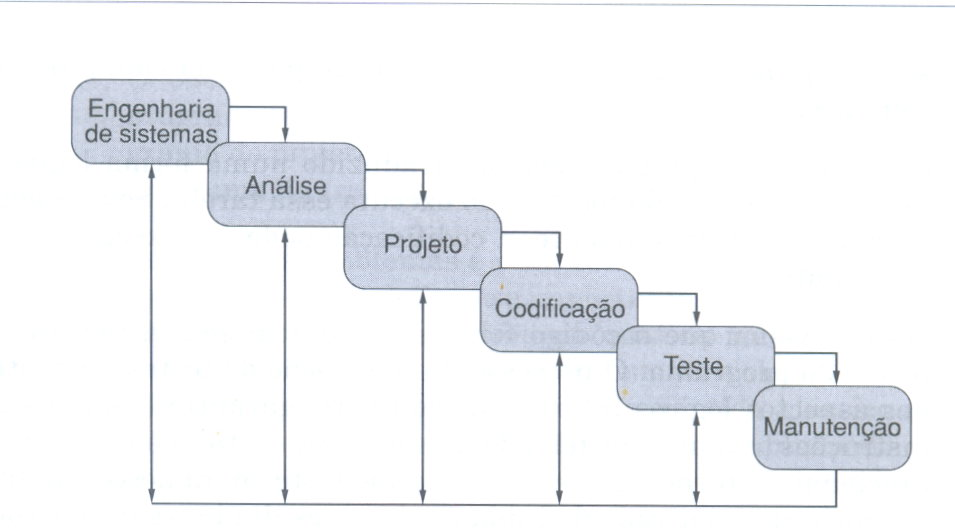
6- Delivery/release - Entrega do projeto com a maxima qualidade possivel e abrangendo todos os requisitos solicitados pelo cliente.

7- Maintenance - garantir o suporte do time apos a entrega e realizar possiveis manutenções.

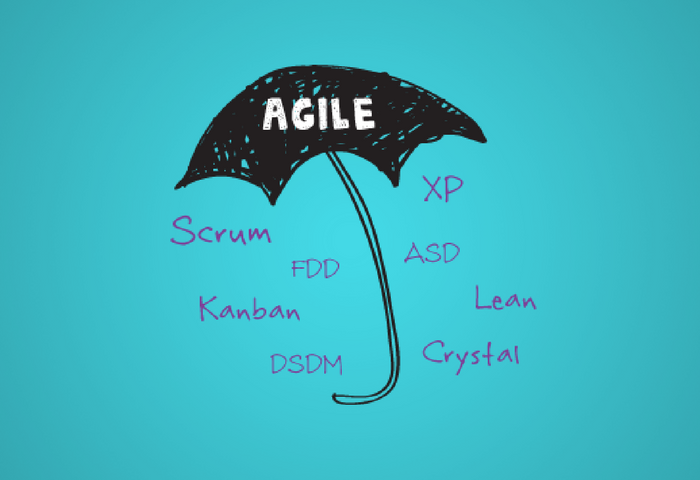
**Ciclo de desenvolvimento -> metodologias de desenvolvimento de software**

1 Metodologias tradicionais (old but gold)

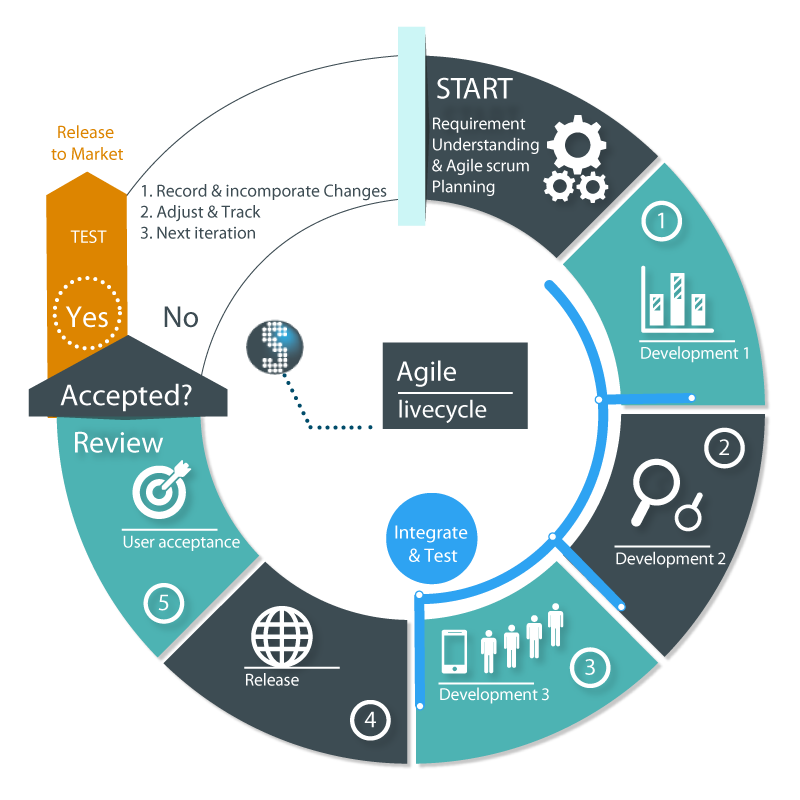
Consistia num prazo longo de desenvolvimento de projeto no qual o projeto era entregue tudo de uma vez. Cada fase do desenvolvimento so podia iniciar quando a outra acabava (waterfall, v model, caracol). Principal problema era a questão de qualidade e riscos de cronograma, custos.



Metodologia ageis (**incremental**)



Entregar o produto em pequenas partes. A cada entrega o produto ia aumentando e o cliente ja poderia ter acesso ao produto nas primeiras semanas de desenvolvimento. Iriamos garantir qualidade em cada entregavel e no produto final



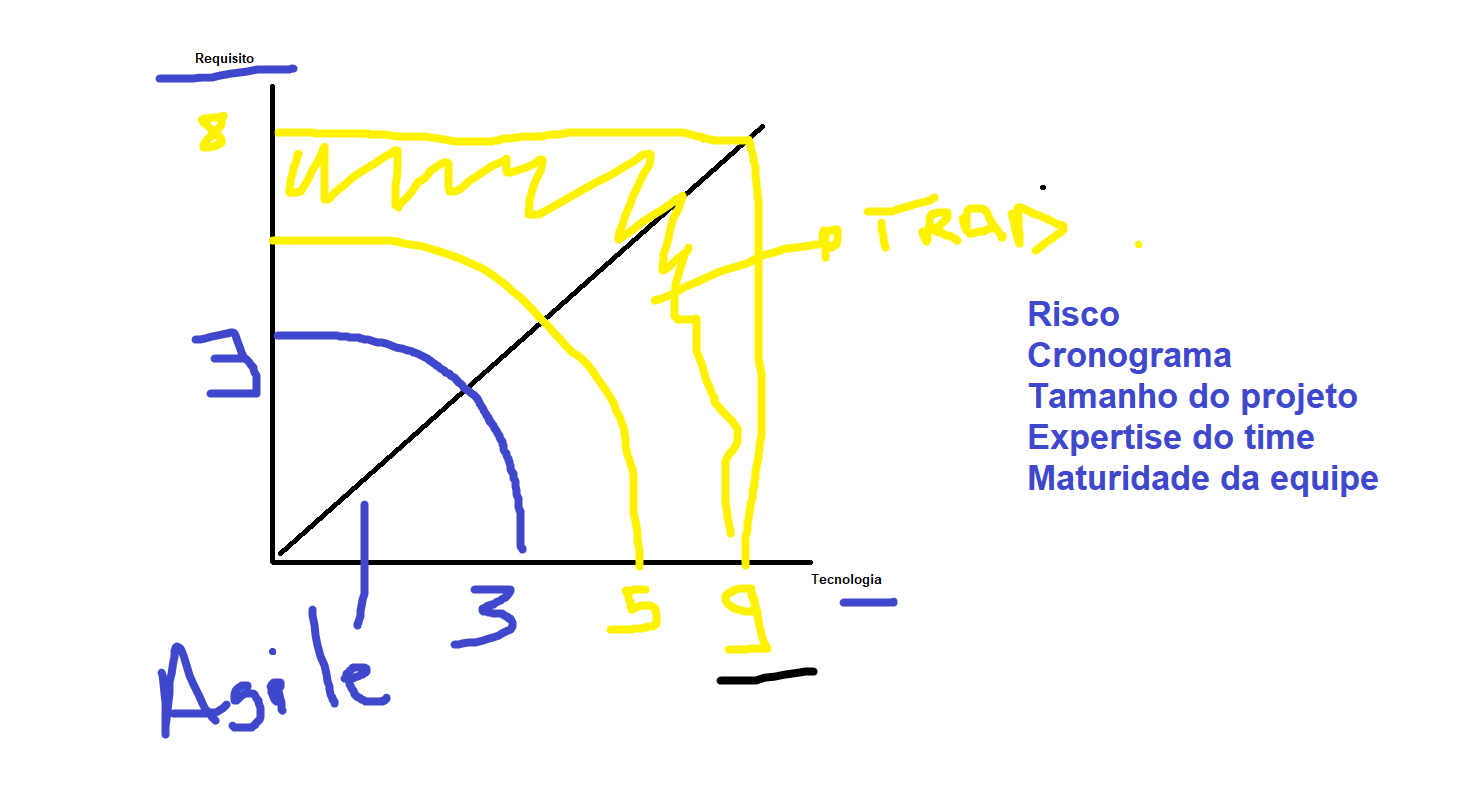
release - liberação

stakeholder - pessoas interessadas no projeto

\*\*\* O que é confiabilidade?

Garantir **Qualidade** ao longo do tempo. Garantir que todas as funcionalidades funcionem de acordo com o esperado (ex: O login nao pode parar de funcionar após desenvolver a tela de perfil)

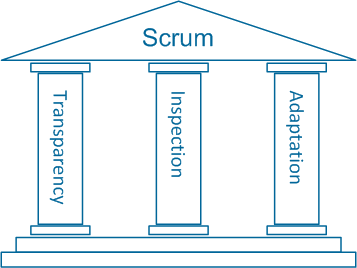
<https://www.edureka.co/blog/interview-questions/software-testing-interview-questions/>



**SCRUM**

Scrum é um **FRAMEWORK** de gerenciamento de projetos ágeis.

Pilares:

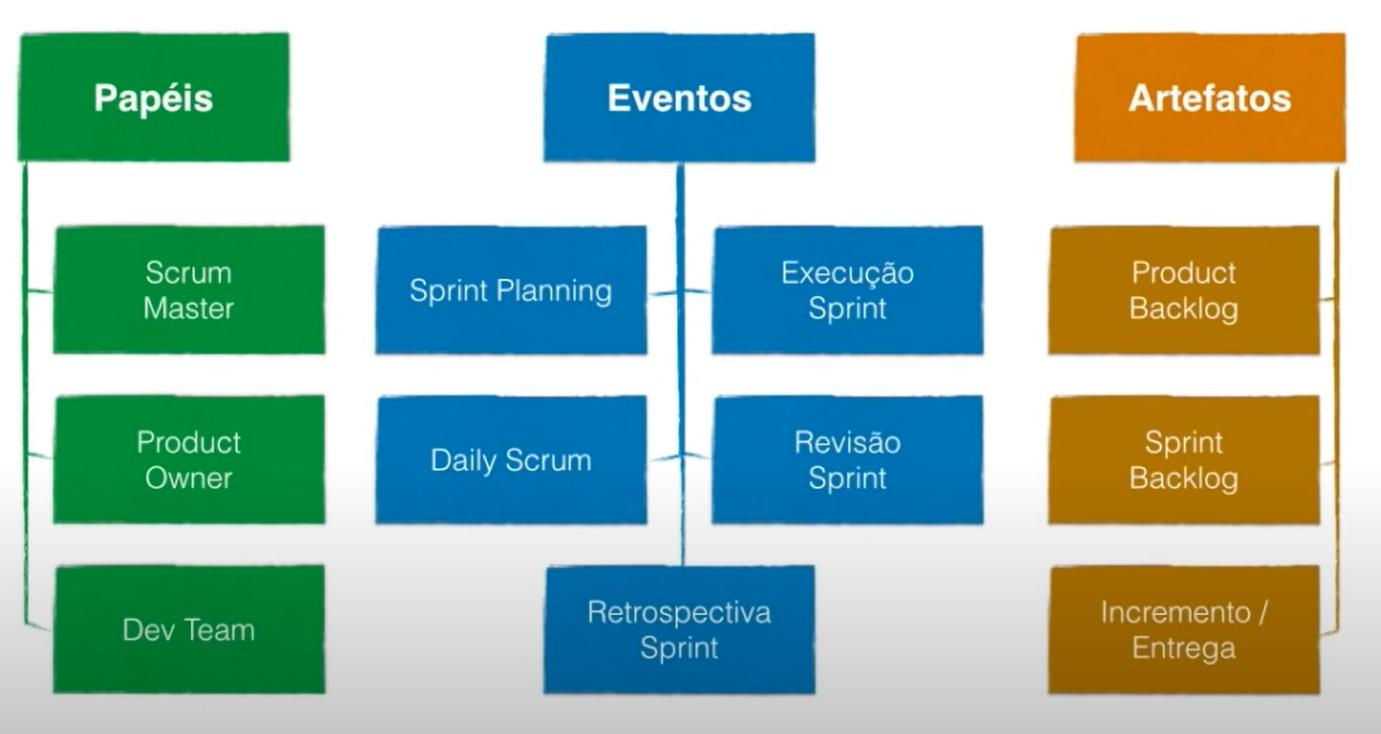


Transparência: é quando todos da equipe tem visão geral do projeto (ex: Status do projeto, atividades dos colegas do time, feedback cliente)

Inspeção: sempre monitorar o desenvolvimento do produto. Isso auxilia num maior controle das atividades e gerenciamento de riscos, cronograma…

Adaptação: justamente ter a capacidade de realizar mudança num cenário controlado no qual será possível minimizar os impactos no cronograma, escopo…

formação:

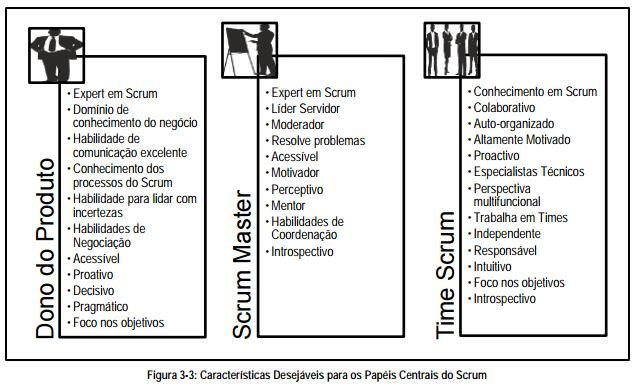


PO - Entende muito do negocio; escreve e organiza os requisitos; mantêm contato direto com o stakeholder

SM - Responsável pelo monitoramento das atividades; Responsável por remover empecilhos;

Dev - O time que desenvolve as funcionalidades através dos requisitos. QA, DEV BE, DEV FRONT, UX, TL …

Papeis



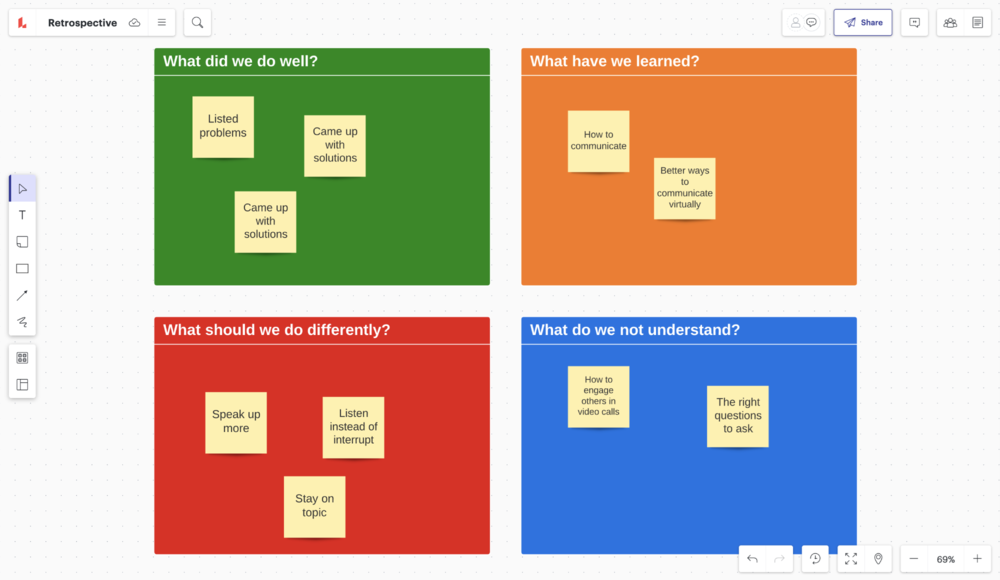
Eventos:

Planning: O time como todo irá analisar as funcionalidades que serão feitas e irão revisá-las. Será feito a votação da complexidade. <https://planningpokeronline.com/F5xnDROlWErljSUkO1NW>

Daily (standup): Reunião diária de no máximo 15min e você fala o que fez, o que tá fazendo e se tem algum impedimento

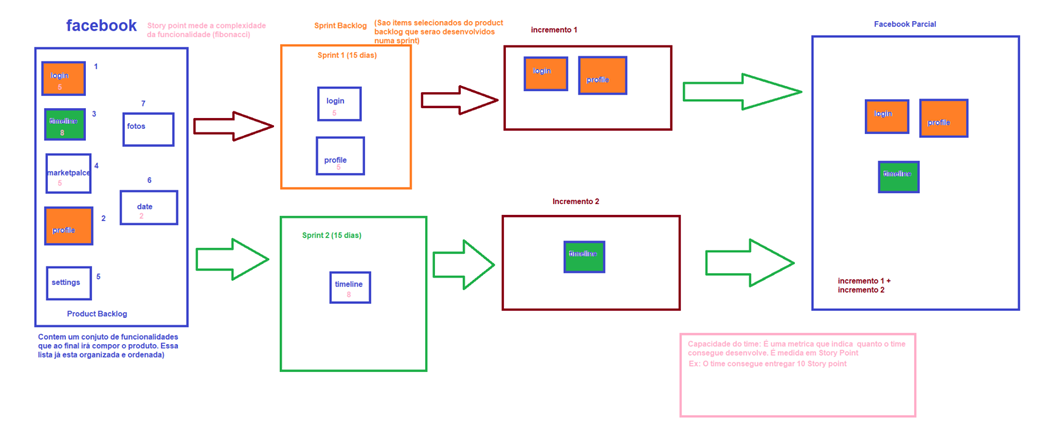
Review: Reunião com os stakeholders para apresentação do resultado (incremento) da sprint. Todo mundo participa (geralmente), mas quem toca é o PO.

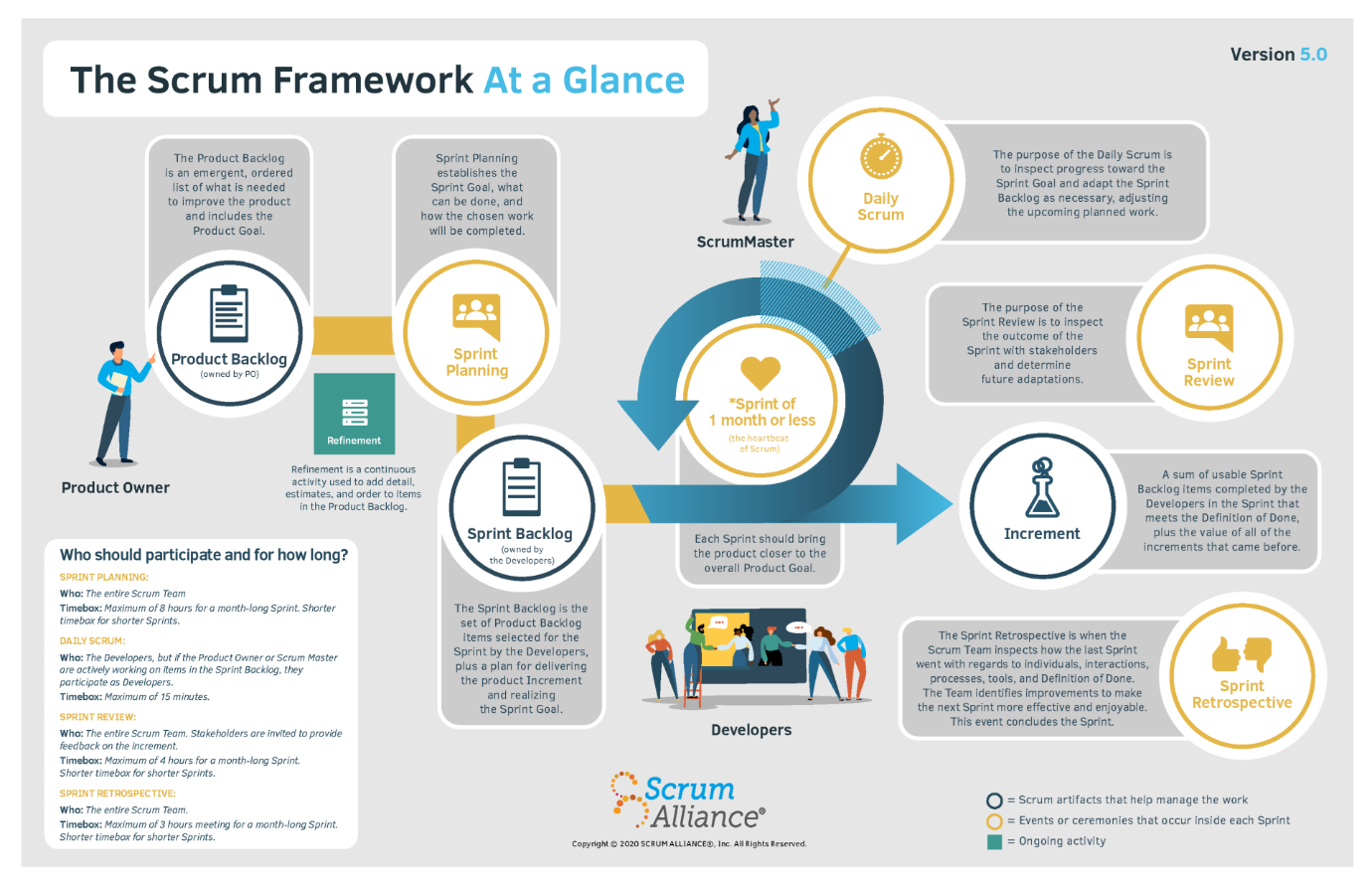
Retrospectiva: Reuniao com o time inteiro do scrum. É analisado o processo em si. Avalia o que foi bom, o que nao foi, o que melhorar…



Fluxo:

Sprint - Uma duração de desenvolvimento (15 dias)



******

[***https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/VER5-scrum-framework\_2020.pdf***](https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/VER5-scrum-framework_2020.pdf)

DOR - Definition of ready

Indica quando uma tarefa está PRONTA para começar a ser desenvolvida:

* Requisitos revisados
* Documentação OK (figma, swagger)

DOD - Definition of done

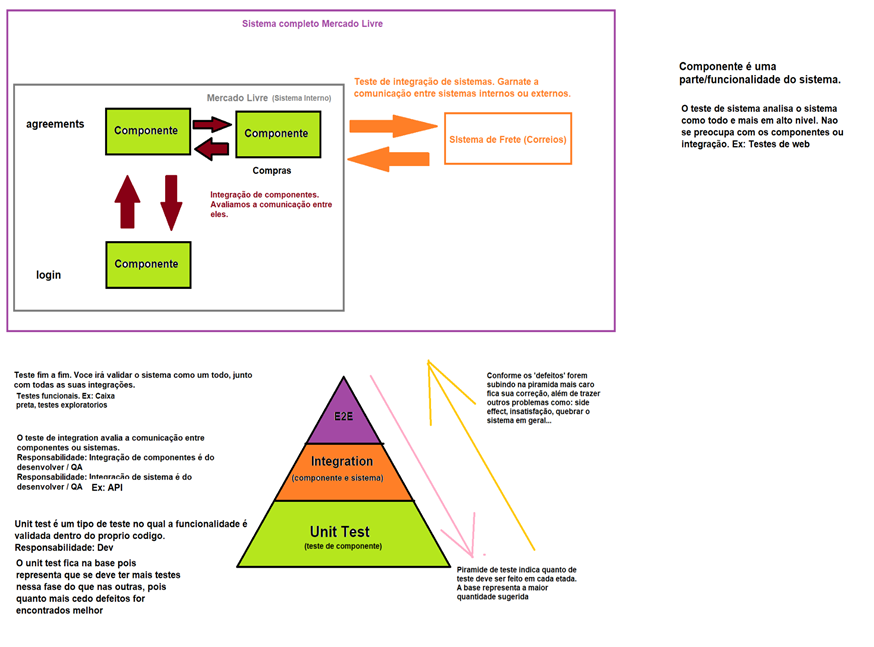
Indica quando a tarefa está PRONTA e pode ser liberado uma release/incremente:

* Todos os requisitos foram desenvolvidos?
* Todos os testes foram executados?
* Nenhum bug crítico está aberto?

<http://paginapessoal.utfpr.edu.br/frufrek/pos-web/p/arquivos/O_manifesto_agil.pdf> estudar -> <https://certiprof.com/pages/scrum-foundations-professional-certificate-sfpc-ptbr> certificao

<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf> guia

**Niveis de testes**



**TDD, ATDD,BDD:**

<https://www.browserstack.com/guide/tdd-vs-bdd-vs-atdd>

Gherkin - É uma linguagem de escrita de requisito

BDD, TDD, ATDD.

TDD (test driven development) - Escreve o teste, o teste falha, dev cria o codigo pro teste passa, o teste passa, testado.

BDD (behavior driven development) - Escreve o TC baseado num comportamento, num cenário.

ATDD (Acceptance test driven development) - Escreve o TC baseado em critérios de aceitação do usuário

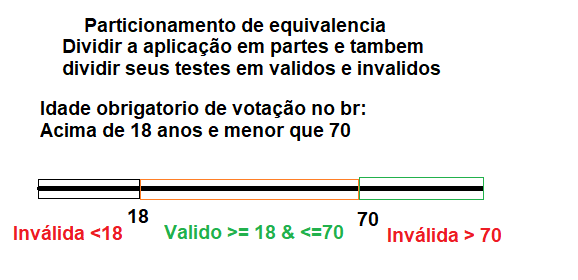
**Tipos de testes**

* Teste Funcional
  + **Validar** o que o sistema **deve fazer**.
  + adicionar item ao carrinho
* Teste Não Funcional
  + **Valida** o **quão bom** o sistema é. Validamos usabilidade, eficiência de performance, segurança… (ex: valida quanto tempo o sistema responde ao completar uma venda. Ex2: Valida se o sistema criptografa a senha do usuário no banco de dados)
    - Segurança (avalia as camadas de segurança de um sistema. Existe um profissional especifico para isso e chama Pen Tester)
    - Performance (avalia o desempenho de um sistema. Jmeter\*\*\*)
    - Usabilidade (avaliar o sistema de acordo com a experiencia do usuario. A cor do do site se adequa ao tipo de sistema (ex: pfizer). tamanho da fonte, em quantos cliques o usuario acessa determinada funcao. (https://uxdesign.blog.br/as-oito-regras-de-ouro-do-design-de-interfaces-836fb166d36b)
    - Compatibilidade (o sistema deve funcionar no chrome, firefox e safari)
* Teste de Caixa Branca
  + é quando voce **tem acesso a estrutura interna do sistema (codigo)** (geralmente o dev que faz). Voce validar estrutura, condições, looping, identação, funcoes… pode ser criados testes unitarios, por exemplo…)
* Teste de Caixa Preta
  + **é quando voce nao tem acesso a estrututra**. Seu teste é baseado em **entrada e saida**
* Teste relacionado a mudança - **Confirmação**
  + Quando voce encontra um **bug** e o dev corrige. Voce precisa retestar.
* Teste relacionado a mudança - **Regressão**
  + é quando voce testa boa parte do sistema (**executa novamente os casos de testes antigo**) para garantir que nada quebrou depois de novas inserções de funcionalidades, bug fixes… No final de cada sprint nos executados testes de regressão, ou sanity ou smoke.

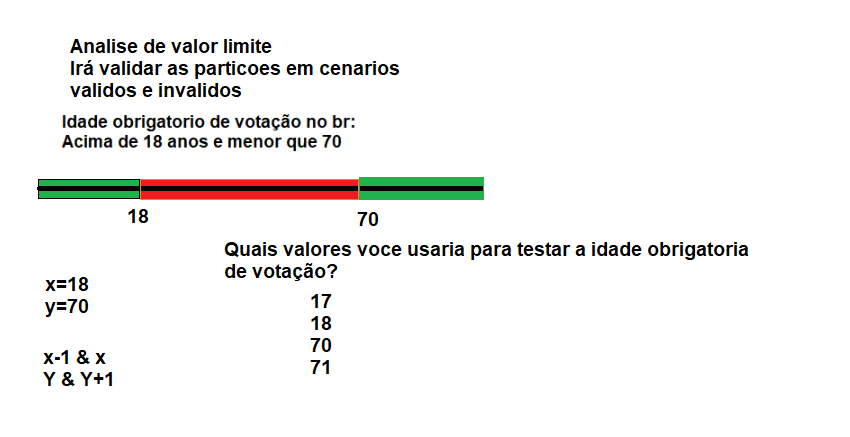
**Técnicas de Testes**

Teste de cx preta

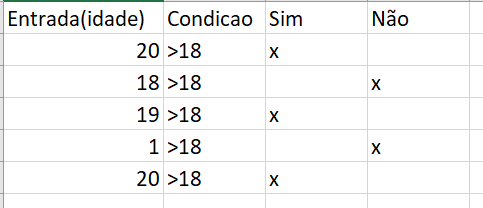
* **particionamento de equivalencia**

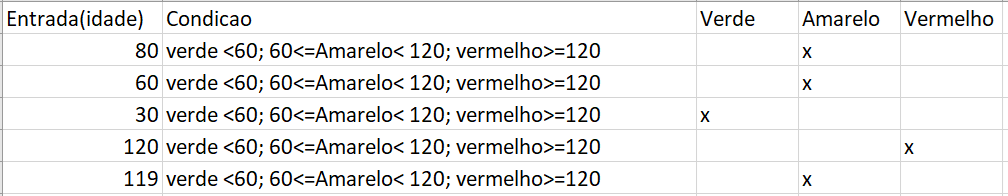
****

* **Analise de valor limite**

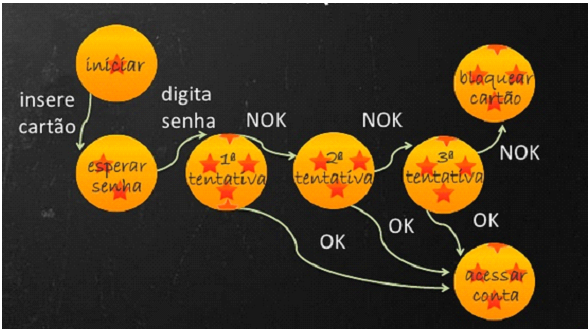
****

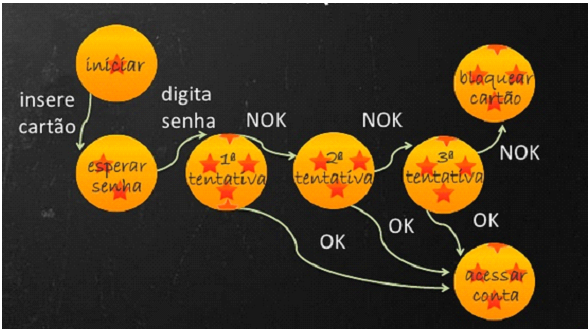
* **Tabela de decisao**

****

****

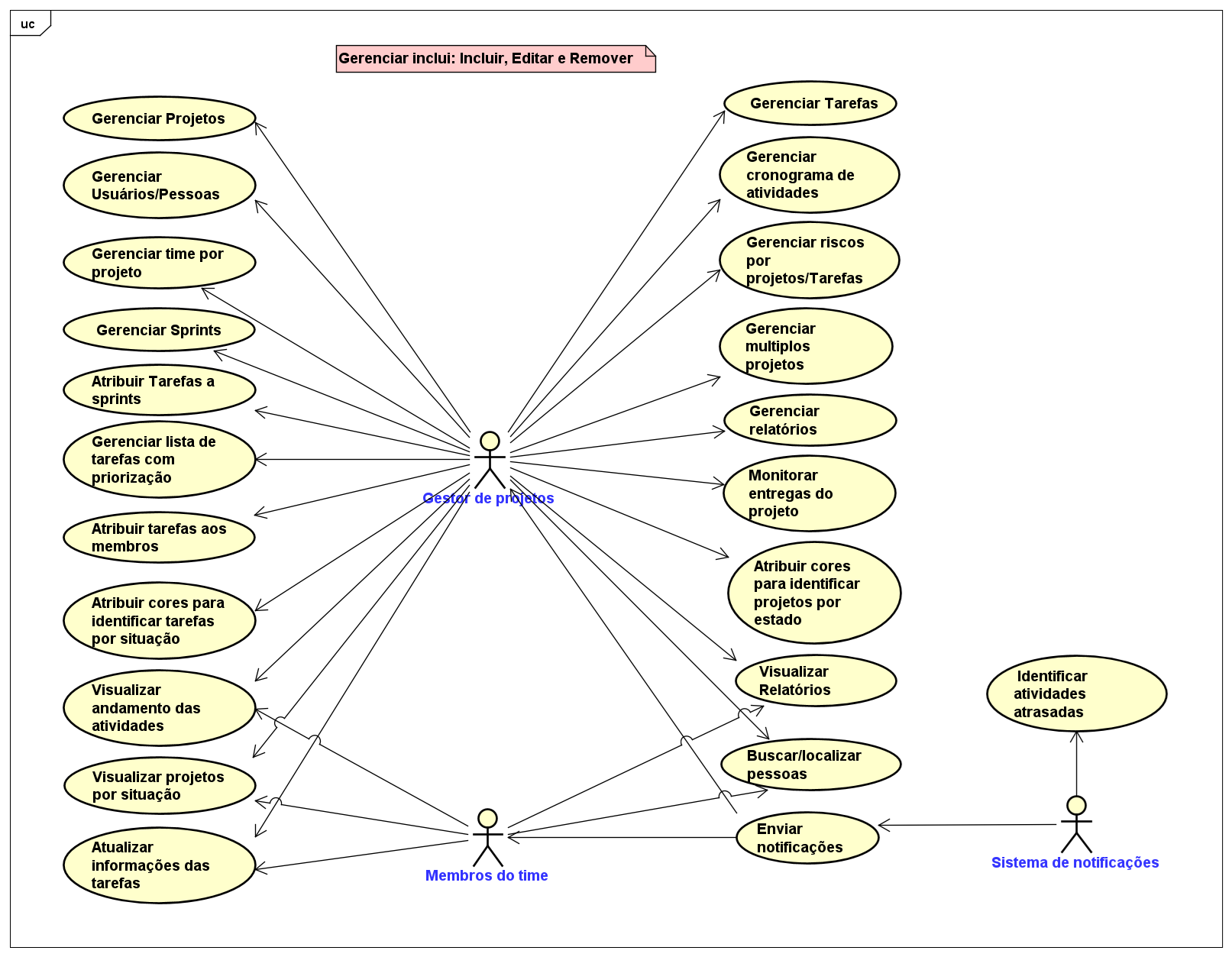
* **Transicao de estado**

****

****

* **Caso de uso (https://dtic.tjpr.jus.br/widget/wiki?p\_p\_id=36&p\_p\_lifecycle=0&p\_p\_state=pop\_up&p\_p\_mode=view&\_36\_struts\_action=%2Fwiki%2Fview&p\_r\_p\_185834411\_nodeName=Governan%C3%A7a-TIC&p\_r\_p\_185834411\_title=Modelo+de+Caso+de+Uso&p\_r\_p\_185834411\_nodeId=6228681&\_36\_viewMode=print)**

**teste baseados em uma ação de um usuario(ator). Em um sistema de vendas, um cliente faz um fluxo diferente do vendedor.**

****

**Teste de caixa-branca (Pular)**

* **cobertura de instrucao**
* **cobertura de decisao**

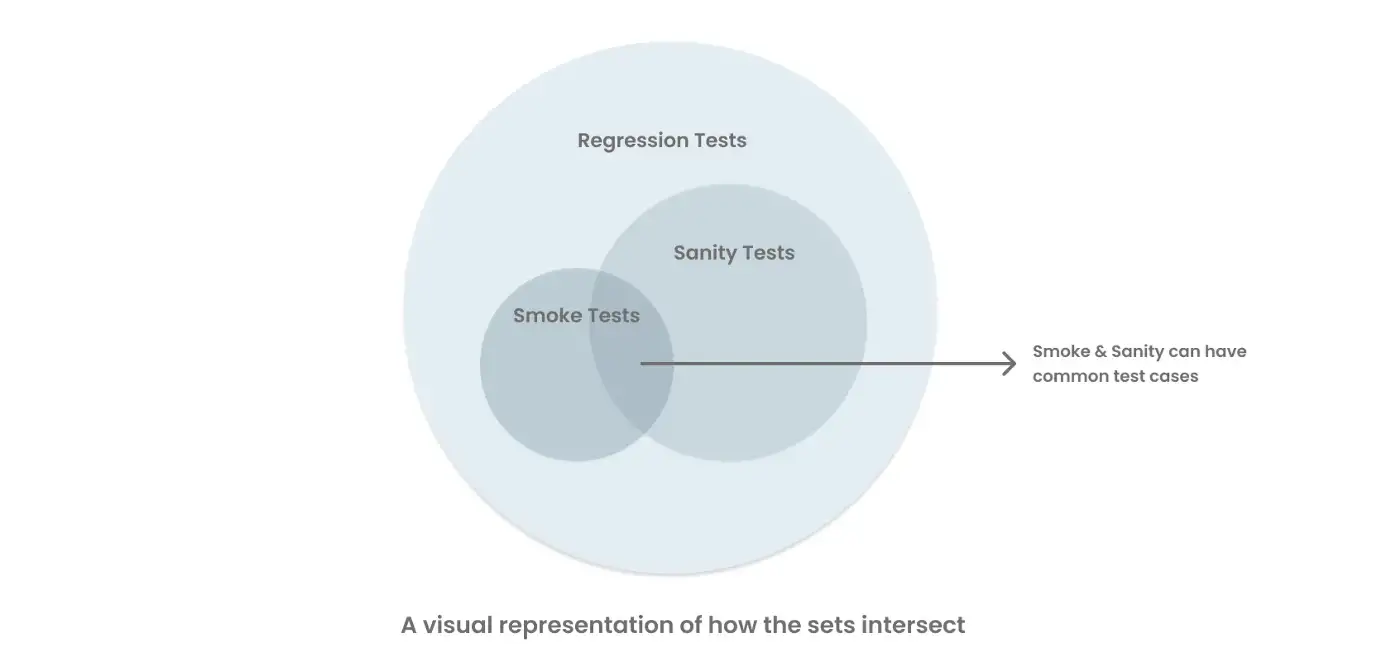
**Tecnicas baseadas na experiência**

* **Suposicao de erro**
  + A suposição de erro é uma técnica usada para prever a ocorrência de erros, defeitos e falhas, com base no conhecimento do testador. Esse conhecimento pode ter sido adquirido em outros sistemas, em erros passados…
* **Exploratorio**
  + Testes aleatórios num sistema no qual o testador não precisa ter um conhecimento prévio da aplicação (clica em qualquer lugar e ve o que acontece)
* **Baseado em Checklist**
  + É quando voce ja tem uma lista pre definida do que testa.
* **ad-hoc**
  + é um teste aleatorio, bem parecido com o exploratorio, a diferenca é que o testador ja tem um conhecimento previo da aplicação.

**Testes relacionado a mudança:**

**São testes que irão verificar se o sistema como um todo esta funcionando corretamente, ou seja, temos que garantir se nada que entrou de novo (funcionalidades, correções de bugs…) não quebrou o que já tava funcionando.**

**A principal diferenca entre os testes de regressao, sanidade e smoke são: Quando e a quantidade de testes que serão executados.**

****

**Como definir os casos de testes entre essas divisões. Pelo objetivo do teste e sua prioridade.**

**Teste de Regression:** É um conjunto grande de testes que valida a aplicacao como um todo, inclui testes de todas as prioridades. É um teste mais longo e que requer tempo para a execucao. Quando houve mudancas em varios lugares e que o risco de ‘side effect’ é grande.

**Teste de Sanity:** É um conjunto com uma quantidade maior de testes e que geralmente é executado por **setores(funcionalidade)** de um sistema.

**Teste de Smoke(teste de build):** São testes que visam validar o caminho critico do sistema, ou seja, **valida as funções criticas que o sistema deve fazer(olha como um todo a aplicacao)**. Testes de prioridade alta geralmente sao adicionado no smoke. Objetivo dele é ser rapido (em questao de execucao dos testes) e dar um feedback rapido sobre como esta o sistema. **Sempre que gerar um build(uma versao) roda ele.**

Qual deles compensa automatizar primeiro?

Smoke! Porque ele será executado diversas vezes e seus testes validam partes cruciais do sistema, então o feedback rápido é essencial.

**Dataset (massa de dados)**

São dados que serão usados no seu teste. Dados positivos e negativos.

Você pode descrever essa massa de teste no teste case ou usar uma planilha com dados, por exemplo. Voce tambem pode gerar automaticamente esses dados atraves de sites.(<https://www.4devs.com.br/>)

em python pode se usar bibliotecas para gerar esses dados <https://faker.readthedocs.io/en/master/>

from faker import Faker

if \_\_name\_\_ == '\_\_main\_\_':

fake = Faker()

print(fake.email())

print(fake.name())

print(fake.address())

print(fake.country())

<https://ncgovote.org/pt/top-50-perguntas-de-entrevista-de-teste-de-software-para-saber-em-2021/>

**HTML/CSS:**

HTML[define a estrutura basica do conteudo web ex: texto, divisoes, botoes...]/CSS[aparência/apresentação cor, tamanho, alinhamento] (faz parte do front end

Utilizar o inspect (dev tools). Voce vera o codigo html e o css.

É muito utilizado para descobrir os IDs dos elementos para num futuro automatizar. Pode se usar algum plugin para auxiliar nessa busca: Letxpath

A aba network mapea todas as chamadas feitas para o backend ou chamadas de outros arquivos do frontend.