FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1 — Smalltalk

[75.07/95.02] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre de 2019

Alumno:	Federico N. Pratto
Número de padrón:	96.223
Email:	fede.pratto@gmail.com

Índice

1. Introducción

El presente informe reúne la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de chat en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

Supuse que el usuario que utilice el chat no instanciara, de forma directa, objetos de ninguna clase que no sea la de *AlgoChat*.
Ademas se supuso que, el usuario no cometerá errores al utilizar el programa cuando se trate de tipos de clases. Por ejemplo, si un mensaje espera recibir un objeto de la clase *ByteString*, el usuario no ingresara uno de la clase *SmallInteger*.
Luego, supuse que al crear una *Conversacion*, el constructor recibirá al menos un *Usuario*.

3. Diagramas de clase

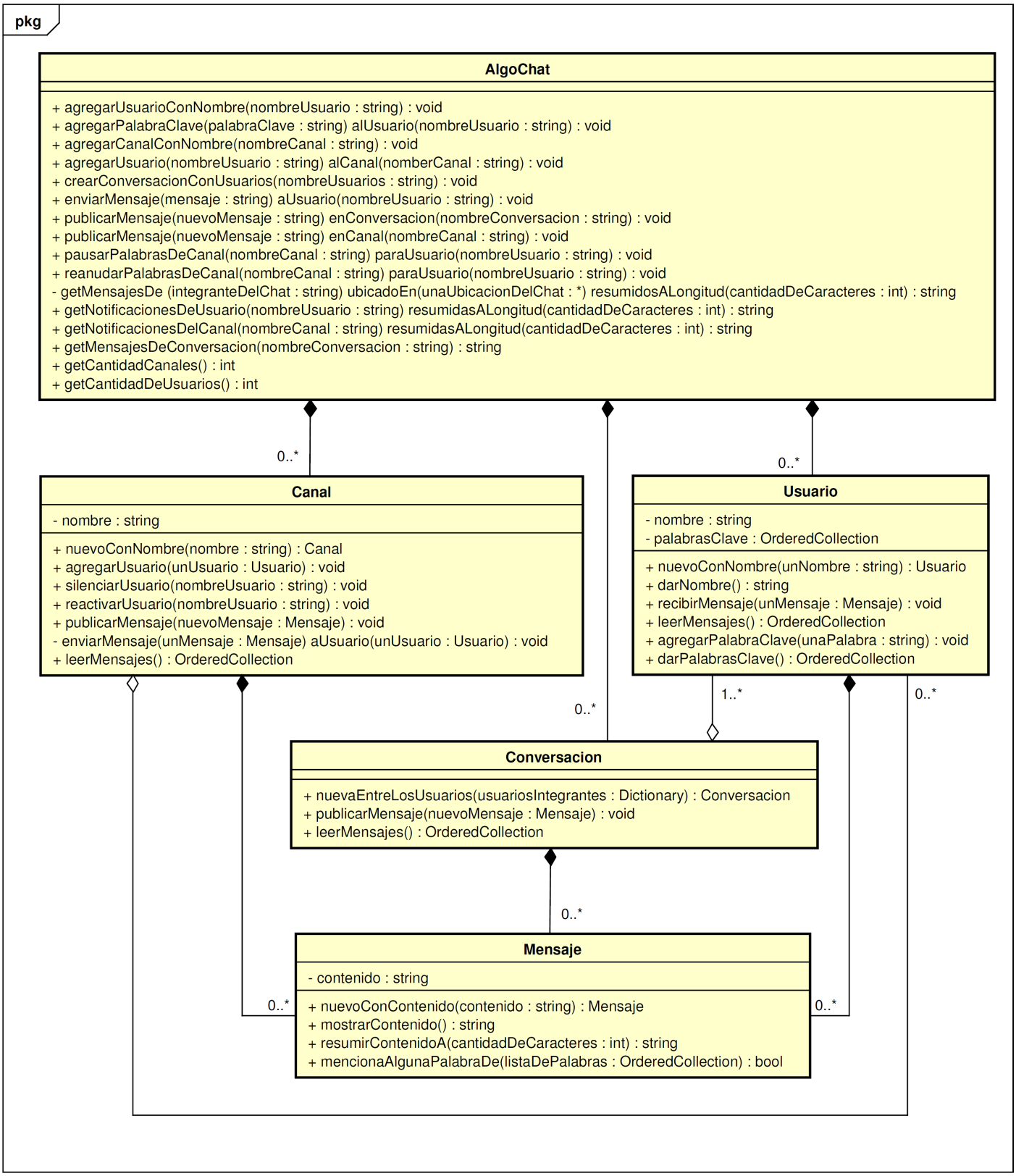

Figura 1: Diagrama de clases general.

La resolución del trabajo consiste en cinco clases las cuales colaboran entre si de la siguiente manera:

3.1. Usuario

Clase a partir de la cual crearemos nuevos objetos *Usuario* para integrar nuestro chat, y cuyas responsabilidades serán:

- Recibir un objeto *Mensaje*, y almacenarlo en un listado, donde estarán todos los *Mensajes* que el *Usuario* haya recibido desde su creación.
- Indicar cual es su nombre.
- Devolver un listado con todos sus *Mensajes*.
- Recibir una palabra clave, y almacenarla en un listado, para ser notificado si esta es mencionada en un *Canal* al que pertenezca.
- Devolver un listado con todas sus palabras clave.

Ademas de esto, el *Usuario* podrá ser agregado a través del *AlgoChat* tanto a *Canales* como *Conversaciones*, con otros de su mismo tipo.

3.2. Canal

Clase a partir de la cual se podrán crear *Canales* de anuncios en el chat, estos consistirán básicamente en un lugar del chat con un nombre en particular (Por ejemplo: 'Anuncios') donde se podrán publicar *Mensajes*. Y cuyas responsabilidades serán:

- Aceptar a un nuevo *Usuario* que podra ser agregado al *Canal*.
- Recibir un *Mensaje*, y almacenarlo en un listado, donde estarán todos los *Mensajes* que se publicaron en el *Canal* desde su creación.
- Al recibir un nuevo *Mensaje*, deberá enviar el mismo a sus *Usuarios*, si es que estos son mencionados (@nombreUsuario) o aparece alguna de las palabras clave, que hayan activado previamente, en el contenido del *Mensaje*.
- Silenciar un *Usuario*. Al hacer esto el *Usuario* en cuestión dejara de recibir notificaciones si alguna de las palabras clave que el activo son mencionadas en un el contenido de un *Mensaje*, pero continuara siendo informado si es mencionado su nombre.
- Reactivar a un *Usuario*, esto consiste básicamente en volver a notificar al *Usuario* si sus palabras claves son mencionadas en futuros *Mensajes*.
- Devolver un listado con todos sus *Mensajes*.

Una peculiaridad con respecto a como se almacenan los *Usuarios* dentro de un *Canal*, para hacer lo mas eficiente posible el saber quien ha sido silenciado y quien no, puede leerse en la **sección 4.1**.

3.3. Conversacion

Clase a partir de la cual se podrán crear *Conversaciones* entre *Usuarios* en el chat. Estas consistirán en un lugar donde se podrán publicar *Mensajes*, los cuales serán recibidos por todos los integrantes de la *Conversación*. Sus responsabilidades son las siguientes:

- Recibir un *Mensaje*, y almacenarlo en un listado, donde estarán todos los *Mensajes* que se publicaron en la *Conversación* desde su creación.
- Al recibir un nuevo *Mensaje*, deberá reenviar el mismo a todos sus *Usuarios*.
- Devolver un listado con todos sus *Mensajes*.

3.4. Mensaje

Clase a partir de la cual se podrán crear *Mensajes* para ser enviados a *Usuarios*, *Conversaciones* o *Canales*. Sus responsabilidades son las siguientes:

- Devolver su contenido, es decir, el texto que lo compone.
- Devolver su contenido resumido a una determinada cantidad de caracteres solicitada. Si la dicha cantidad es menor o igual a cero, el *Mensaje* devolverá su contenido sin resumir.
- Indicar si una palabra que recibe aparece mencionada en su contenido.

3.5. AlgoChat

Clase a partir de la cual sera instanciado el Chat, sus responsabilidades son:

- Crear nuevos *Usuarios*, *Conversaciones* y *Canales* en el Chat.
- Dotar de *Usuarios* a un *Canal*.
- Enviar mensajes a *Usuarios*, *Conversaciones* y *Canales* en el Chat.
- Solicitar a un *Canal* que pause o reactive las notificaciones por palabra clave de un *Usuario* determinado.
- Devolver los mensajes de un *Usuario*, *Conversacion* o *Canal* resumidos o no a una cierta cantidad de caracteres.

4. Detalles de implementación

4.1. Canal: Almacenamiento de Usuarios

Al momento de almacenar los *Usuarios* que se van agregando al *Canal* decidí hacerlo de la siguiente manera:

```
agregarUsuario: unUsuario

    | usuario |

    usuario := Array new: 2.
    usuario at: 1 put: unUsuario.
    usuario at: 2 put: 'activo'.

    usuarios at: (unUsuario darNombre) ifAbsentPut: usuario.
```

El *Canal* recibe al objeto *unUsuario*, y lo almacena en el primer indice de un nuevo *Array* de dos elementos. Luego, en el segundo indice de dicho *Array* se almacena el estado del *Usuario* dentro del *Canal* (activo/silenciado).

Dicho estado sera por definición activo cuando se agrega un nuevo *Usuario*, e ira mutando a lo largo de la ejecución del programa en función de si el *Usuario* es silenciado o reactivado, para ese *Canal* en particular.

Finalmente, se almacena este *Array* en el *Dictionary*, al que referencia el atributo *usuarios* del *Canal*, utilizando como clave el nombre del *Usuario*. Si ya habia una clave con dicho valor en *usuarios* (Es decir, el *Usuario* ya pertenece al *Canal*) no se almacena el *Array*.

4.2. Conversación

La *Conversacion* estará compuesta en su estructura por:

- Una *OrderedCollection* donde se almacenaran los objetos *Mensaje* en el orden que sean recibidos.
- Un *Dictionary* donde se almacenaran los *Usuarios* que integren la *Conversacion* segun:

  clave: *nombreDelUsuario* –> valor: *Usuario*.

4.3. Usuario

Un usuario estará compuesto en su estructura por:

- Un *String* representando el nombre.
- Una *OrderedCollection* donde se almacenaran los objetos *Mensaje* en el orden que sean recibidos.
- Una *OrderedCollection* donde se almacenaran las palabras clave que se le vayan agregando.

5. Diagramas de secuencia

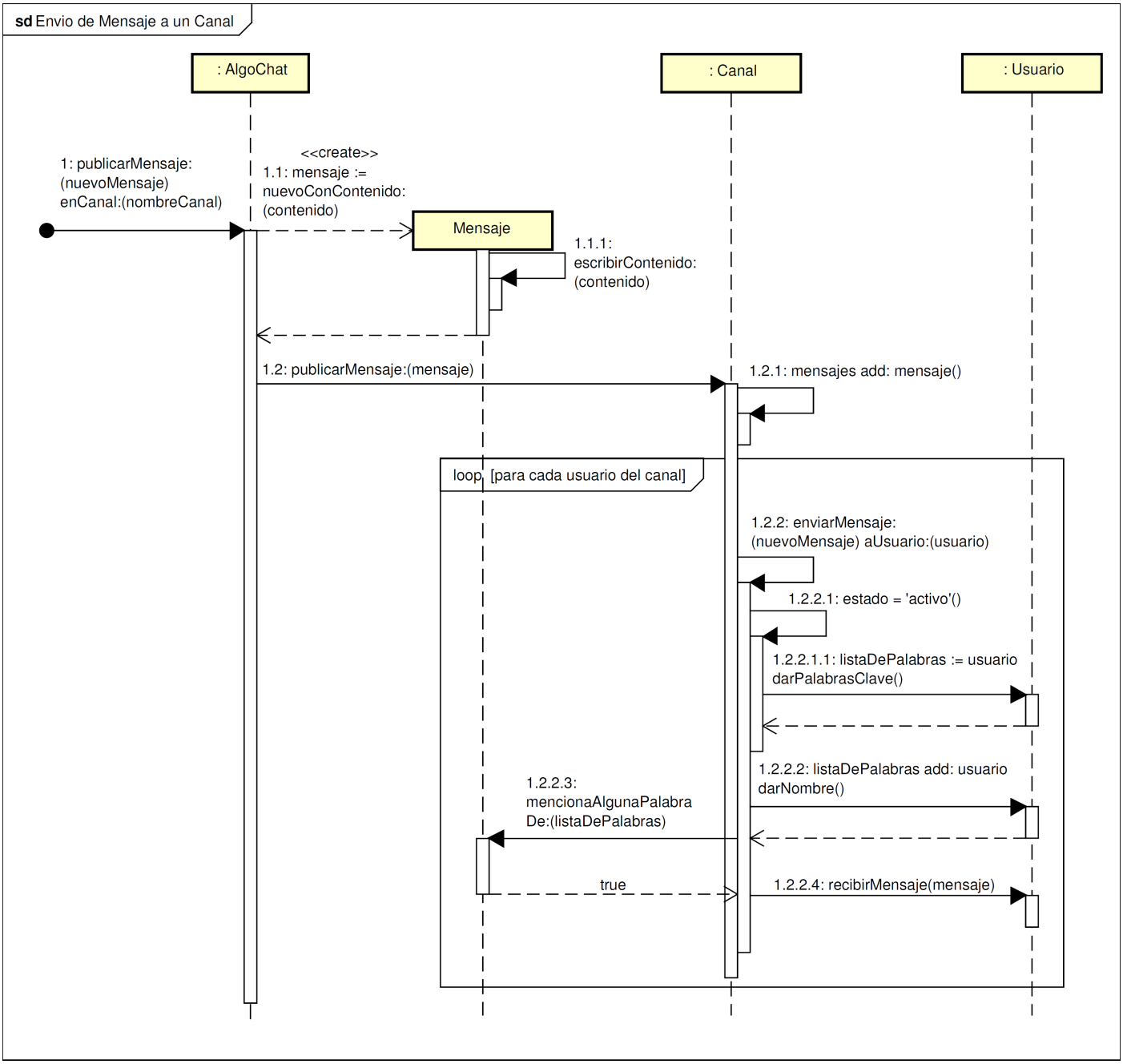

Figura 2: Envió de un *Mensaje* a un *Canal*.

En el diagrama de arriba puede observarse como es el proceso para enviar un *Mensaje* nuevo a un *Canal* del chat. El proceso consta de los siguientes pasos:

1. Se envía el mensaje *publicarMensaje: nuevoMensaje enCanal: nombreCanal* al *AlgoChat*
2. El *AlgoChat* crea un nuevo Mensaje con el contenido recibido.
3. El *AlgoChat* envía el mensaje *publicarMensaje:mensaje* al *Canal* con el *Mensaje* recién creado.
4. El *Canal* almacena el *Mensaje* recibido en su lista de mensajes, y luego realiza el siguiente *loop* para cada Usuario del *Canal*:
   a) Se crea una lista vacía de palabras a buscar en el *Mensaje*.
   b) El *Canal* se envía a si mismo el mensaje *enviarMensaje: nuevoMensaje aUsuario: usuario*. El cual verifica si el *Usuario* se encuentra activo (ver Sección 4.1).
      - Caso afirmativo: Solicita al usuario sus palabras clave, enviadole el mensaje *darPalabrasClave*, y las guarda en la lista arriba mencionada. Luego, solicita al usuario su nombre, enviadole el mensaje *darNombre* y lo agrega a la lista anexandole un '@' adelante.
      - Caso negativo: Unicamente solicita al usuario su nombre.

   c) Envía al *Mensaje* el mensaje *mencionaAlgunaPalabraDe: listaDePalabras* pasandole la lista de palabras a buscar.
   d) Si el *Mensaje* mencionaba alguna de las palabras, se envía el mensaje *recibirMensaje: mensaje* al *Usuario* pasandole el *Mensaje* para que lo guarde.

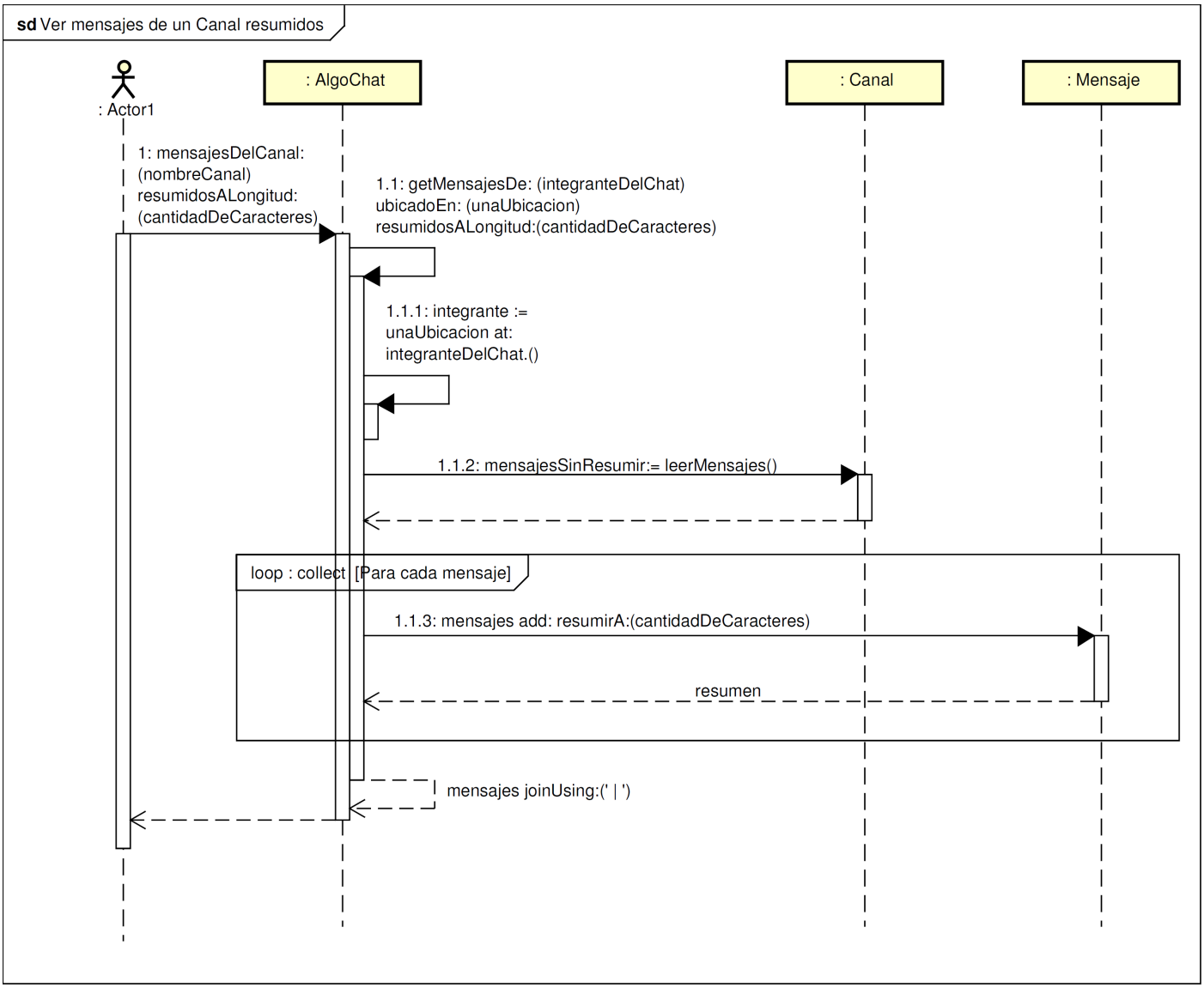

Figura 3: Ver los *Mensajes* publicados en una *Conversacion*.

En el diagrama de arriba puede apreciarse como es el proceso para solicitar a un *Canal* los *Mensajes* publicados en el mismo, resumidos a una cierta cantidad de caracteres cada uno. El proceso es el siguiente:

1. Un *Actor* envía al chat el mensaje *mensajesDelCanal: nombreCanal resumidosALongitud: cantidadDeCaracteres*.
2. El AlgoChat se envia a si mismo el mensaje *getMensajesDe: integranteDelChat ubicadoEn: unaUbicacion resumidosALongitud: cantidadDeCaracteres* el cual puede resumir los *Mensajes* de cualquier integrante del chat (*Usuario, Conversacion o Canal*) y realiza las siguientes acciones:
   a) Guarda en una variable *integrante* al *Canal* del cual se quieren obtener los *Mensajes*.
   b) Envia al *Canal* el mensaje *leerMensajes* que le retorna una lista con todos los *Mensajes* del *Canal*.
   c) Abre un *loop* dentro del cual le envia a cada *Mensaje* del listado recibido el mensaje *resumirA: cantidadDeCaracteres*. Este devuelve el contenido de cada mensaje resumido.
   d) Se almacenan los resúmenes en una nueva lista a medida que avanza el proceso.
3. Se devuelve un *String* formado por todos los elementos de la lista de resumidos unidos por pipes. Ej: 'Hola | mundo'

6. Excepciones

No se hizo uso de excepciones en la resolución de este trabajo practico.