Diagrama de Atividade com plantUML

O diagrama de atividade é uma ferramenta valiosa no processo de desenvolvimento de software porque facilita a compreensão dos fluxos de trabalho e das interações entre diferentes partes de um sistema. Abaixo estão algumas razões para sua utilização e o momento ideal para elaborá-lo:

Por que utilizar o diagrama de atividades no desenvolvimento de software?

1. Clarificação de Processos Complexos: Representa de forma visual e sequencial o fluxo de atividades, facilitando a compreensão de processos que envolvem várias etapas, condições e possíveis ramificações.
2. Comunicação Efetiva: Serve como uma linguagem visual comum entre desenvolvedores, analistas, stakeholders e outros membros da equipe, ajudando a alinhar expectativas e a garantir que todos entendam o fluxo do sistema ou do processo.
3. Identificação de Problemas e Otimização: Ajuda a identificar gargalos, redundâncias e pontos de falha em processos, permitindo melhorias antes da implementação.
4. Documentação e Manutenção: Atua como um registro que pode ser utilizado durante a fase de manutenção, oferecendo uma visão clara do fluxo de atividades que pode auxiliar em futuras atualizações ou correções no sistema.

Quando elaborar o diagrama de atividades?

O diagrama de atividade é geralmente criado nas fases iniciais do ciclo de desenvolvimento, principalmente nas etapas de Análise e Design. Veja em quais momentos específicos ele é útil:

1. Fase de Levantamento de Requisitos:
   - Durante a análise de requisitos, o diagrama de atividades pode ser usado para capturar e representar o fluxo dos requisitos funcionais. Isso é especialmente útil para requisitos que envolvem várias etapas ou interações complexas.

2. Modelagem de Casos de Uso:
   - O diagrama de atividades é útil para detalhar o comportamento de um caso de uso específico, descrevendo o fluxo das ações e decisões que ocorrem dentro do caso de uso. Ele complementa os diagramas de casos de uso ao oferecer uma visão mais detalhada do processo.

3. Planejamento de Arquitetura e Design:
   - Ao projetar a arquitetura do sistema, o diagrama de atividades pode ajudar a definir as interações entre componentes e serviços, esclarecendo as responsabilidades de cada um e como eles se comunicam.

4. Planejamento de Fluxo de Trabalho e Processos Internos:
   - Pode ser útil para modelar fluxos de trabalho específicos dentro do sistema, como um processo de aprovação ou um pipeline de processamento de dados, o que é importante para sistemas de automação.

5. Fase de Testes:
   - Pode ser usado para planejar cenários de teste, uma vez que o diagrama mostra todos os possíveis caminhos de execução, inclusive condições e alternativas que podem ser verificadas por meio de testes.

6. Revisão e Validação do Sistema:
   - Antes da implementação ou da fase final de testes, o diagrama de atividades pode ser revisado com stakeholders para validar o fluxo de processos e garantir que todas as funcionalidades foram cobertas.

Utilizar o diagrama de atividades no momento certo permite melhorar a qualidade do desenvolvimento e a comunicação entre os envolvidos, ajudando a garantir que o software atenda às necessidades e funcione conforme o esperado.

Passo a passo para a elaboração de um diagrama de atividades:

Diagrama de Atividade com plantUML

1. Defina o Escopo e Objetivo
   - Determine o processo ou fluxo que o diagrama irá representar. Isso pode ser um processo de negócio, um fluxo de tarefas dentro de um sistema ou um conjunto de passos dentro de uma funcionalidade específica.

2. Identifique as Ações e Atividades
   - Liste todas as atividades principais que precisam ser executadas no processo. Cada atividade deve ser uma ação específica que move o processo adiante.

3. Organize as Atividades em Ordem Lógica
   - Coloque as atividades na sequência correta, conforme elas ocorrerão no fluxo. Lembre-se de considerar dependências ou sequências obrigatórias entre as atividades.

4. Defina os Pontos de Início e Término
   - Identifique o ponto inicial (nó inicial) e o ponto final (nó final) do diagrama. Esses pontos indicam o começo e o término do processo ou fluxo.

5. Adicione Decisões e Ramificações
   - Identifique pontos no fluxo onde decisões precisam ser tomadas. Para isso, adicione **nós de decisão** (diamantes) que indicam diferentes caminhos, baseados em condições.

6. Conecte as Atividades com Fluxos de Controle
   - Conecte as atividades, decisões e nós de início e término usando setas, que representam o fluxo de controle entre as atividades.

7. Inclua Nós de Junção e Divisão, se Necessário
   - Se o processo exige que atividades ocorram em paralelo ou que duas atividades paralelas se juntem, use nós de divisão (barra horizontal) e junção para representar esses casos.

8. Represente Objetos e Artefatos, se Relevante
   - Se o diagrama de atividades precisa representar objetos específicos ou artefatos utilizados ou produzidos durante o processo, adicione **nós de objeto** e conecte-os às atividades apropriadas.

9. Revise e Valide o Diagrama
   - Confirme se todas as atividades, decisões, fluxos de controle e objetos foram incluídos e estão em uma sequência lógica. Valide o diagrama com outros envolvidos no projeto para garantir a precisão.

10. Documente e Organize o Diagrama

Diagrama de Atividade com plantUML

- Acrescente legendas ou notas, se necessário, para facilitar a compreensão. Organize o diagrama de maneira visualmente clara e coerente para facilitar a leitura.

Diagrama de Atividade com plantUML

Ação simples

O rótulo Activities começa com : e termina com ;.

@startuml
:Hello world;
:Este é definido em
*várias **linhas**;*
@enduml
Start/Stop/End

Você pode usar palavras-chave start e stop para denotar o início e o fim de um diagrama.

@startuml
start
:Hello world;
:Este é definido em
*várias **linhas**;*
stop
@enduml

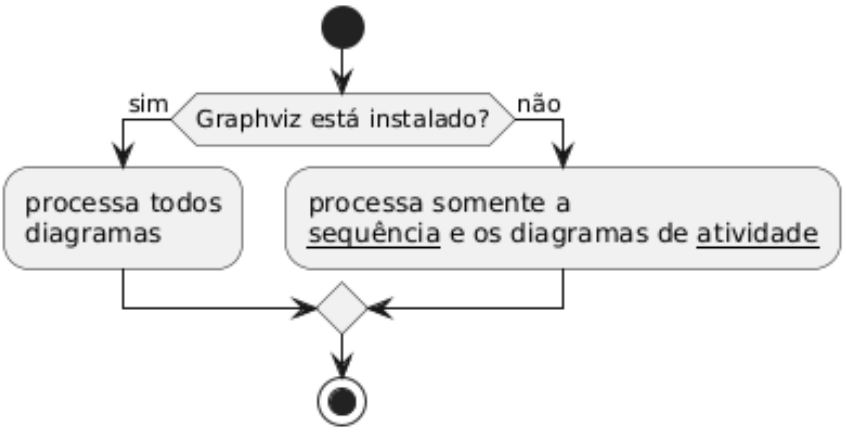
@startuml
start
:Hello world;
:Este é definido em
*várias **linhas**;*
end
@enduml

Diagrama de Atividade com plantUML

if, then, else e endif

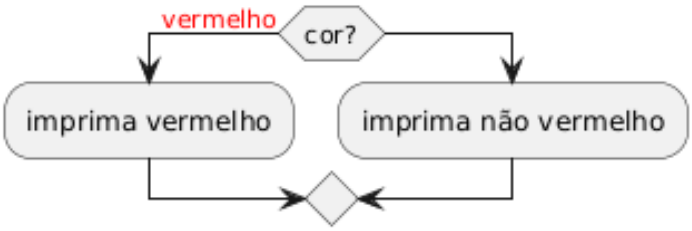
As 3 sintaxes são possíveis: Se (...) então (...) ... [else (...) ...] endif

@startuml
start
if (Graphviz está instalado?) then (sim)
* :processa todos\ndiagramas;*
else (não)
* :processa somente a*
* __sequência__ e os diagramas de __atividade__;*
endif
stop
@enduml



if (...) is (...) then ... [else (...) ...] endif

@startuml
if (cor?) is (<color:red>vermelho) then
:imprima vermelho;
else
:imprima não vermelho;
endif
@enduml



if (...) equals (...) then ... [else (...) ...] endif

@startuml
if (contador?) equals (5) then
:imprima 5;
else
:imprima != 5;
endif
@enduml

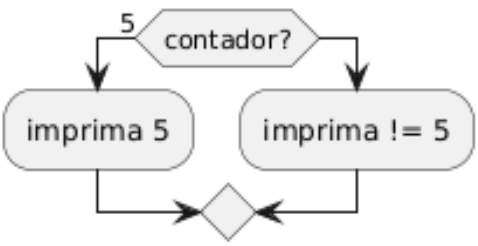
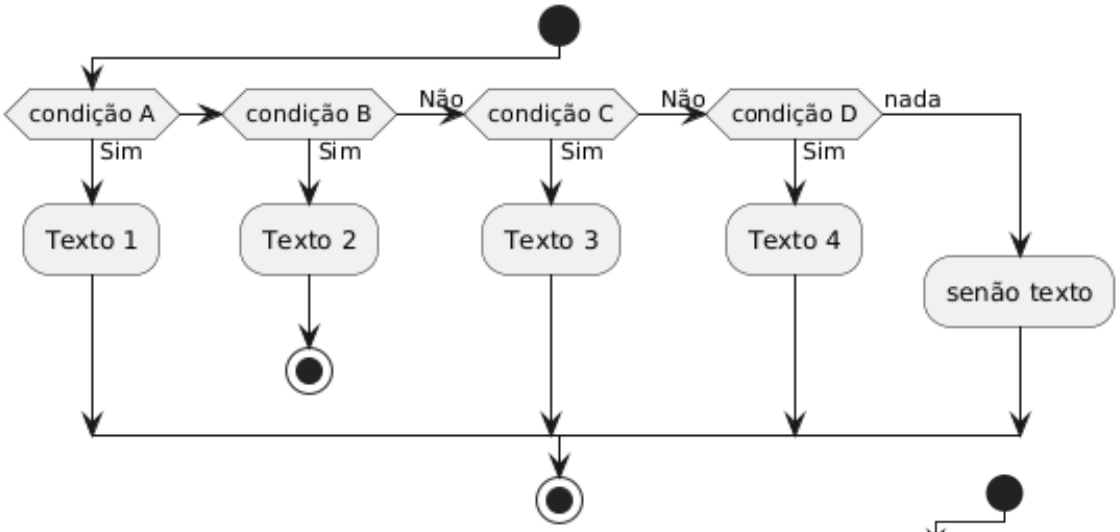


Diagrama de Atividade com plantUML

Vários testes (modo horizontal):

```
@startuml
start
if (condição A) then (Sim)
  :Texto 1;
elseif (condição B) then (Sim)
  :Texto 2;
  stop
(Não) elseif (condição C) then (Sim)
  :Texto 3;
(Não) elseif (condição D) then (Sim)
  :Texto 4;
else (nada)
  :senão texto;
endif
stop
@enduml
```



Vários testes (modo vertical)

```
@startuml
!pragma useVerticalIf on
start
if (condição A) then (sim)
  :Texto 1;
elseif (condição B) then (sim)
  :Texto 2;
  stop
elseif (condição C) then (sim)
  :Texto 3;
elseif (condição D) then (sim)
  :Texto 4;
else (nada)
  :senão Texto;
endif
stop
@enduml
```

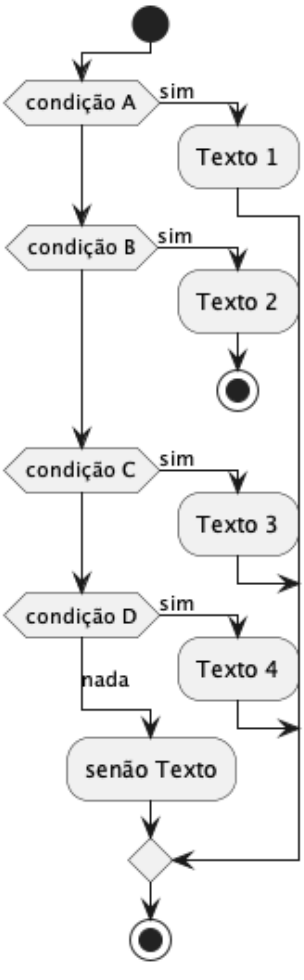
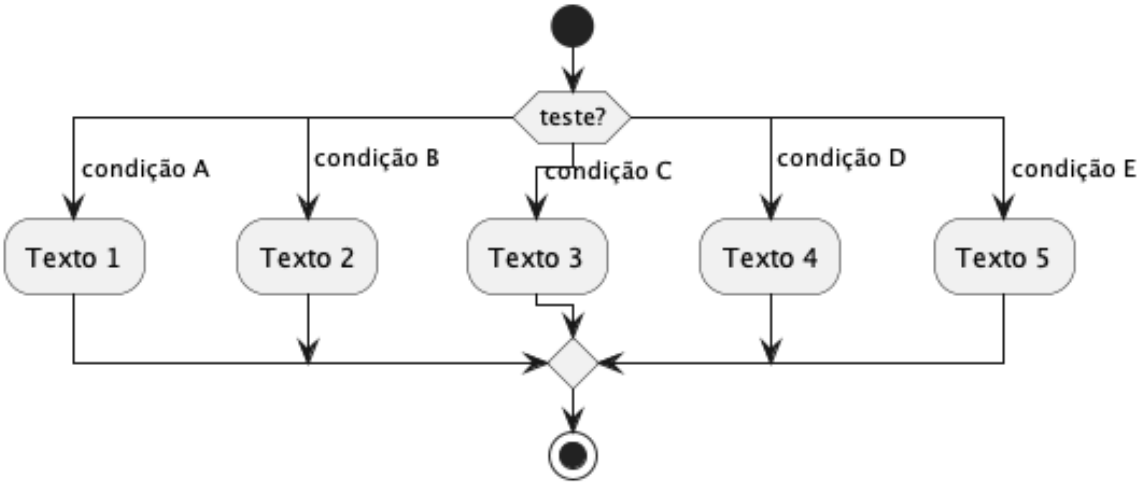


Diagrama de Atividade com plantUML

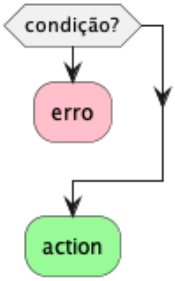
switch, case, endswitch

```
@startuml
start
switch (teste?)
case ( condição A )
:Texto 1;
case ( condição B )
:Texto 2;
case ( condição C )
:Texto 3;
case ( condição D )
:Texto 4;
case ( condição E )
:Texto 5;
endswitch
stop
@enduml
```



Condicional com stop sobre uma ação [kill, detach]

```
@startuml
if (condição?) then
  #pink:erro;
  kill
endif
#palegreen:action;
@enduml
```




```
@startuml
if (condição?) then
  #pink:erro;
  detach
endif
#palegreen:action;
@enduml
```

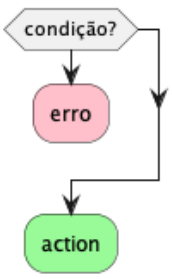


Diagrama de Atividade com plantUML

Repetição [repeat]

```
@startuml
start
repeat
  :leia o dado;
  :gera os diagramas;
repeat while (mais dado?) is (sim) not (não)
stop
@enduml
```

Loop com ações de repeat e backward

```
@startuml
start
repeat :foo como rótulo inicial;
  :leia dado;
  :gere os diagramas;
backward:Este é o voltar;
repeat while (mais dado?) is (sim)
->não;
stop
@enduml
```

Diagrama de Atividade com plantUML

Break sobre um repeat [break]

```
@startuml
start
repeat
  :Teste alguma coisa;
    if (Algo deu errado?) then (não)
      #palegreen:OK;
      break
    endif
    ->Não OK;
    :Alerta "Erro com texto longo";
repeat while (Algo deu errado com o texto longo?) is (sim) not (não)
->//Passo mesclado//;
:Alerta "Sucesso";
stop
@enduml
```

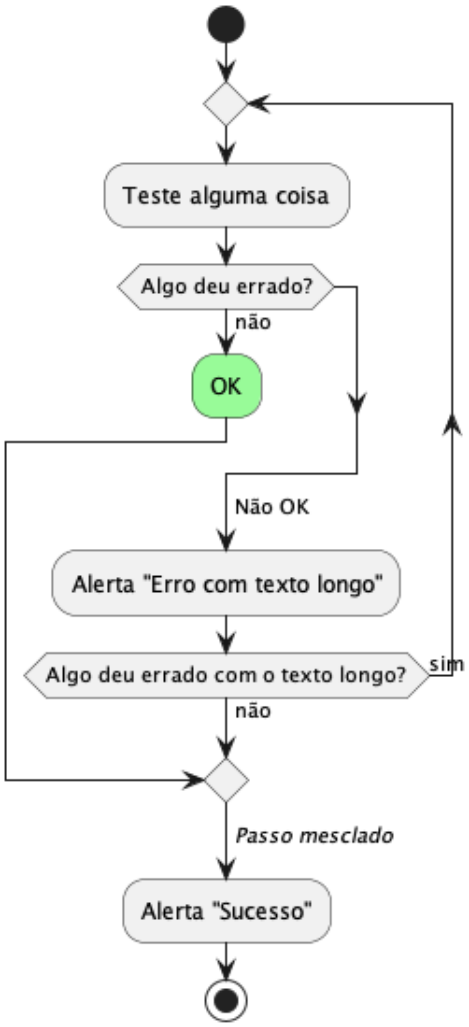
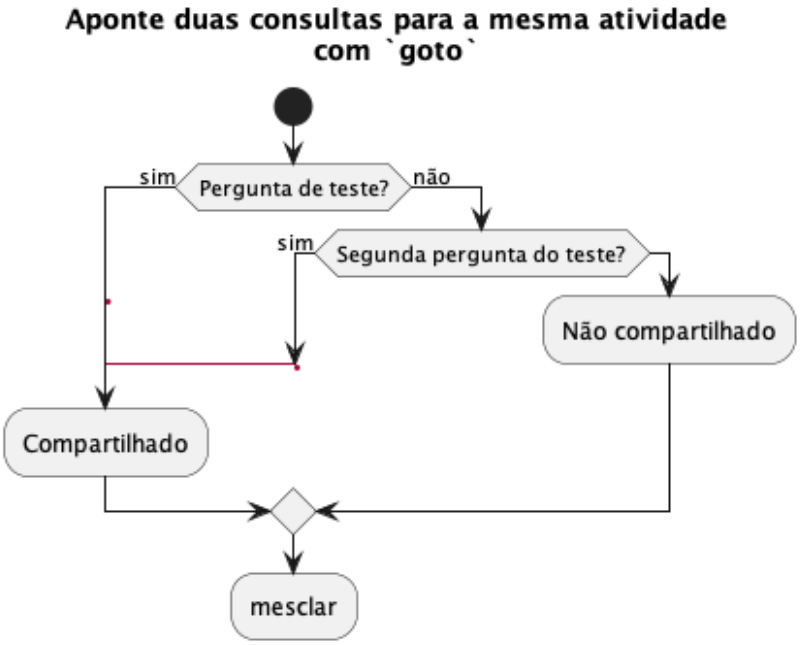


Diagrama de Atividade com plantUML

goto, label

```
@startuml
title Aponte duas consultas para a mesma atividade\ncom `goto`
start
if (Pergunta de teste?) then (sim)
'label de espaço apenas para alinhamento
label sp_lab0
label sp_lab1
'label real
label lab
:Compartilhado;
else (não)
if (Segunda pergunta do teste?) then (sim)
label sp_lab2
goto sp_lab1
else
:Não compartilhado;
endif
endif
:mesclar;
@enduml
```



while simples

```
 @startuml
start
while (Dados disponíveis?)
  :Leia Dados;
  :Gerar diagramas;
endwhile
stop
@enduml
```

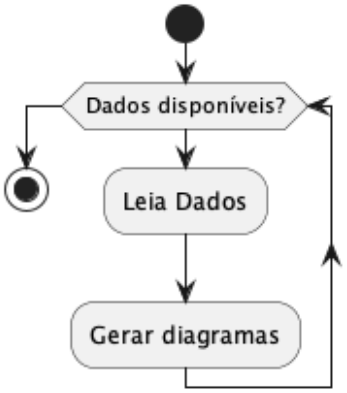
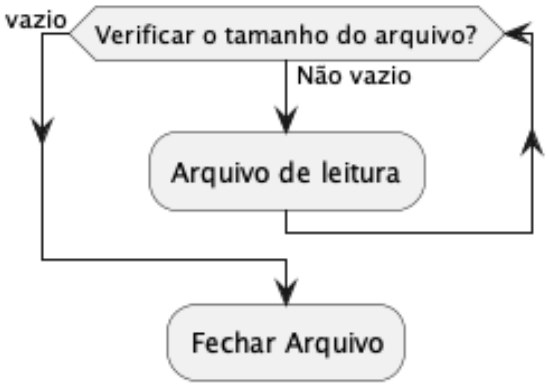
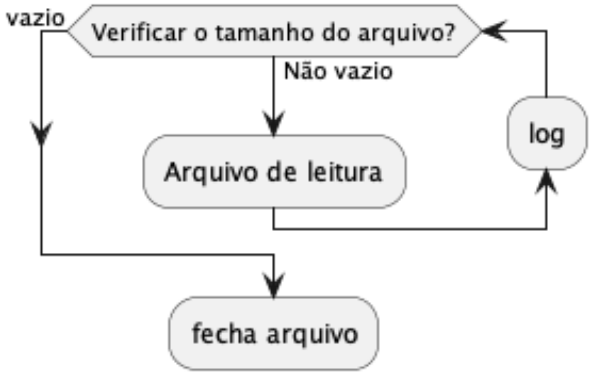


Diagrama de Atividade com plantUML

```
@startuml
while (Verificar o tamanho do arquivo?) is (Não vazio)
  :Arquivo de leitura;
endwhile (vazio)
:Fechar Arquivo;
@enduml
```




```
@startuml
while (Verificar o tamanho do arquivo?) is (Não vazio)
  :Arquivo de leitura;
  backward:log;
endwhile (vazio)
:fecha arquivo;
@enduml
```




```
@startuml
:Passo 1;
if (condição 1) then
  while (loop infinito)
   :Passo 2;
  endwhile
  -[hidden]->
  detach
else
  :finaliza normalmente;
  stop
endif
@enduml
```

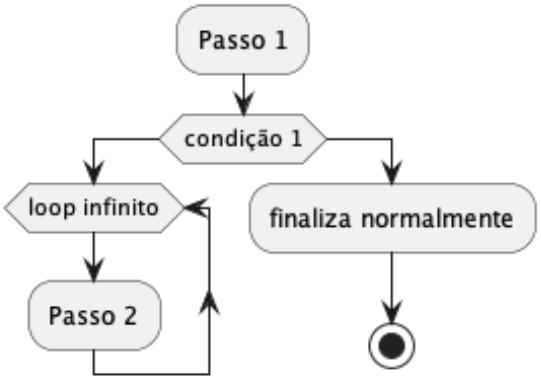
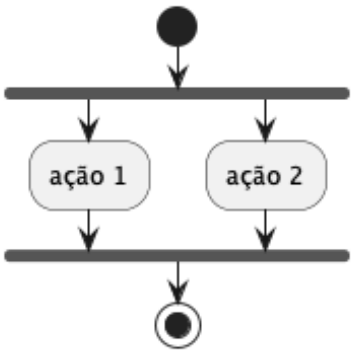


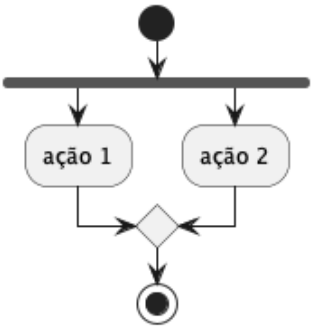
Diagrama de Atividade com plantUML

Processamento paralelo [fork, fork again, end fork, end merge]

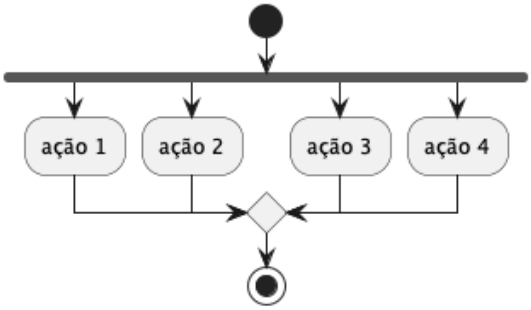
```
@startuml
start
fork
  :ação 1;
fork again
  :ação 2;
end fork
stop
@enduml
```



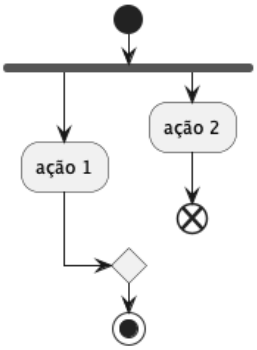

```
@startuml
start
fork
  :ação 1;
fork again
  :ação 2;
end merge
stop
@enduml
```




```
@startuml
start
fork
  :ação 1;
fork again
  :ação 2;
fork again
  :ação 3;
fork again
  :ação 4;
end merge
stop
@enduml
```




```
@startuml
start
fork
  :ação 1;
fork again
  :ação 2;
  end
end merge
stop
@enduml
```




```
@startuml
start
fork
  :ação A;
fork again
  :ação B;
end fork {ou}
stop
```

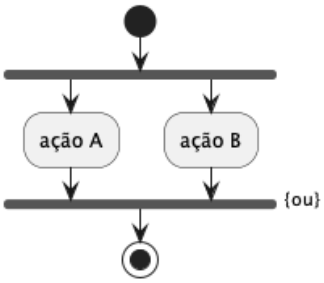
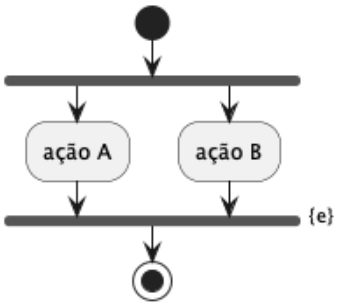


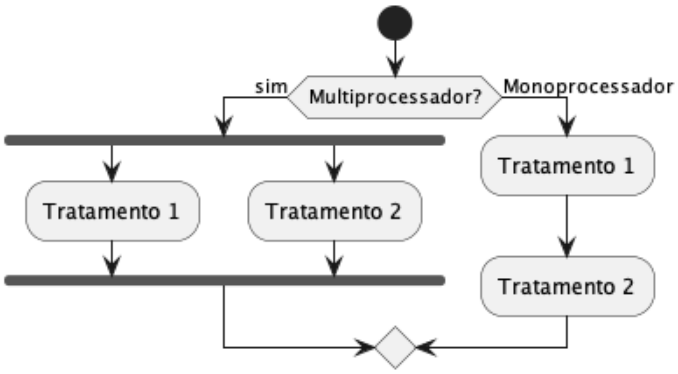
Diagrama de Atividade com plantUML

```
@enduml


@startuml
start
fork
  :ação A;
fork again
  :ação B;
end fork {e}
stop
@enduml
```

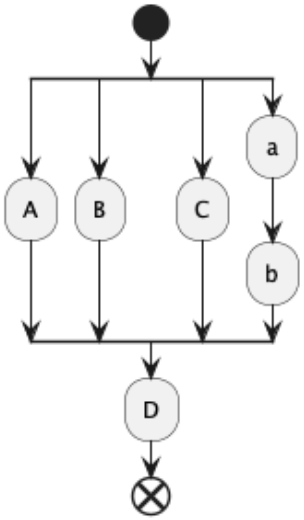



```
@startuml
start
if (Multiprocessador?) then (sim)
  fork
    :Tratamento 1;
  fork again
    :Tratamento 2;
  end fork
else (Monoprocessador)
  :Tratamento 1;
  :Tratamento 2;
endif
@enduml
```



split, split again e end split para denotar a divisão de processos

```
@startuml
start
split
   :A;
split again
   :B;
split again
   :C;
split again
   :a;
   :b;
end split
:D;
end
@enduml
```




```
@startuml
split
   -[hidden]->
   :A;
split again
   -[hidden]->
   :B;
split again
   -[hidden]->
   :C;
end split
:D;
```

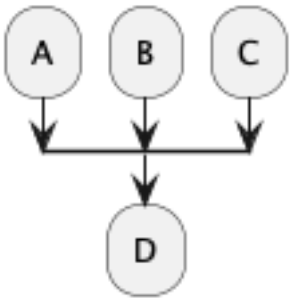
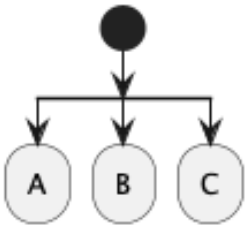
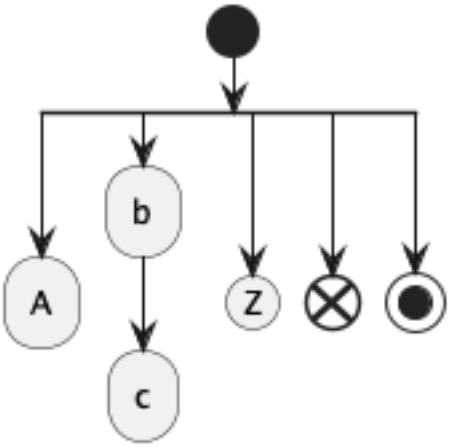


Diagrama de Atividade com plantUML

```
@enduml
@startuml
start
split
   :A;
   kill
split again
   :B;
   detach
split again
   :C;
   kill
end split
@enduml
```




```
@startuml
start
split
   :A;
   kill
split again
   :b;
   :c;
   detach
split again
   (Z)
   detach
split again
   end
split again
   stop
end split
@enduml
```




```
@startuml
start
:foo1;
floating note left: Esta é uma nota
:foo2;
note right
  Esta nota está em várias
  //linhas// e pode
  conter <b>HTML</b>
  ====
  * Chamar o método ""foo()"" é proibido
end note
stop
@enduml
```

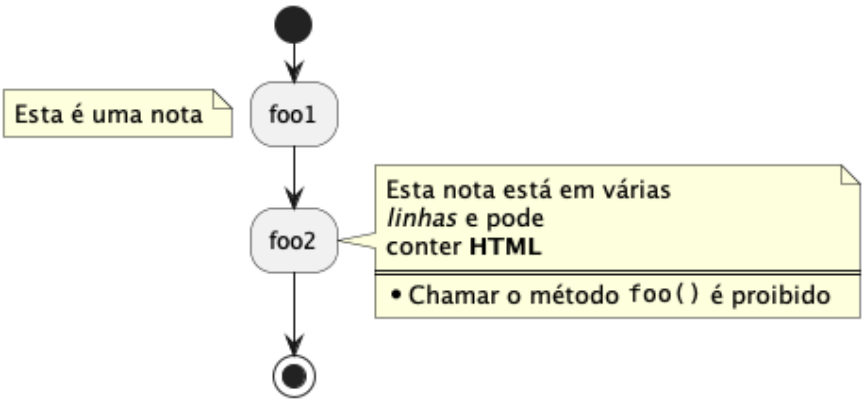
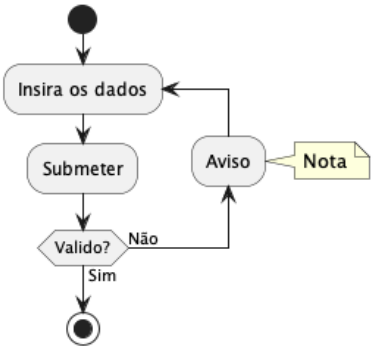
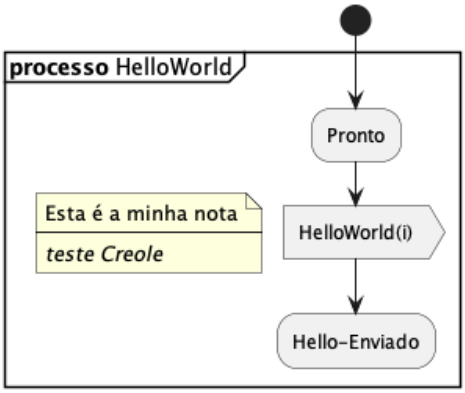


Diagrama de Atividade com plantUML

```
@startuml
start
repeat :Insira os dados;
:Submeter;
backward :Aviso;
note right: Nota
repeat while (Valido?) is (Não) not (Sim)
stop
@enduml
```




```
@startuml
start
partition "**processo** HelloWorld" {
    note
        Esta é a minha nota
        ----
        //teste Creole//
    end note
    :Pronto;
    :HelloWorld(i)>
    :Hello-Enviado;
}
@enduml
```



Cores

```
@startuml
start
:Começando o progresso;
#HotPink:Lendo arquivos de configuração
Esses arquivos devem ser editados neste momento!;
#AAAAAA:Fim do processo;
@enduml
```

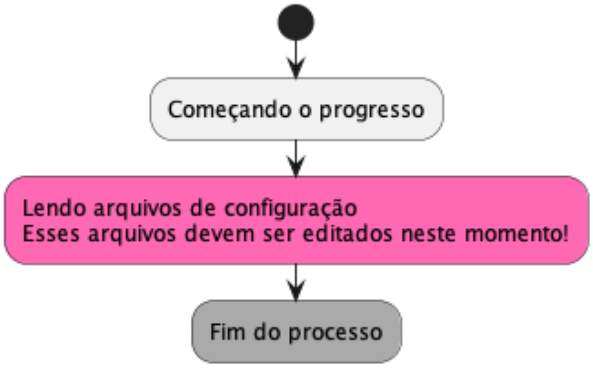
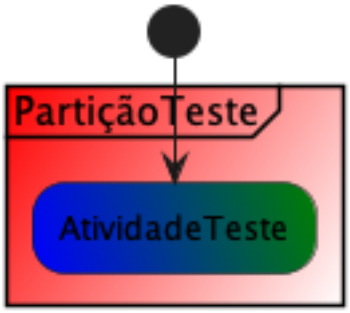


Diagrama de Atividade com plantUML

```
@startuml
start
partition #red/white PartiçãoTeste {
#blue\green:AtividadeTeste;
}
@enduml
```



Linhas sem setas

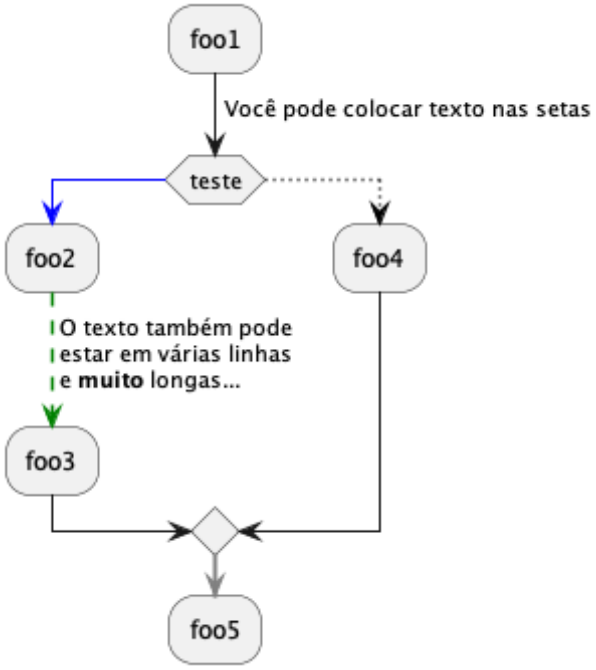
```
@startuml
skinparam ArrowHeadColor none
start
:Hello world;
:Isso está definido em
várias **linhas**;
stop
@enduml
```


```
@startuml
skinparam ArrowHeadColor none
start
repeat :Insira os dados;
:Submeter;
backward :Aviso;
repeat while (Valido?) is (Não) not (Sim)
stop
@enduml
```

Diagrama de Atividade com plantUML

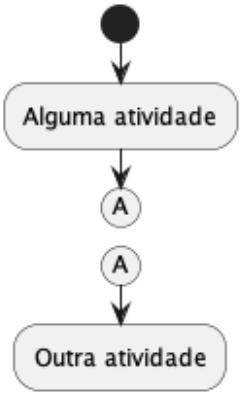
Setas

```
@startuml
:foo1;
-> Você pode colocar texto nas setas;
if (teste) then
-[#blue]->
:foo2;
-[#green,dashed]-> O texto também pode
estar em várias linhas
e **muito** longas...;
:foo3;
else
-[#black,dotted]->
:foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
```



Conector

```
@startuml
start
:Alguma atividade;
(A)
detach
(A)
:Outra atividade;
@enduml
```



Cor no conector

```
start
:O conector abaixo
Gostaria que ele fosse azul;
#blue:(B)
:Este próximo conector sente
 que ela estaria melhor
fora do verde;
#green:(G)
stop
@enduml
```

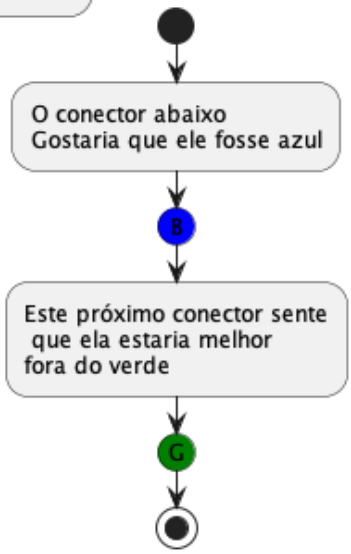
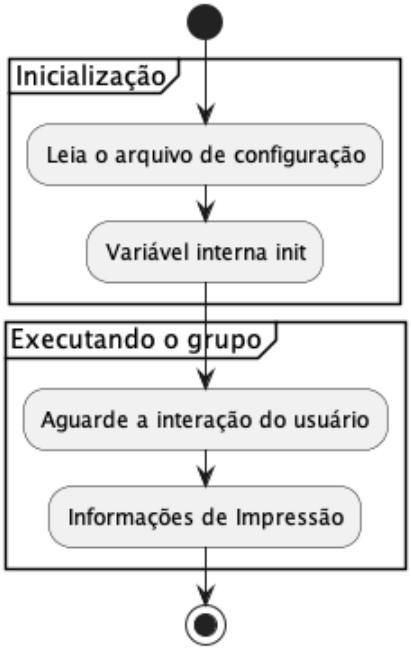


Diagrama de Atividade com plantUML

Agrupamento ou particionamento

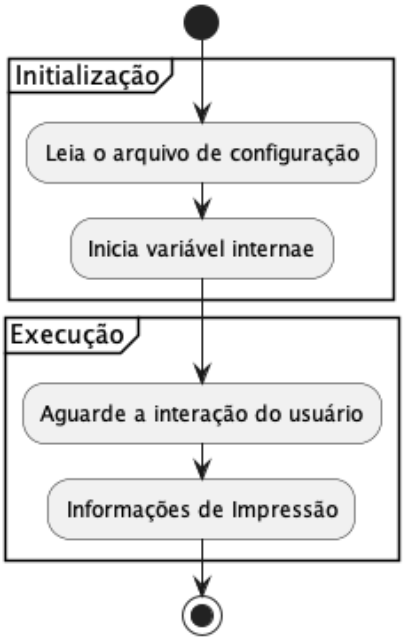
Agrupamento

```
@startuml
start
group Inicialização
:Leia o arquivo de configuração;
:Variável interna init;
end group
group Executando o grupo
:Aguarde a interação do usuário;
:Informações de Impressão;
end group
stop
@enduml
```



Particionamento

```
@startuml
start
partition Initialização {
:Leia o arquivo de configuração;
:Inicia variável internae;
}
partition Execução {
:Aguarde a interação do usuário;
:Informações de Impressão;
}
stop
@enduml
```




```
@startuml
start
partition #lightGreen "Interface de Entrada" {
:Leia o arquivo de configuração;
:Iniciar Variável interna;
}
partition Execução {
:Aguardar por interação do usuário;
:Imprimir informação;
}
stop
@enduml
```

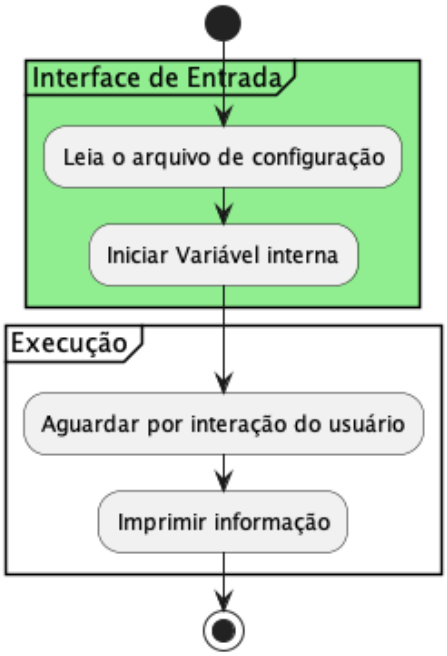
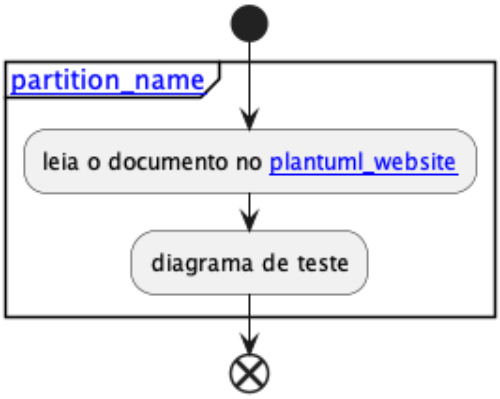


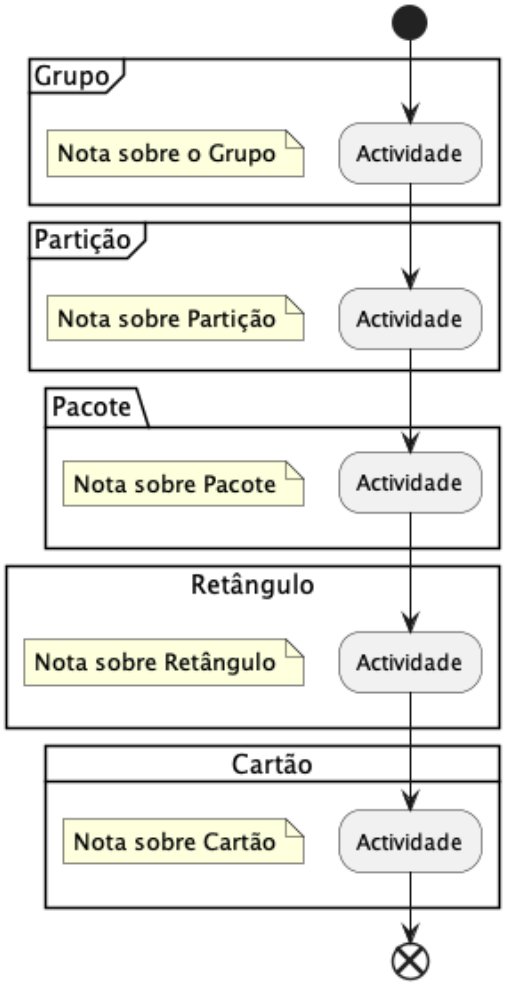
Diagrama de Atividade com plantUML

```
@startuml
start
partition "[[http://plantuml.com partition_name]]" {
:leia o documento no [[http://plantuml.com plantuml_website]];
:diagrama de teste;
}
end
@enduml
```



Grupo, partição, Pacote, Retângulo ou Cartão

```
@startuml
start
group Grupo
:Actividade;
end group
floating note: Nota sobre o Grupo
partition Partição {
:Actividade;
}
floating note: Nota sobre Partição
package Pacote {
:Actividade;
}
floating note: Nota sobre Pacote
rectangle Retângulo {
:Actividade;
}
floating note: Nota sobre Retângulo
card Cartão {
:Actividade;
}
floating note: Nota sobre Cartão
end
@enduml
```



"Faixas de natação" Swimlanes

Usando pipe |, você pode definir swimlanes. Também é possível mudar a cor das pistas de natação.

```
@startuml
|Faixa 1|
start
:foo1;
|#AntiqueWhite|Faixa 2|
:foo2;
:foo3;
|Faixa 1|
:foo4;
|Faixa 2|
:foo5;
stop
@enduml
```

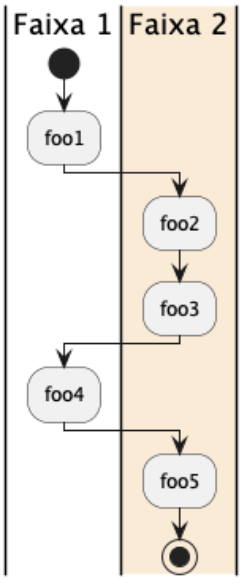
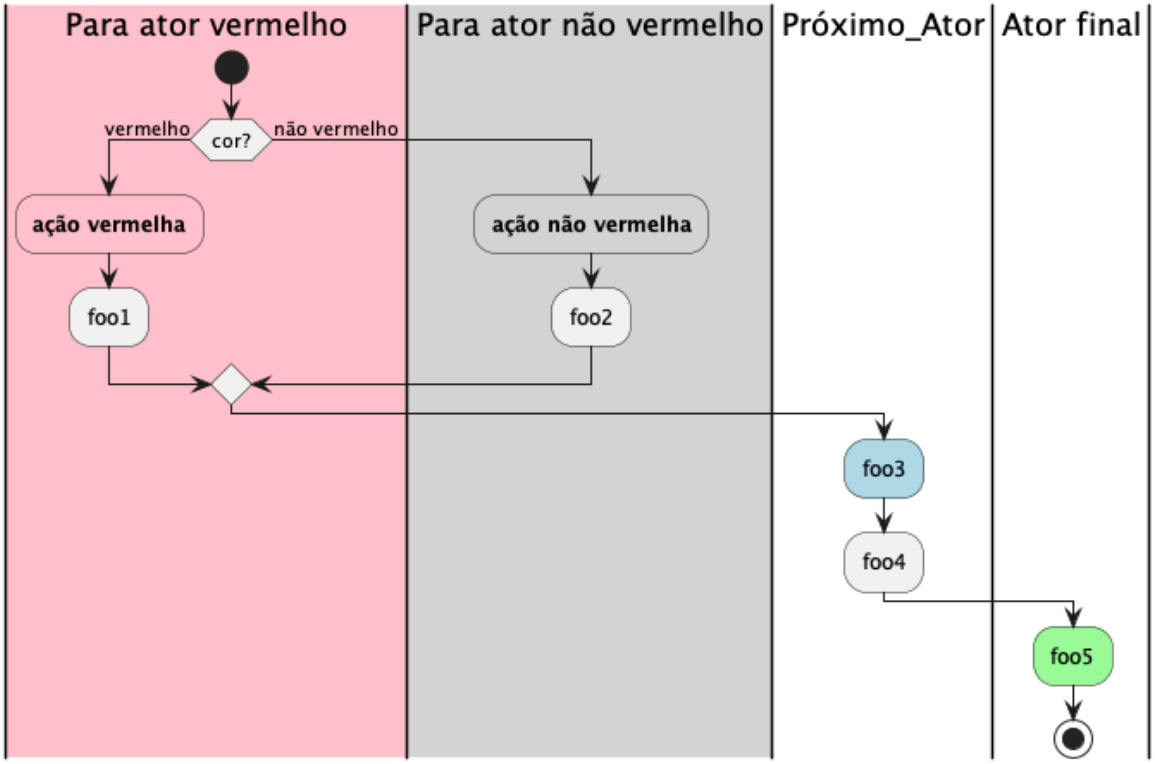


Diagrama de Atividade com plantUML

```
@startuml
|#pink|Para ator vermelho|
start
if (cor?) is (vermelho) then
#pink:**ação vermelha**;
:foo1;
else (não vermelho)
|#lightgray|Para ator não vermelho|
#lightgray:**ação não vermelha**;
:foo2;
endif
|Próximo_Ator|
#lightblue:foo3;
:foo4;
|Ator final|
#palegreen:foo5;
stop
@enduml
```




```
@startuml
|#palegreen|p| Pescador
|c| Cozinheiro
|#gold|f| Faminto
|p|
start
:Ir pescar;
|c|
:Fritar peixe;
|f|
:Comer o peixe;
stop
@enduml
```

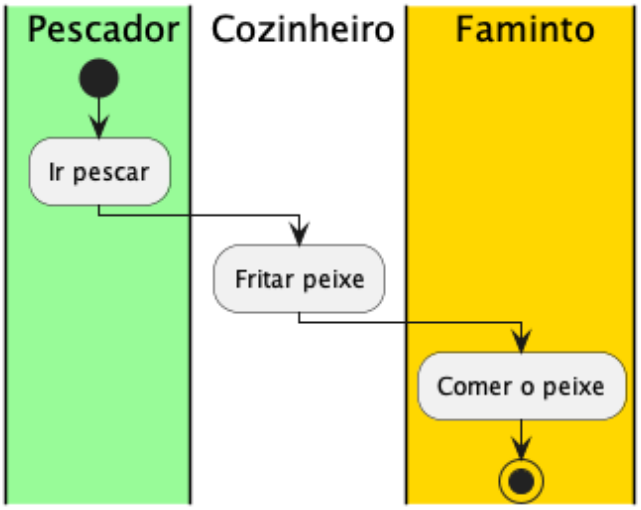
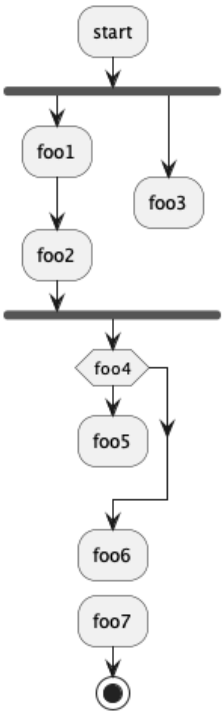


Diagrama de Atividade com plantUML

Detach ou Kill

detach

```
@startuml
:start;
fork
:foo1;
:foo2;
fork again
:foo3;
detach
endfork
if (foo4) then
:foo5;
detach
endif
:foo6;
detach
:foo7;
stop
@enduml
```



kill

```
@startuml
:start;
fork
:foo1;
:foo2;
fork again
:foo3;
kill
endfork
if (foo4) then
:foo5;
kill
endif
:foo6;
kill
:foo7;
stop
@enduml
```

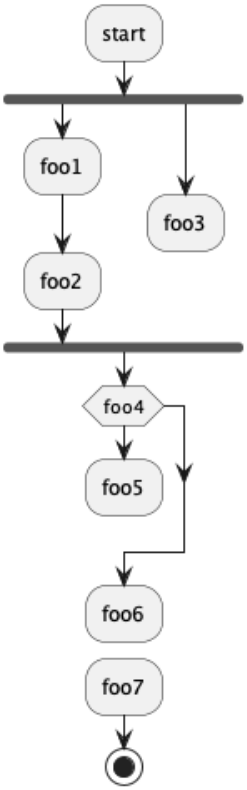


Diagrama de Atividade com plantUML

SDL (*Specification and Description Language*)

Tabela do Nome da Forma SDL

Nome	Sintaxe	Esteriótipo
Input (entrada)	<	<<input>>
Output (saída)	>	<<output>>
Procedure (procedimento)	|	<<procedure>>
Load (carga)	\	<<load>>
Save (salvar)	/	<<save>>
Continuous	}	<<continuous>>
Task	]	<<task>>

```
@startuml
:Ready;
:next(o)|
:Receiving;
split
:nak(i)<
:ack(o)>
split again
:ack(i)<
:next(o)
on several lines|
:i := i + 1]
:ack(o)>
split again
:err(i)<
:nak(o)>
split again
:foo/
split again
:bar\\
split again
:i > 5}
stop
end split
:finish;
@enduml
```

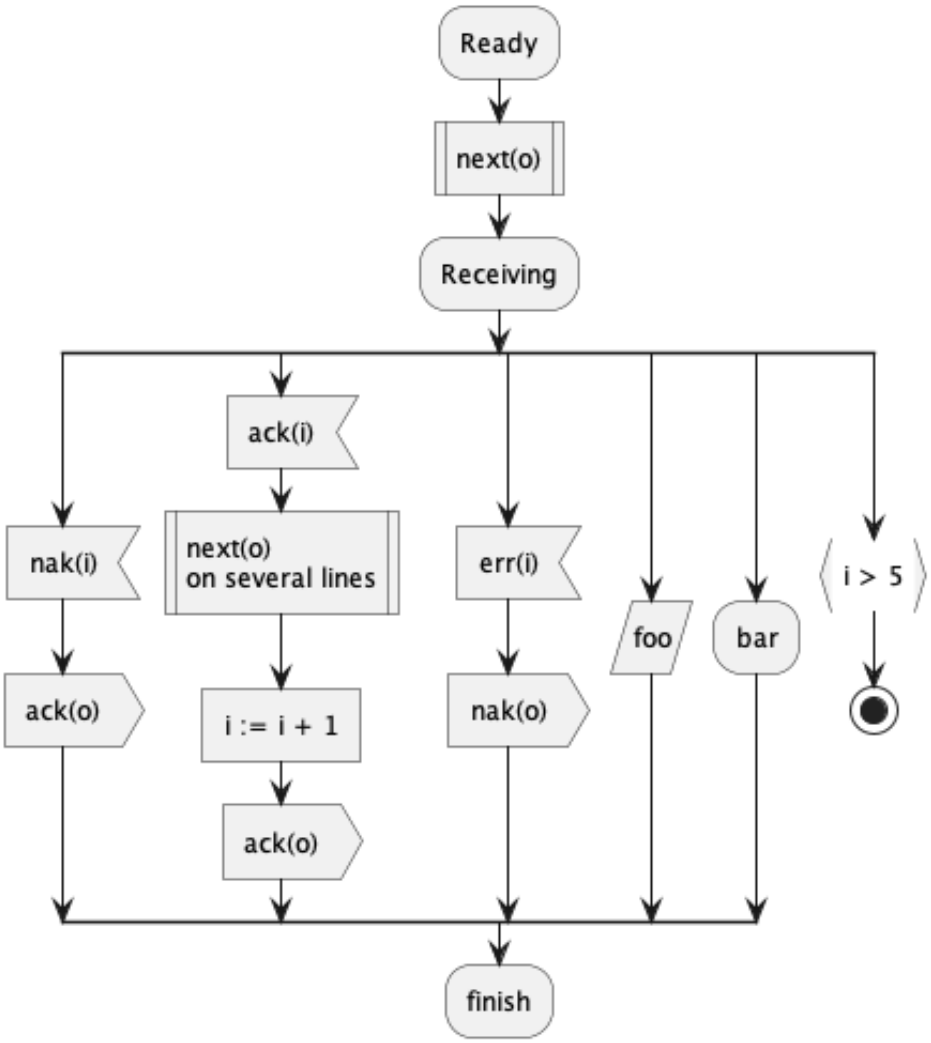
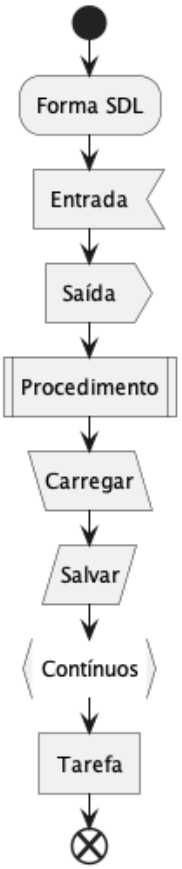


Diagrama de Atividade com plantUML

```
@startuml
start
:Forma SDL;
:Entrada; <<input>>
:Saída; <<output>>
:Procedimento; <<procedure>>
:Carregar; <<load>>
:Salvar; <<save>>
:Contínuos; <<continuous>>
:Tarefa; <<task>>
end
@enduml
```




```
@startuml
:Pronto;
:next(o); <<procedure>>
:Recebendo;
split
:nak(i); <<input>>
:ack(o); <<output>>
split again
:ack(i); <<input>>
:next(o)
em várias linhas;
<<procedure>>
:i := i + 1; <<task>>
:ack(o); <<output>>
split again
:err(i); <<input>>
:nak(o); <<output>>
split again
:foo; <<save>>
split again
:barra; <<load>>
split again
:i > 5; <<continuous>>
stop
end split
:fim;
@enduml
```

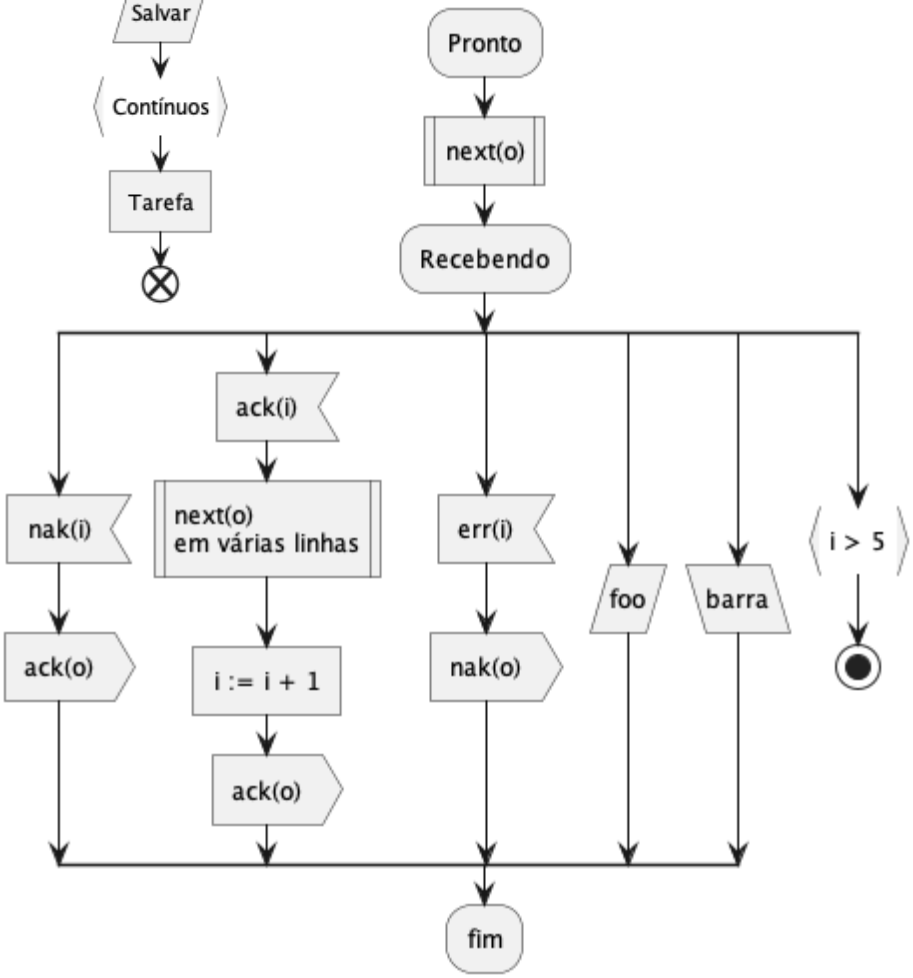


Diagrama de Atividade com plantUML

Exemplo Completo

```
@startuml
start
:ClickServlet.handleRequest();
:new page;
if (Page.onSecurityCheck) then (true)
:Page.onInit();
if (isForward?) then (no)
:Process controls;
if (continue processing?) then (no)
stop
endif
if (isPost?) then (yes)
:Page.onPost();
else (no)
:Page.onGet();
endif
:Page.onRender();
endif
else (false)
endif
if (do redirect?) then (yes)
:redirect process;
else
if (do forward?) then (yes)
:Forward request;
else (no)
:Render page template;
endif
endif
stop
@enduml
```

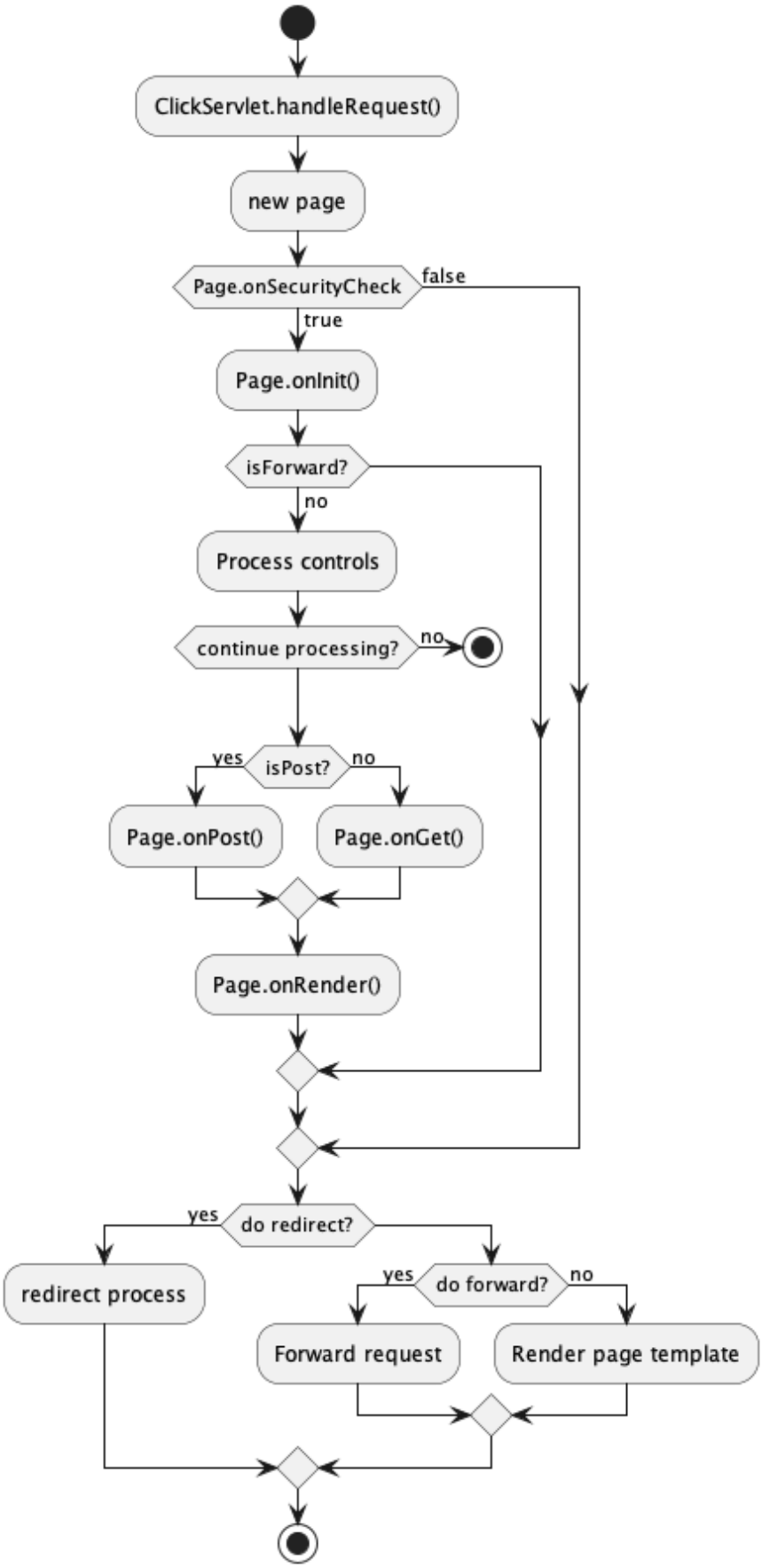
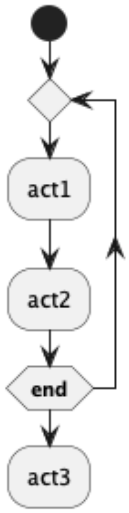


Diagrama de Atividade com plantUML

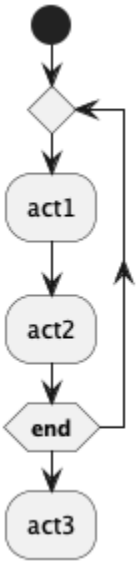
Estilo condicional

Estilo interno (por padrão)

```
@startuml
skinparam conditionStyle inside
start
repeat
:act1;
:act2;
repeatwhile (<b>end)
:act3;
@enduml
```

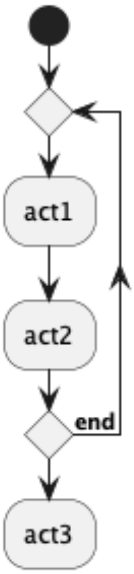



```
@startuml
start
repeat
:act1;
:act2;
repeatwhile (<b>end)
:act3;
@enduml
```

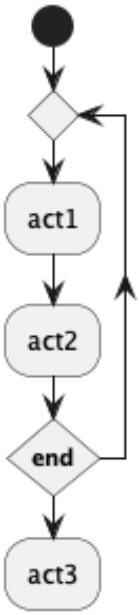


Estilo Diamante

```
@startuml
skinparam conditionStyle diamond
start
repeat
:act1;
:act2;
repeatwhile (<b>end)
:act3;
@enduml
```




```
@startuml
skinparam conditionStyle InsideDiamond
start
repeat
:act1;
:act2;
repeatwhile (<b>end)
:act3;
@enduml
```




```
@startuml
skinparam conditionStyle foo1
start
repeat
:act1;
:act2;
repeatwhile (<b>end)
:act3;
@enduml
```

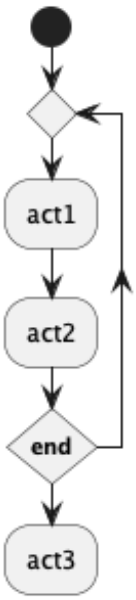
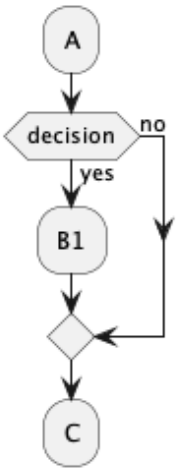


Diagrama de Atividade com plantUML

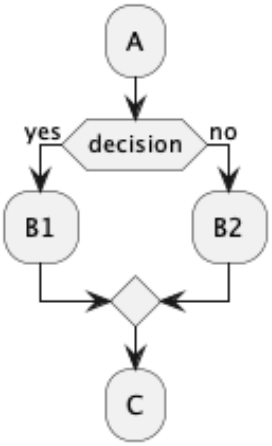
Estilo condicional End

Estilo diamante (default)

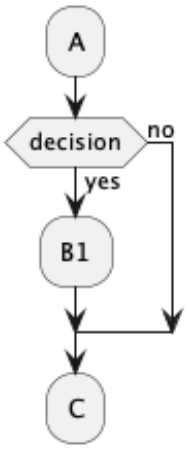
```
@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
:B1;
else (no)
endif
:C;
@enduml
```




```
@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
:B1;
else (no)
:B2;
endif
:C;
@enduml
```




```
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
:B1;
else (no)
endif
:C;
@enduml
```




```
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
:B1;
else (no)
:B2;
endif
:C;
@enduml
@enduml
```

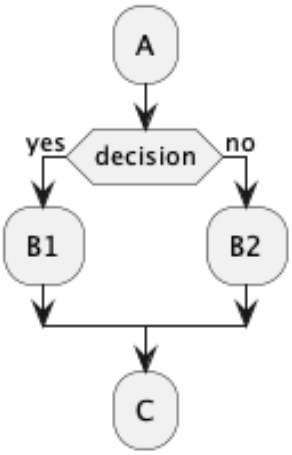
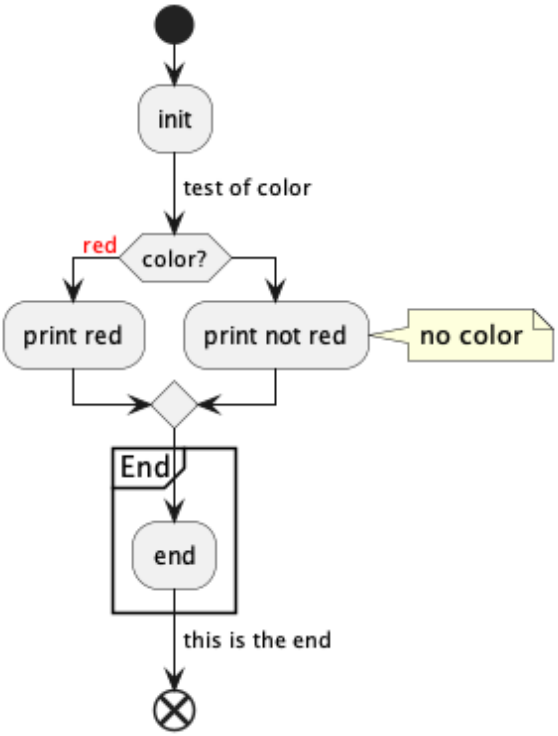


Diagrama de Atividade com plantUML

Usando estilo (global)

Sem estilo

```
@startuml
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
```



Com estilo

```
@startuml
<style>
activityDiagram {
BackgroundColor #33668E
BorderColor #33668E
FontColor #888
FontName arial
diamond {
BackgroundColor #ccf
LineColor #00FF00
FontColor green
FontName arial
FontSize 15
}
arrow {
FontColor gold
FontName arial
FontSize 15
}
partition {
LineColor red
FontColor green
RoundCorner 10
BackgroundColor PeachPuff
}
note {
FontColor Blue
LineColor Navy
BackgroundColor #ccf
}
}
document {
BackgroundColor transparent
}
</style>
start
:init;
```

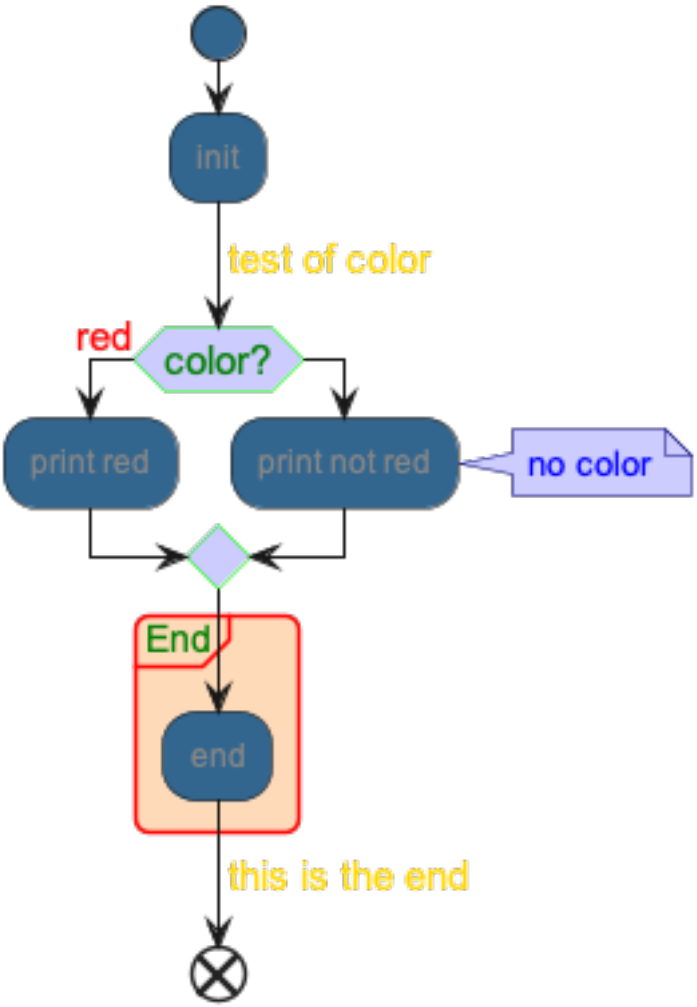


Diagrama de Atividade com plantUML

```
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
```