Metodología de Ingeniería de Software para web IFCD036PO

CONTENIDO PROGRAMÁTICO

1 Diagrama de secuencia

Crear diagramas de secuencia con PlantUML es notablemente sencillo. Esta facilidad de uso se atribuye en gran medida a la naturaleza amigable de su sintaxis, diseñada para ser intuitiva y fácil de recordar.

- **Sintaxis intuitiva:**

En primer lugar, los usuarios aprecian la sintaxis sencilla e intuitiva de PlantUML. Este diseño bien pensado significa que incluso aquellos que son nuevos en la creación de diagramas encuentran fácil comprender los conceptos básicos de forma rápida y sin complicaciones.

- **Correlación texto-gráfico:**

Otra característica distintiva es la estrecha semejanza entre la representación textual y el resultado gráfico. Esta correlación armoniosa garantiza que los borradores textuales se traduzcan con bastante precisión en diagramas gráficos, proporcionando una experiencia de diseño cohesiva y predecible sin sorpresas desagradables en el resultado final.

- **Proceso de elaboración eficiente:**

La fuerte correlación entre el texto y el resultado gráfico no sólo simplifica el proceso de creación, sino que también lo acelera significativamente. Los usuarios se benefician de un proceso más ágil con menos necesidades de revisiones y ajustes que requieren mucho tiempo.

- **Visualización durante la redacción:**

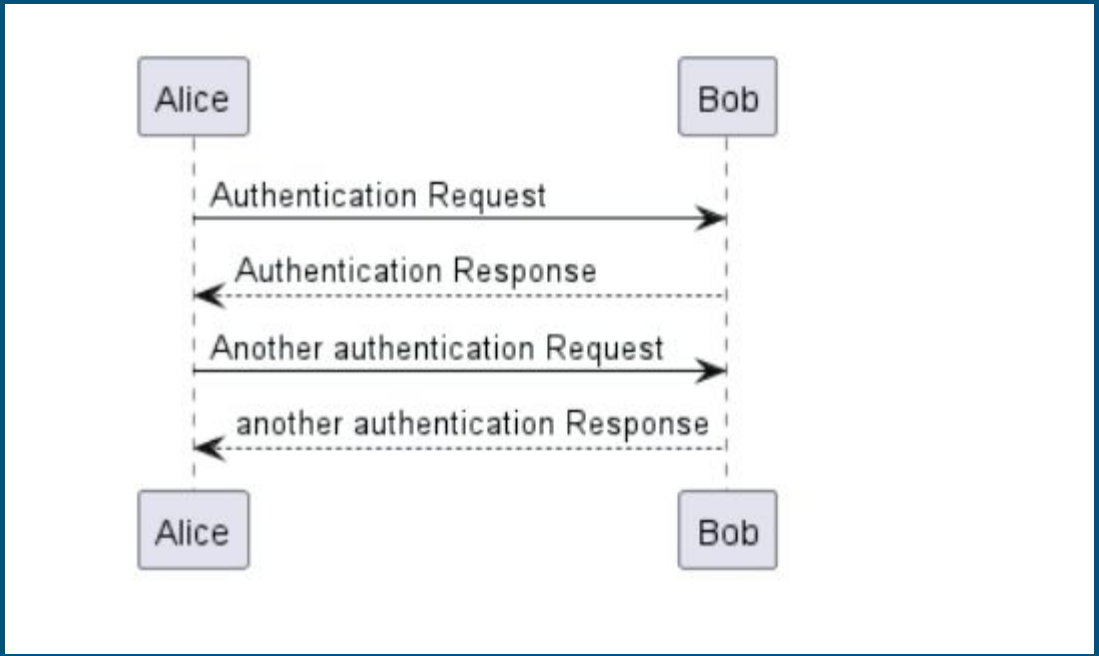
La posibilidad de visualizar el resultado gráfico final mientras se redacta el texto es una función que muchos consideran inestimable. Fomenta de forma natural una transición fluida del borrador inicial a la presentación final, mejorando la productividad y reduciendo la probabilidad de errores.

La secuencia -> es usada para dibujar un mensaje entre dos participantes. Los participantes tienen que ser declarados explícitamente.

Para definir una flecha punteada, se debe usar -->

También se puede usar <- y <--. No provoca cambios en el dibujo, pero puede mejorar la legibilidad. Tenga en cuenta que esto sólo es posible en diagramas de secuencia, las reglas son diferentes para otros diagramas.

```
@startuml
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response
@enduml
```


Alice
Bob
Authentication Request
Authentication Response
Another authentication Request
another authentication Response
Alice
Bob

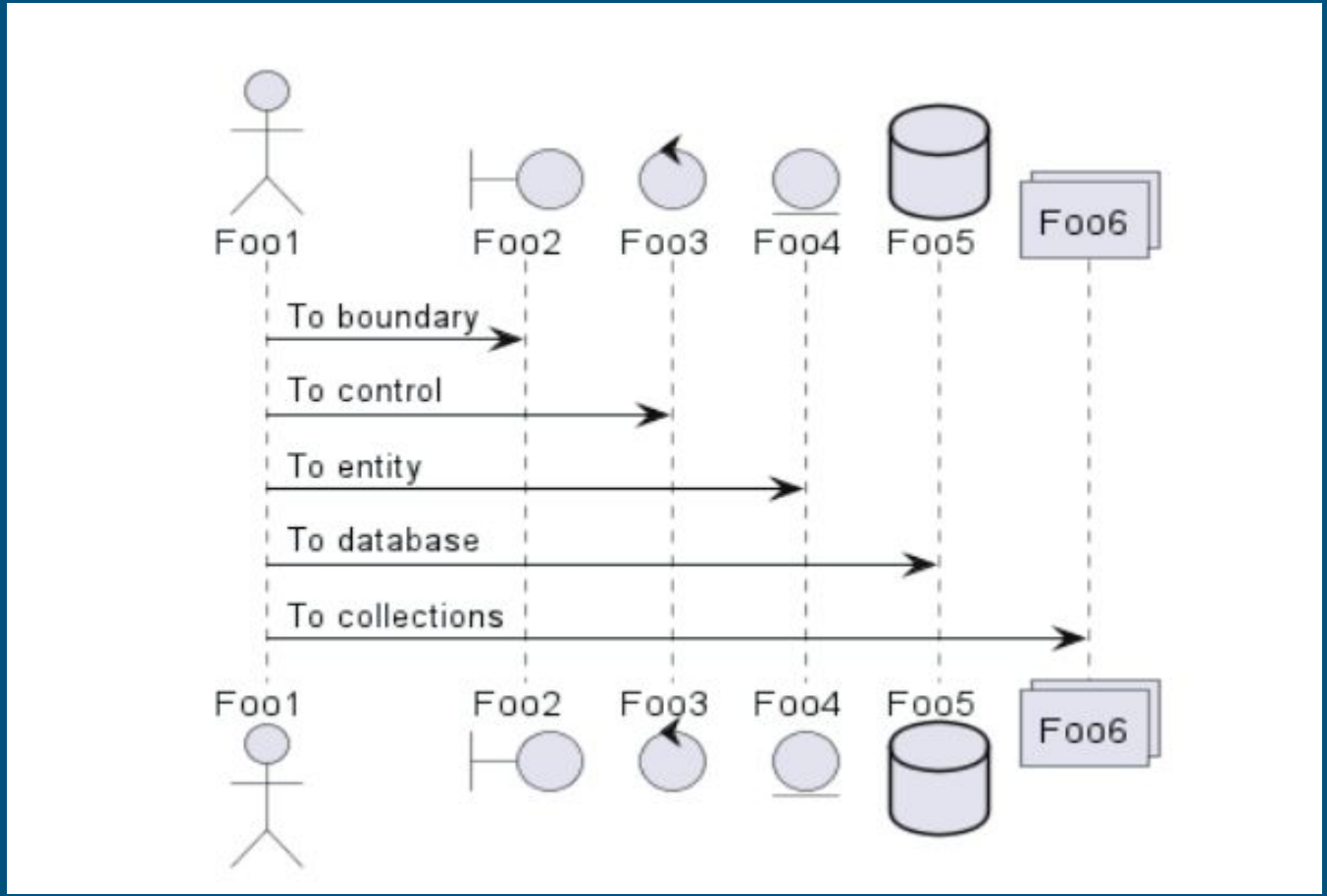
1.2 Declarando participantes

Es posible cambiar el orden de los participantes usando la palabra reservada `participant`.

También es posible el uso de otras palabras reservadas para declarar un participante:

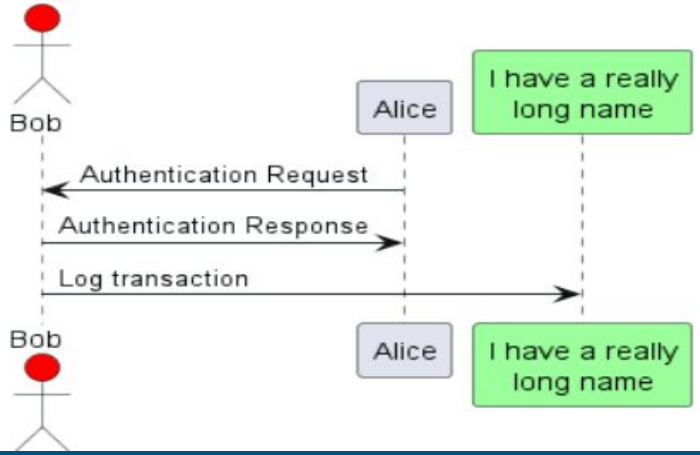
- `actor`
- `boundary`
- `control`
- `entity`
- `database`
- `collections`

```
@startuml
actor Foo1
boundary Foo2
control Foo3
entity Foo4
database Foo5
collections Foo6
Foo1 -> Foo2 : To boundary
Foo1 -> Foo3 : To control
Foo1 -> Foo4 : To entity
Foo1 -> Foo5 : To database
Foo1 -> Foo6 : To collections

@enduml
```


Foo1
Foo2
Foo3
Foo4
Foo5
Foo6
To boundary
To control
To entity
To database
To collections
Foo1
Foo2
Foo3
Foo4
Foo5
Foo6

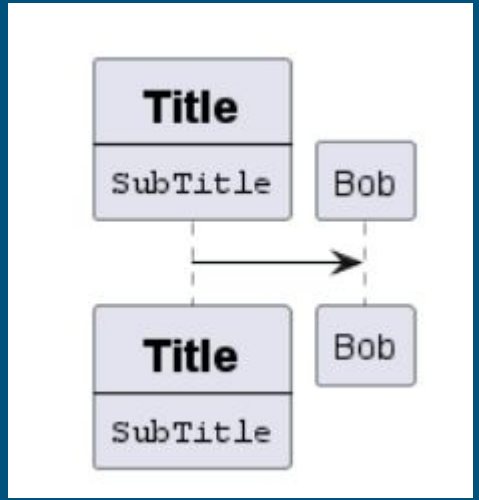
También es posible cambiar el color de fondo de los actores o participantes.

```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
   participant L as "I have a really\nlong name"  #99FF99
  '/

Alice->Bob: Authentication Request
Bob->Alice: Authentication Response
Bob->L: Log transaction
@enduml
```



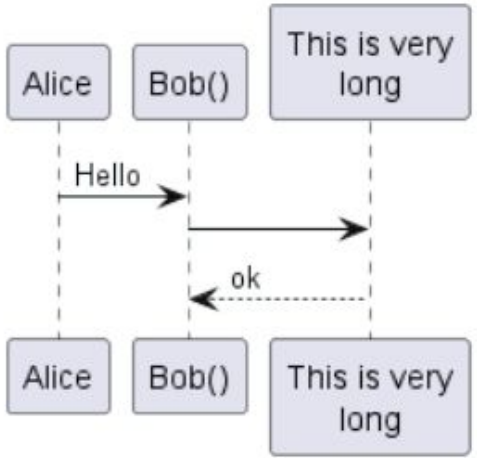
Puedes declarar el participante en múltiples líneas.

```
@startuml
participant Participant [
    =Title
    ----
    ""SubTitle""
]

participant Bob

Participant -> Bob
@enduml
```



Puedes usar comillas para definir participantes. Y puedes usar la palabra reservada **as** para asignar un alias a esos participantes.

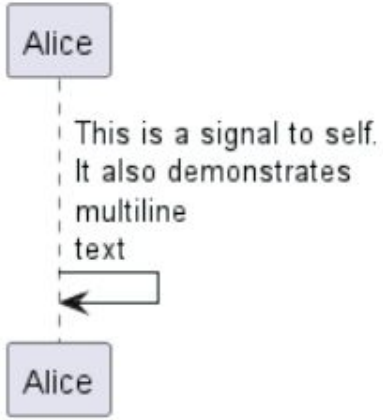
```
@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long
' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok
@enduml
```



Un participante puede enviarse un mensaje a sí mismo.

También es posible tener varias líneas utilizando .

```
@startuml
Alice -> Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```

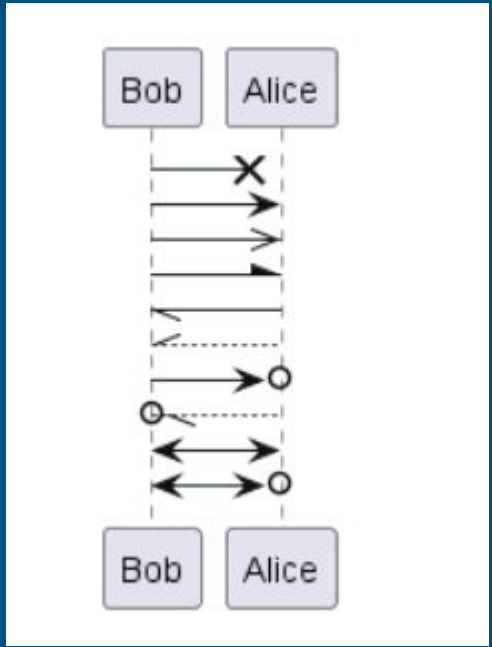



```
@startuml
Alice <- Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```

Puede cambiar el estilo de la flecha de diferentes formas:

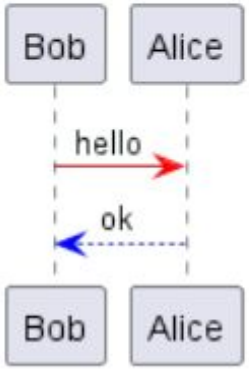
- añade una `x` al final para indicar un mensaje perdido
- utilice `\` o `/` en lugar de `<` o `>` para tener solo la parte inferior o superior de la flecha
- repite la cabeza de la flecha (por ejemplo, `>>` o `//`) para tener un trazo más fino

```
@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\- Alice
Bob //-- Alice

Bob ->o Alice
Bob o\\-- Alice

Bob <-> Alice
Bob <->o Alice
@enduml
```



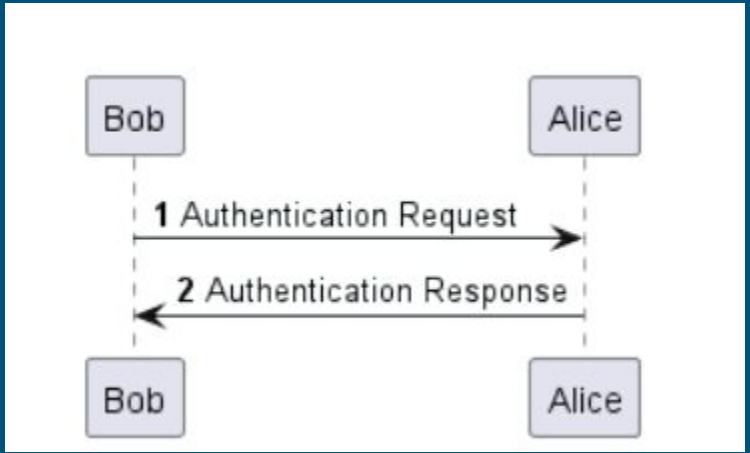
Puede cambiar el color de flechas individuales usando la siguiente notación:

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



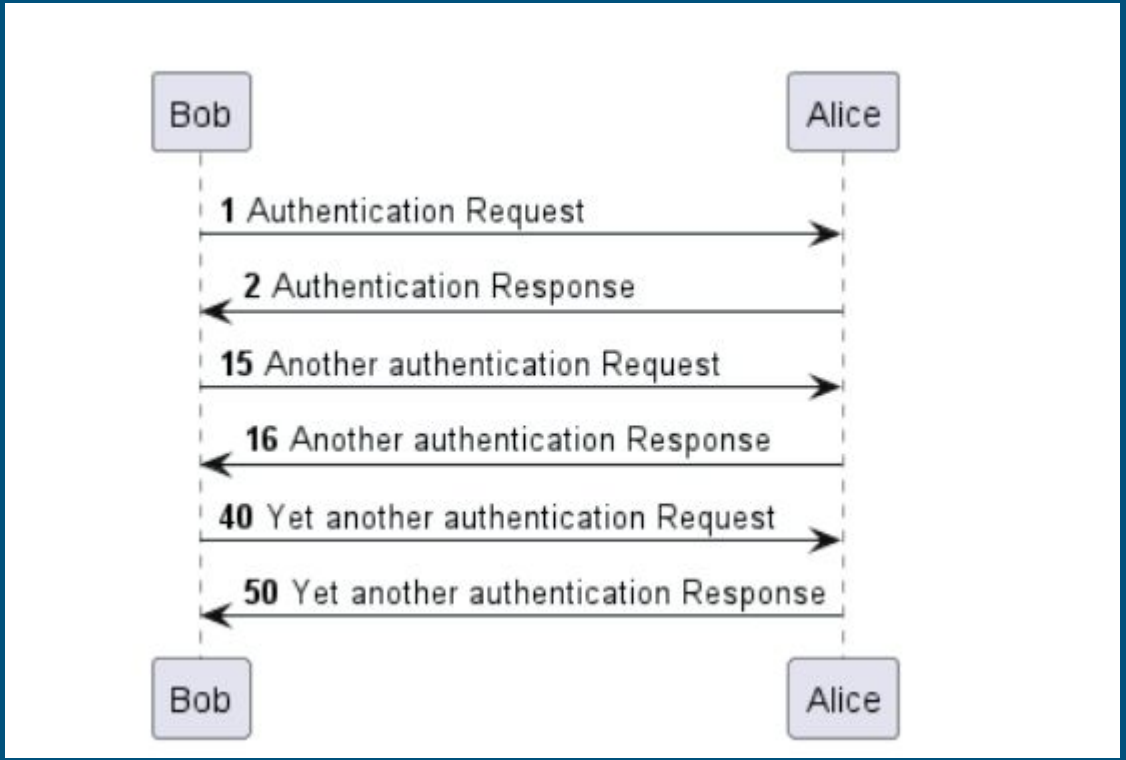
La palabra clave `autonumber` es usada para añadir automáticamente números a los mensajes.

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
@enduml
```



Puedes especificar un número de comienzo con `autonumber //número inicial//` , y también un incremento con `autonumber //número inicial// //incremento//`.

```
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml
```

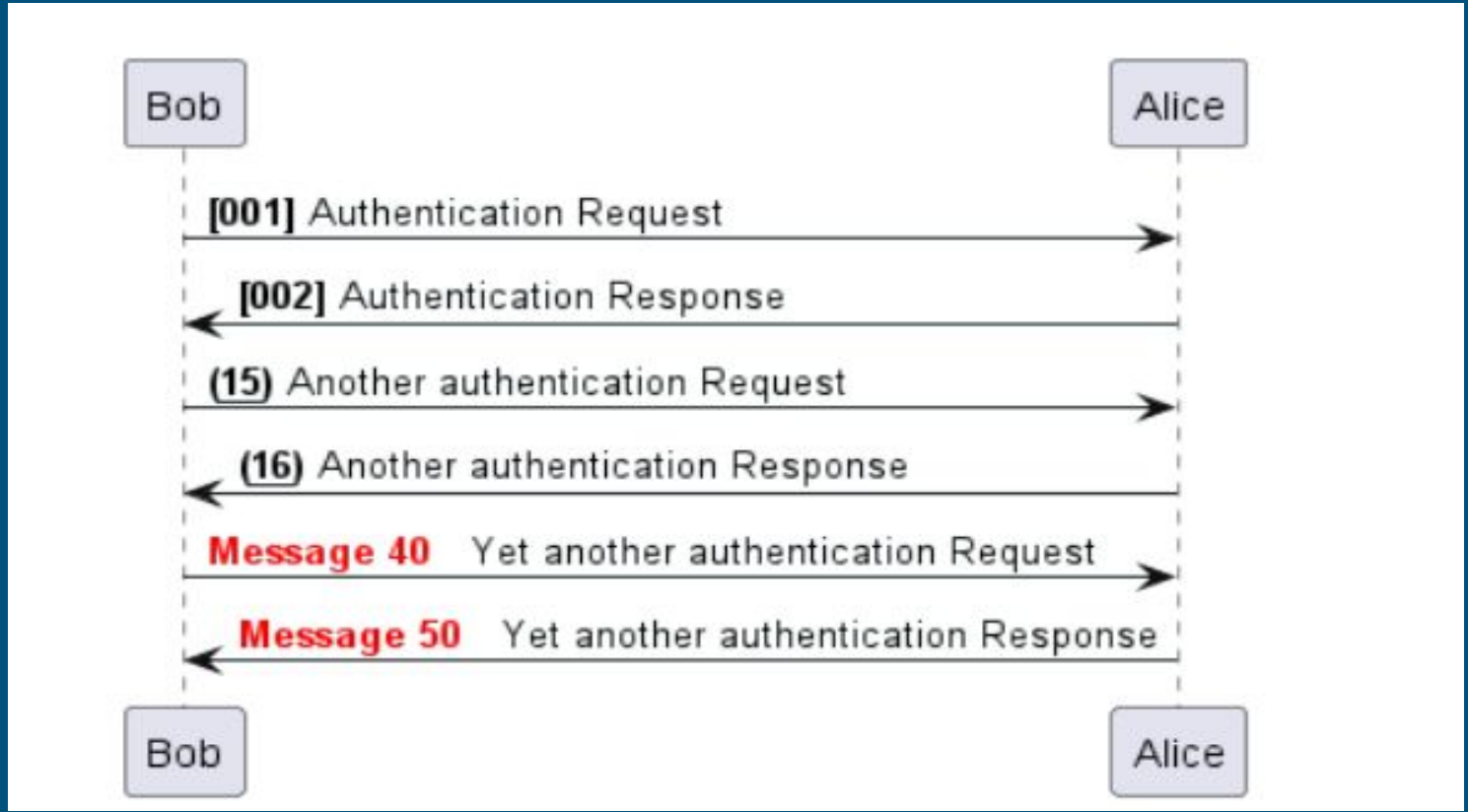

Bob
Alice
1 Authentication Request
2 Authentication Response
15 Another authentication Request
16 Another authentication Response
40 Yet another authentication Request
50 Yet another authentication Response
Bob
Alice

Puedes especificar un formato para su número usándolo entre comillas dobles.

El formateo se hace mediante la calse Java `DecimalFormat` (`0` denota un dígito, `#` denota un digito y cero si está ausente).

Puedes usar alguna etiqueta HTML en el formato.

```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
```

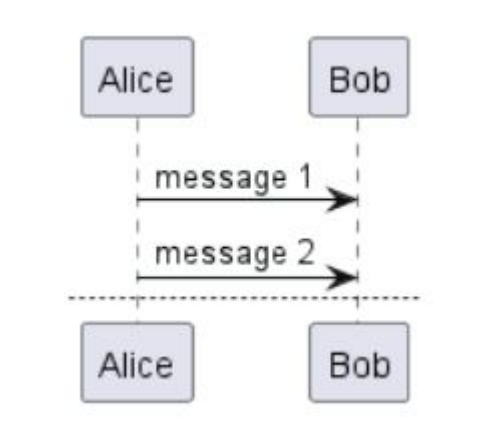

Bob
Alice
[001] Authentication Request
[002] Authentication Response
(15) Another authentication Request
(16) Another authentication Response
Message 40 Yet another authentication Request
Message 50 Yet another authentication Response
Bob
Alice

La palabra reservada **newpage** es empleada para dividir un diagrama en varias imágenes.

Puedes colocar un título para la página nueva justo después de la palabra reservada **newpage** .

Esto es bastante práctico con *Word* para devolver diagramas grandes en varias páginas.

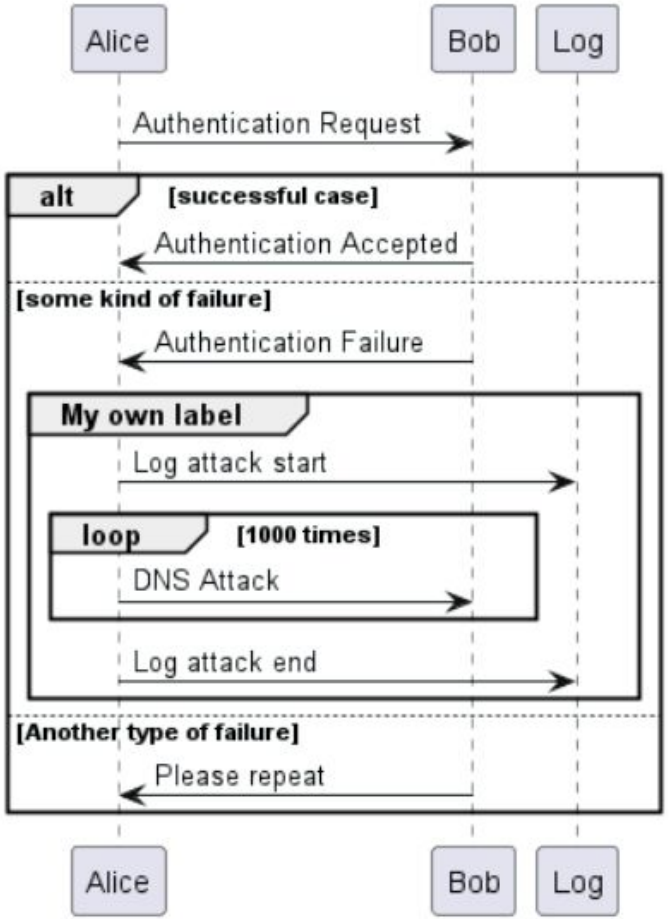
```
@startuml

Alice -> Bob : message 1
Alice -> Bob : message 2

newpage

Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page

Alice -> Bob : message 5
Alice -> Bob : message 6
@enduml
```



Es posible agrupar mensajes usando las siguientes palabras reservadas:

- `alt/else`
- `opt`
- `loop`
- `par`
- `break`
- `critical`
- `group`, seguida de un texto para mostrar

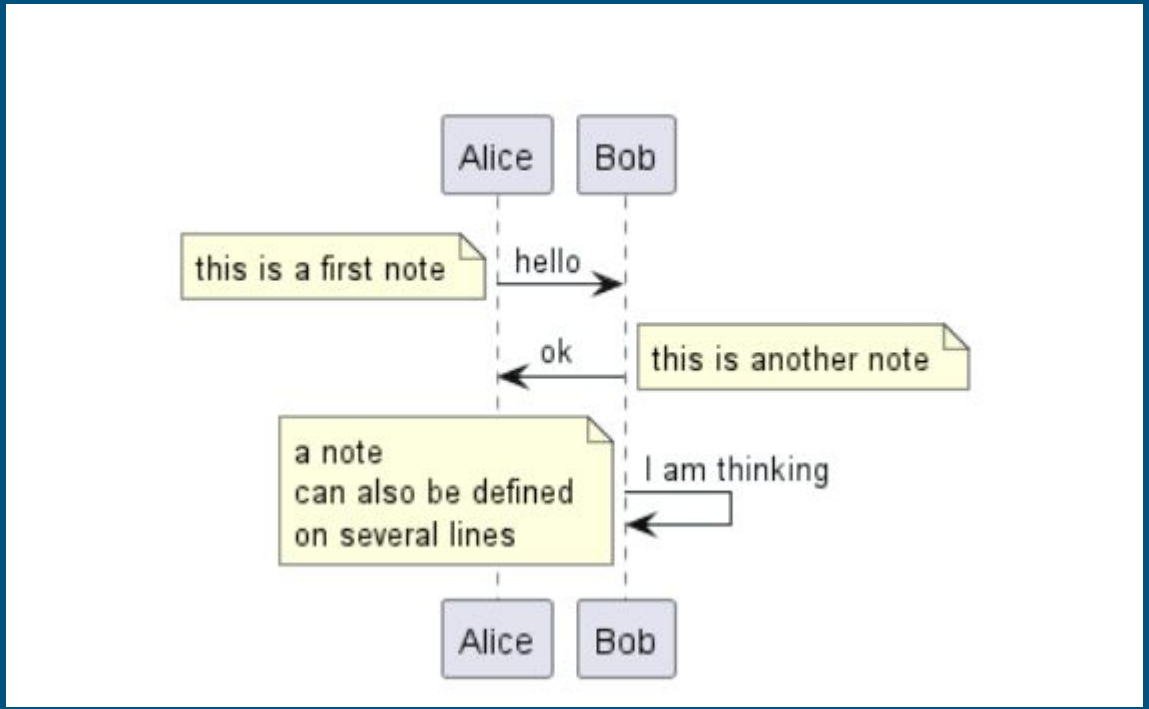
```
@startuml
Alice -> Bob: Authentication Request

alt successful case

    Bob -> Alice: Authentication Accepted

else some kind of failure

    Bob -> Alice: Authentication Failure
    group My own label
    Alice -> Log : Log attack start
        loop 1000 times
            Alice -> Bob: DNS Attack
        end
    Alice -> Log : Log attack end
    end

else Another type of failure

   Bob -> Alice: Please repeat

end
@enduml
```



Es posible colocar notas en mensajes usando las palabras reservadas `note left` o `note right` *inmediatamente después del mensaje.*

Puedes tener una nota multi-líneas usando la palabra reservada `end note` .

```
@startuml
Alice->Bob : hello
note left: this is a first note

Bob->Alice : ok
note right: this is another note

Bob->Bob : I am thinking
note left
a note
can also be defined
on several lines
end note
@enduml
```

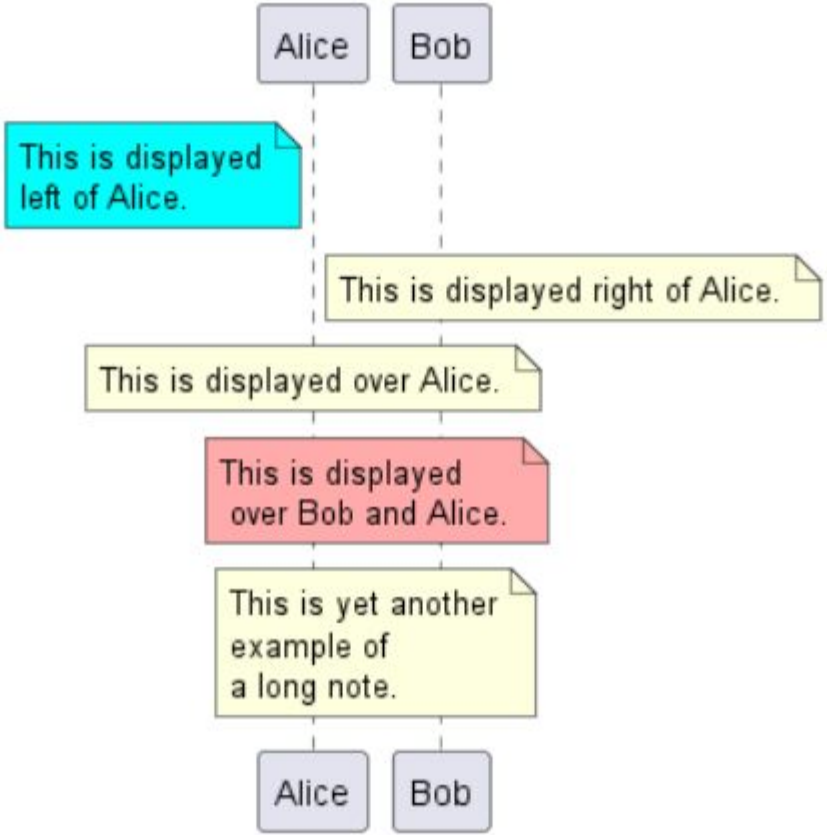

Alice
Bob
this is a first note
hello
ok
this is another note
a note
can also be defined
on several lines
I am thinking
Alice
Bob

También es posible colocar notas relativas al participante con las palabras reservadas <code>note left of</code> , `note right of` o `note over` .

Es posible resaltar una nota cambiando su color de fondo.

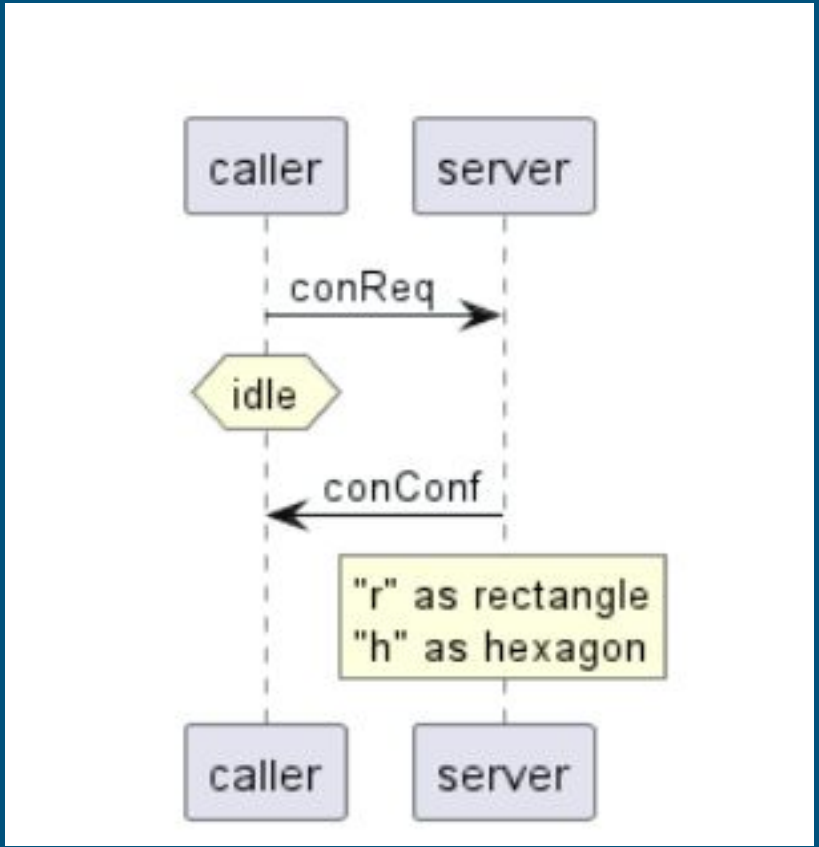
También puedes tener una nota multi-líneas usando la palabra reservada `end note` .

```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

note right of Alice: This is displayed right of Alice.

note over Alice: This is displayed over Alice.

note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

note over Bob, Alice
This is yet another
example of
a long note.

end note
@enduml
```


Alice
Bob
This is displayed left of Alice.
This is displayed right of Alice.
This is displayed over Alice.
This is displayed over Bob and Alice.
This is yet another example of a long note.
Alice
Bob

Puedes usar las palabras reservadas **hnote** y **rnote** para cambiar el aspecto de las notas.

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
 "r" as rectangle
 "h" as hexagon
endrnote
@enduml
```


caller
server
conReq
idle
conConf
"r" as rectangle
"h" as hexagon
caller
server