Practica Final.

Apartado B. Diseño Operativo y Funcional.

Carlos Contreras Sanz NIA: 100300562

Álvaro Gómez Ramos NIA: 100307009

UNIVERSIDAD CARLOS III DE MADRID

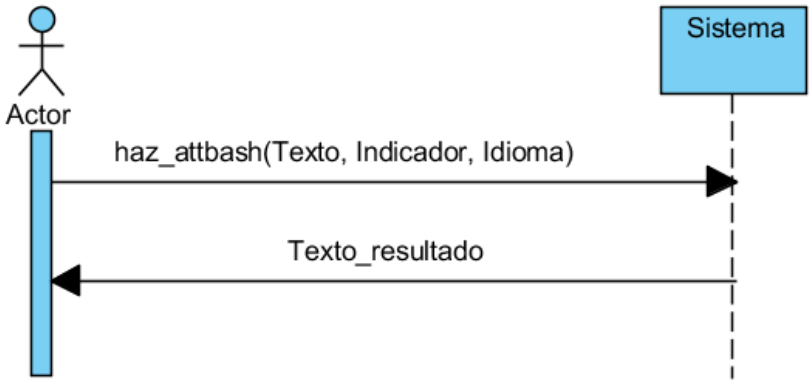
Diseño Operativo.

Diagramas de interacción del Sistema.

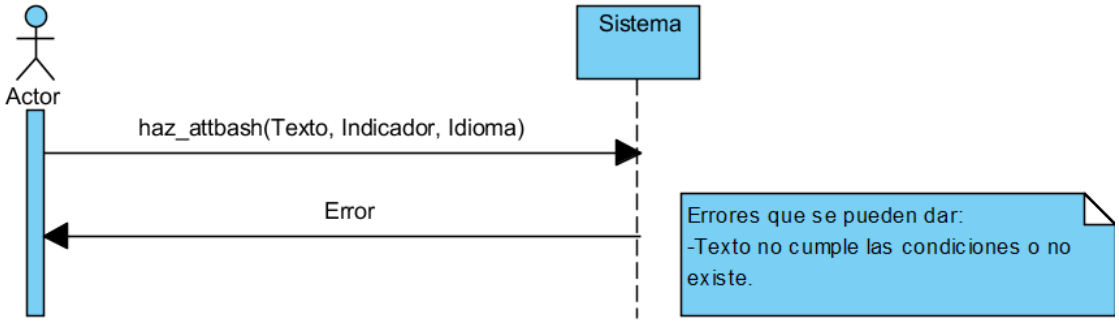
Notación utilizada: En el recuadro marcaremos siempre los errores que se puedan dar con los datos introducidos. Estos fallos se pueden dar individualmente o de forma simultánea, por lo que en cuanto exista un dato erróneo nos devolverá una excepción o mensaje por pantalla.

RF 1: Cifrado/descifrado con attbash.

En el caso de que no surja ningún error, tenemos el siguiente diagrama:

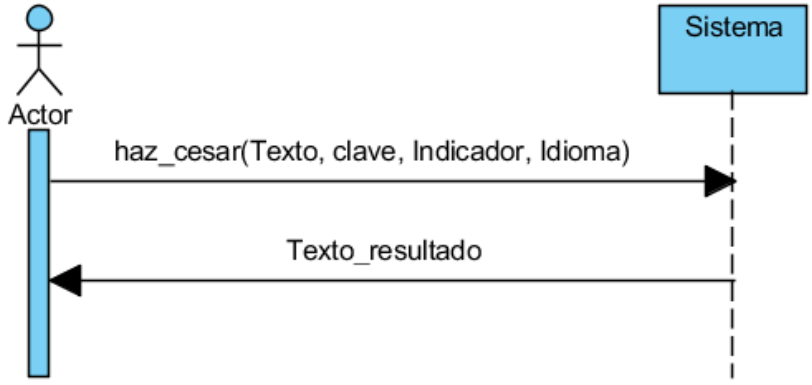


En caso contrario tenemos este diagrama:

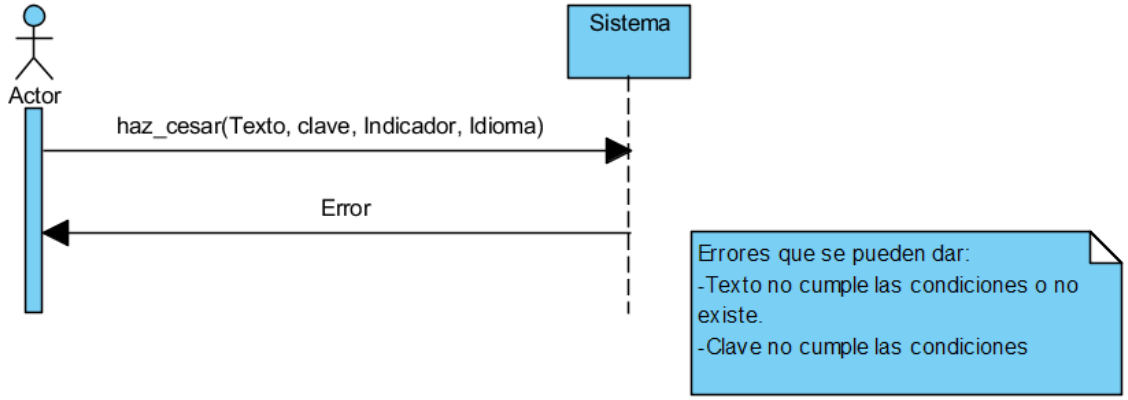


RF 2: Cifrado/descifrado con cesar.

En el caso de que no surja ningún error, tenemos el siguiente diagrama:

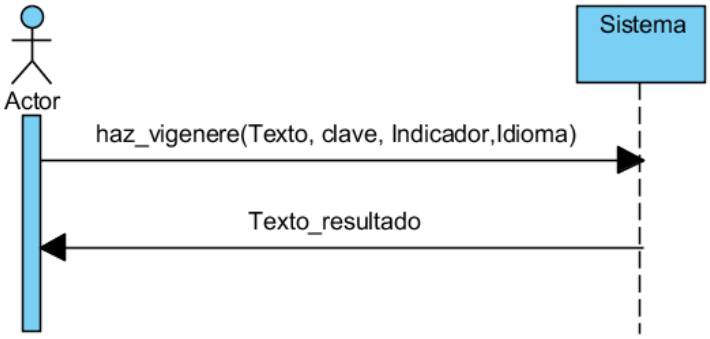


En caso contrario tenemos este diagrama:

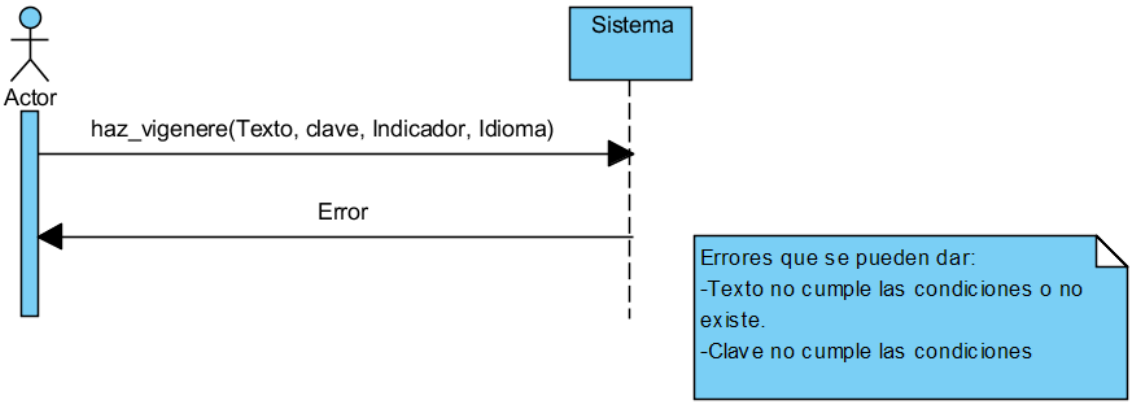


RF 3: Cifrado/descifrado con vigenere.

En el caso de que no surja ningún error, tenemos el siguiente diagrama:

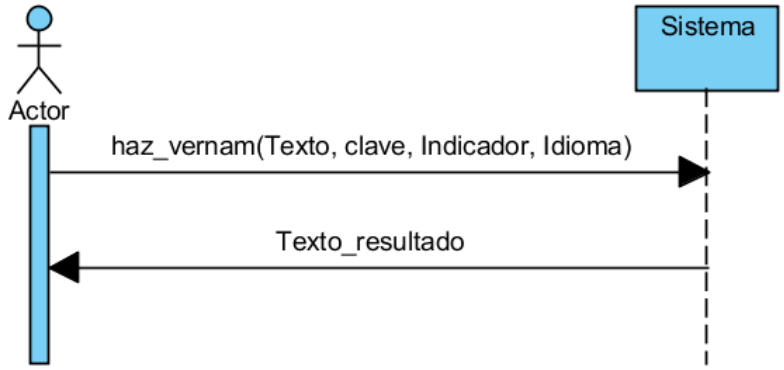


En caso contrario tenemos este diagrama:

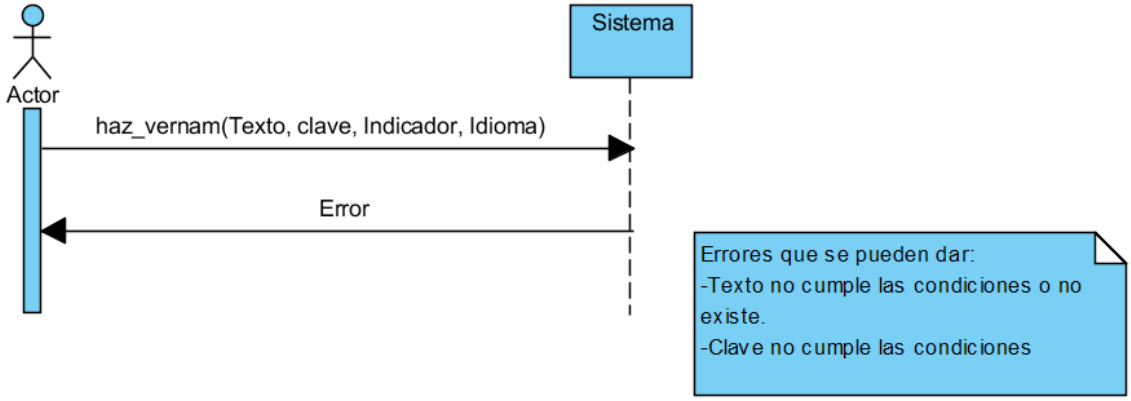


RF 4: Cifrado/descifrado con vernam.

En el caso de que no surja ningún error, tenemos el siguiente diagrama:



En caso contrario tenemos este diagrama:



Especificación de la interfaz pública de la API

HAZ_ATTBASH

`public String haz_Attbash (String texto_in, boolean indicador, String idioma)`

Cifra/descifra el texto dado en texto_in de acuerdo al indicador (true: cifra, false: descifra) según el método de attbash. El texto pertenece al idioma especificado en idioma.

Parámetros:

`texto_in` – Texto a cifrar/descifrar.

`boolean` – True: cifra. False: descifra.

`idioma` – Nombre del idioma al que pertenece el texto

Devuelve:

Texto resultado de cifrar/descifrar.

Throws:

`NullPointerException` – Si alguno de los parámetros es NULL.

`ErrorExcepcion` – Si alguno de los parámetros no es válido.

HAZ_CESAR

`public String haz_Cesar (String texto_in, boolean indicador, int clave, String idioma)`

Cifra/descifra el texto dado en texto_in de acuerdo al indicador (true: cifra, false: descifra) según el método de cesar.El texto pertenece al idioma especificado en idioma.

Parámetros:

`texto_in` – Texto a cifrar/descifrar.

`boolean` – True: cifra. False: descifra.

`clave`– Clave usada para el cifrado/descifrado

`idioma` – Nombre del idioma al que pertenece el texto

Devuelve:

Texto resultado de cifrar/descifrar.

Throws:

`NullPointerException` – Si alguno de los parámetros es `NULL`.

`ErrorExcepcion` – Si alguno de los parámetros no es válido.

HAZ_VIGENERE

`public String haz_Vigenere (String texto_in, boolean indicador, String clave, String idioma)`

Cifra/descifra el texto dado en texto_in de acuerdo al indicador (true: cifra, false: descifra) según el método de vigenere.El texto pertenece al idioma especificado en idioma.

Parámetros:

`texto_in` – Texto a cifrar/descifrar.

`boolean` – True: cifra. False: descifra.

`clave`– Clave usada para el cifrado/descifrado

`idioma` – Nombre del idioma al que pertenece el texto

Devuelve:

Texto resultado de cifrar/descifrar.

Throws:

`NullPointerException` – Si alguno de los parámetros es `NULL`.

`ErrorExcepcion` – Si alguno de los parámetros no es válido.

HAZ_VERNAM

`public String haz_Vernam (String texto_in, boolean indicador, String clave, String idioma)`

Cifra/descifra el texto dado en texto_in de acuerdo al indicador (true: cifra, false: descifra) según el método de vernam.El texto pertenece al idioma especificado en idioma. (nota: idioma irrelevante a la hora de descrifrar)

Parámetros:

`texto_in` – Texto a cifrar/descifrar.

`boolean` – True: cifra. False: descifra.

`clave`– Clave usada para el cifrado/descifrado

`idioma` – Nombre del idioma al que pertenece el texto

Devuelve:

Texto resultado de cifrar/descifrar.

Throws:

`NullPointerException` – Si alguno de los parámetros es `NULL`.

`ErrorExcepcion` – Si alguno de los parámetros no es válido.

Diseño funcional.

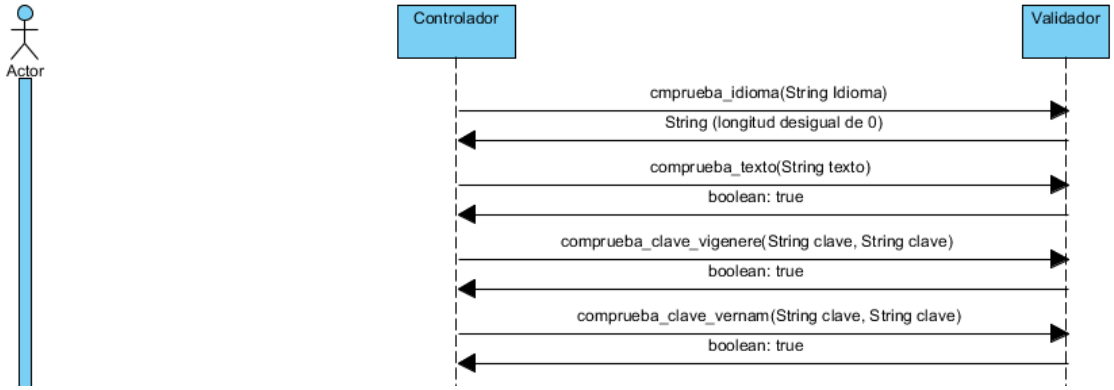
Diagramas de las Clases.

En todos los casos, las comprobaciones que se realizan han de superarse. Si alguna no se superase, se devolvería una excepción al usuario con el motivo de dicho error. Detallaremos en cada caso las comprobaciones a hacer.

Por ello, primero describiremos el funcionamiento y las comprobaciones de la clase validador.

Validador.

Este es conjunto de todas las validaciones posibles de la clase:



Lo primero que hace validador es ver con qué idioma estamos trabajando. Una vez identificado lo devolverá, para que sea usado por los métodos de controlador. En los casos en los que no haya error, el idioma será de longitud mayor de 0, por el contrario, si la longitud resulta ser 0, indicara que ha habido un error, y no se reconoce o soporta el idioma que nos han introducido por parámetros.

Comprobar texto se encarga de asegurarse que el texto cumple las condiciones requeridas (longitud, idioma y caracteres). Se usara en los 4 cifradores/descifradores. Se sirve del método comprueba alfabeto (privado de la clase validador) para comprobar si hay algún carácter no admitido.

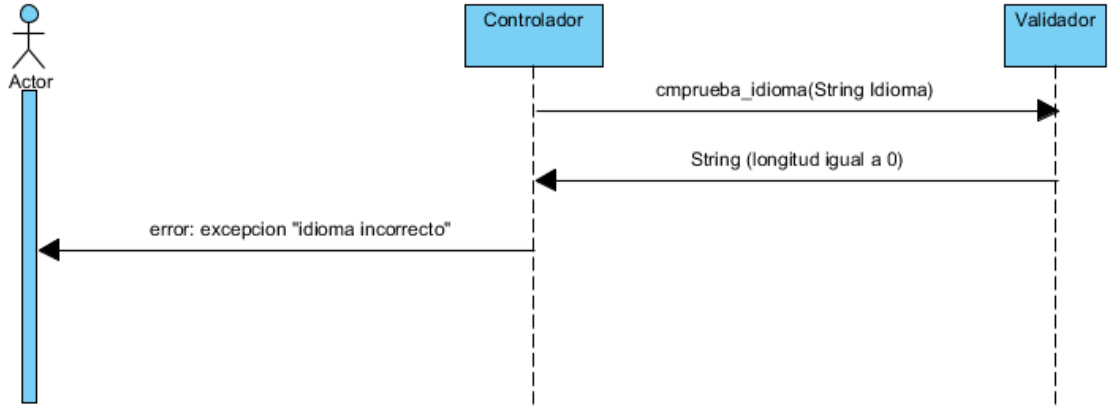
Comprueba clave vigenere se encarga de asegurarse que la clave cumple las condiciones requeridas. De igual manera, comprueba calve vernam hace lo propio para ese método de cifrado, ya que attbash no usa clave y la de cesar es numérica, siempre

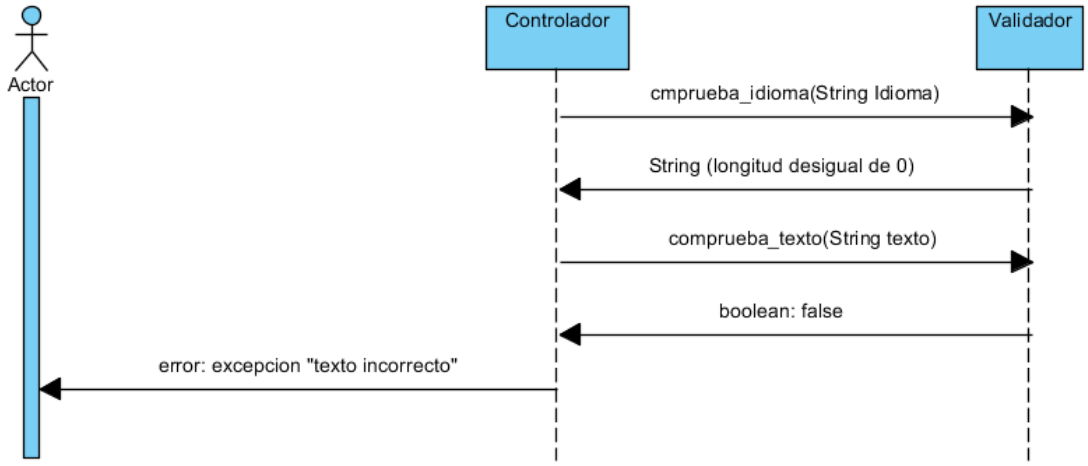
que el método reciba numero funcionara (y en caso de no recibirlo, el error saltaría antes de llegar aquí).

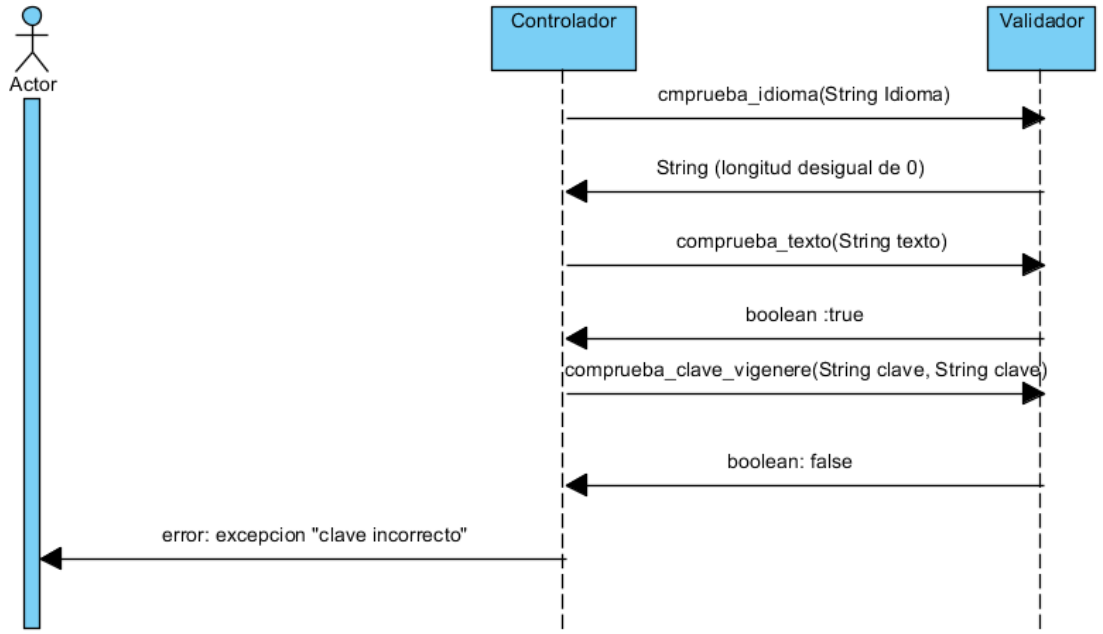
Dado que los nombres son bastante explicativos, diremos que se comprueba la validez de los parámetros y demás operandos usados en cada requisito. Por ejemplo, que para cifrar siempre se comprobara el idioma, después el texto, y después la contraseña si fuese necesario en ese caso.

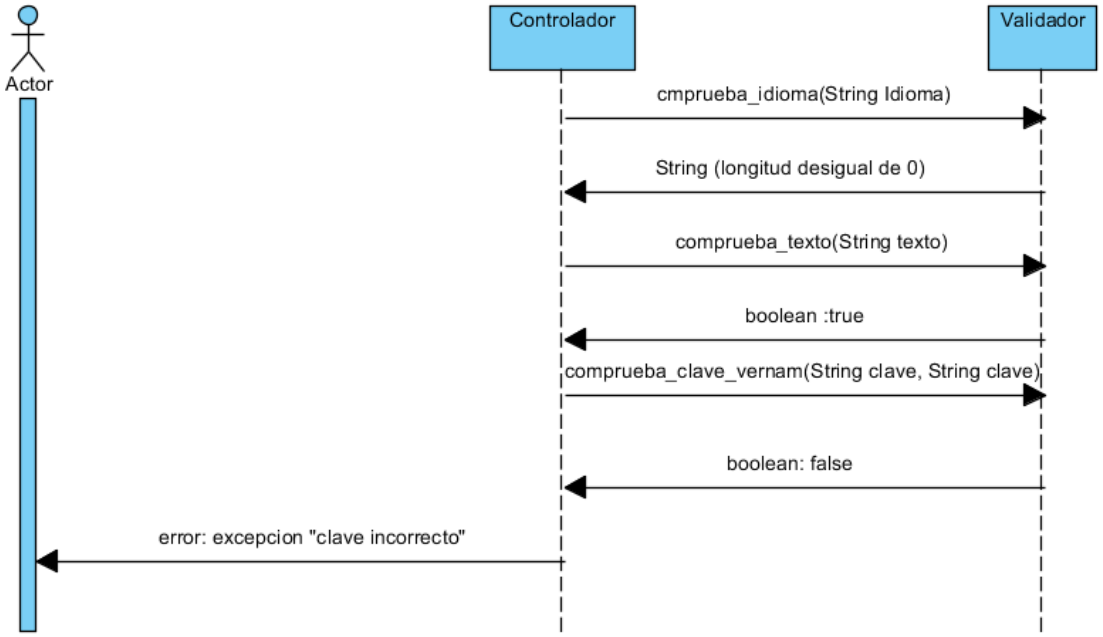
Si se completa el cifrado o descifrado de forma satisfactoria, se informara de ello al usuario por pantalla, además de devolver el texto cifrado/descifrado.

En caso de detectar errores, se procedería de la siguiente forma:









Nótese que en cada caso se comprobara todo lo necesario para esa operación concreta, pero que se mostrará tan solo el primer error detectado (el orden está definido por la primera imagen de validador)

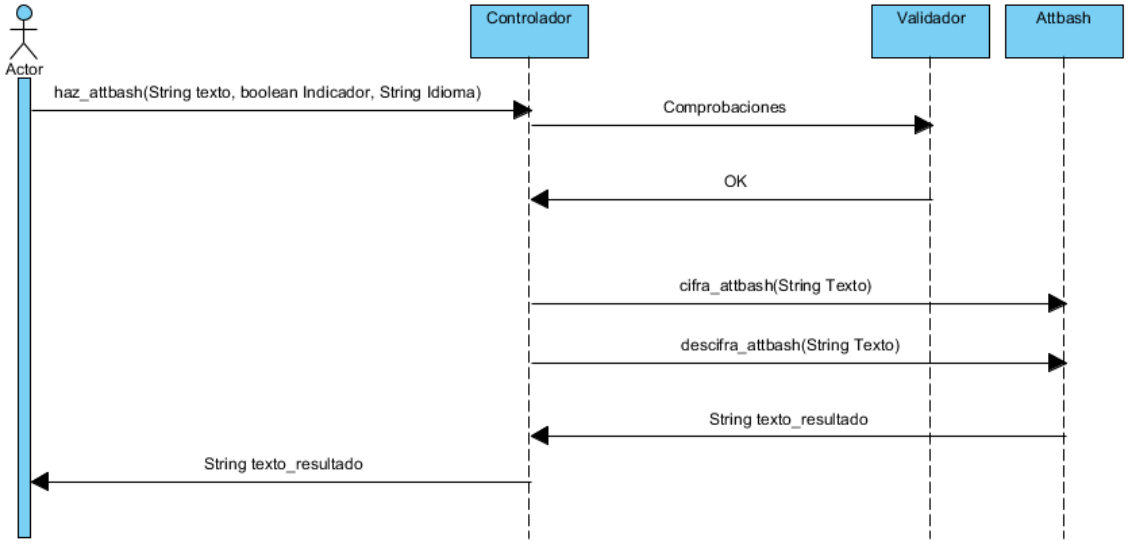
Y téngase presente, de ahora en adelante, que en cada diagrama de cada requisito funcional, la llamada a validador con "(Comprobaciones)" indica que se realizan todas las necesarias en cada caso, y que tan solo se sigue de haberse superado todas. En caso contrario, como ya hemos dicho, se procederá a devolver la excepción pertinente.

Para los siguientes diagramas, téngase en cuenta que tan solo se ejecutara una llamada desde controlador diferente de las comprobaciones. Es decir, en función de si

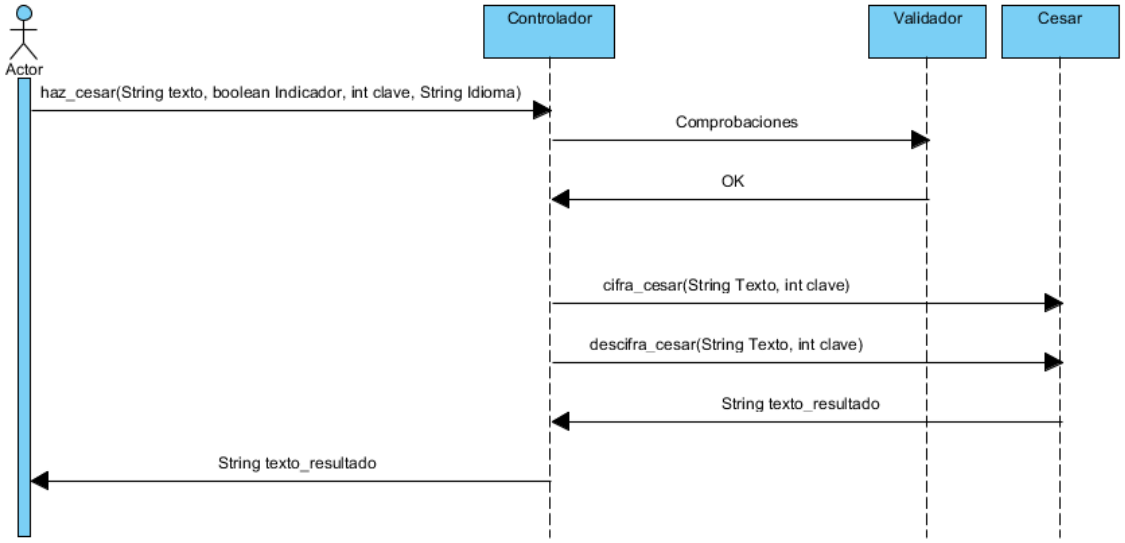
se nos pide cifrar u descifrar se ejecutara uno u otro, pero no ambos, aunque así aparezcan en los diagramas.

(NOTA: se omiten llamadas a métodos de tratamiento de cadenas de caracteres o arrays propios de java, tales como .length (), .toCharArray (),…)

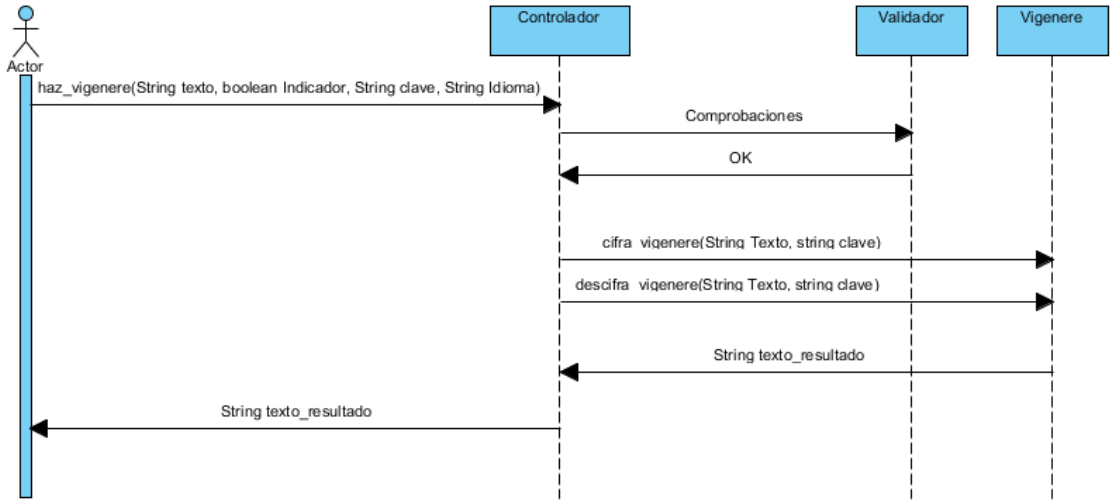
RF1:Cifrado/descifrado con attbash.



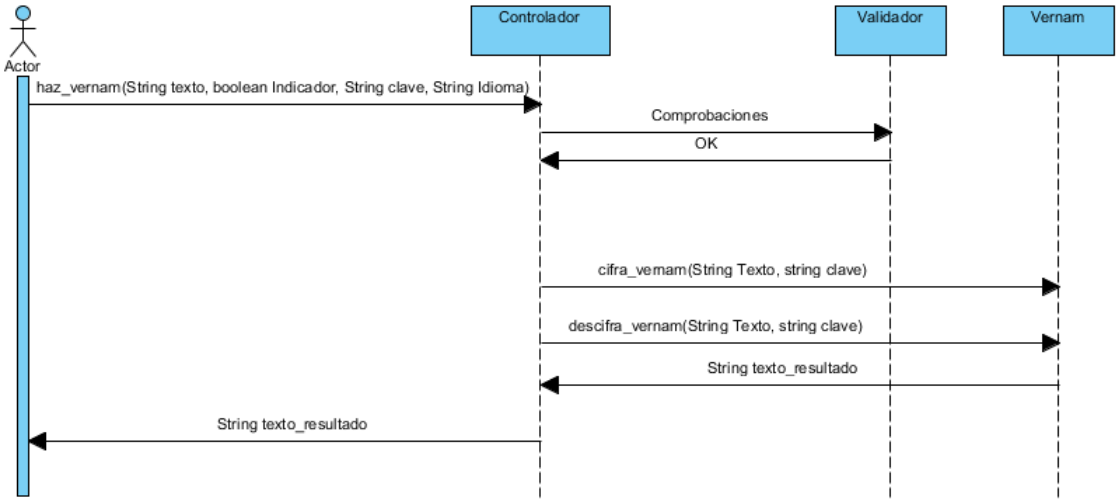
RF2: Cifrado/descifrado con cesar.



RF3: Cifrado/descifrado con vigenere.



RF4: Cifrado/descifrado con vernam.



Fichas crc.

Ficha CRC: Controlador.

Responsabilidades (cohesión):

La clase controlador tiene una cohesión baja, ya que los elementos de esta clase, no están relacionados con una única función. Está compuesta de métodos que la relación entre ellos es ligera.

Colaboraciones (acoplamiento):

El nivel de acoplamiento es medio, ya que llama a algunas clases, se relaciona con varias clases. Pero al tratarse de relación de “salida” digamos, no es tan mala como una de “entrada”, por así decirlo.

Dado que esta clase es la que organiza y controla todo, es lógico que se produzca esta situación.

Nombre de clase:	Controlador	
Responsabilidades:	Asegurarse de que todo es válido antes de seguir (parámetros...)	
	Pre-condición	Método con parámetros
	Post-condición	Si se valida se sigue, si no vuelta al user con error
	Colaboraciones	Validador
	Cifrado/descifrado con attbash	
	Pre-condición	Parámetros validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	Validador, attbash
	Cifrado/descifrado con cesar	
	Pre-condición	Parámetros validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	Validador, cesar
	Cifrado/descifrado con vigenere	
	Pre-condición	Parámetros validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	Validador, vigenere
	Cifrado/descifrado con vernam	
	Pre-condición	Parámetros validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	Validador, vernam

Ficha CRC: Validador.

Responsabilidades (cohesión):

La clase validador tiene una cohesión media. Esta clase se encarga de validar o comprobar que todo lo necesario para ejecutar una acción sobre una imagen es correcto. Por ello un método para cada cosa a comprobar (solo son dos realmente).

Colaboraciones (acoplamiento):

El nivel de acoplamiento es bajo, apenas hay relaciones (no llama a ninguna clase, y tan solo devuelve en cada caso, si algo es válido o no).

Nombre de clase:	Validador	
Responsabilidades:	Asegurarse de que todo es válido antes de seguir (parámetros...)	
	Pre-condición	Llamada con el parámetro a comprobar
	Post-condición	Si se valida devolvemos true, si no false.
	Colaboraciones	-

Ficha CRC: Attbash.

Responsabilidades (cohesión):

La clase cambia formato tiene una cohesión alta. Tan solo se encarga de cifrar con el método de attbash un texto.

Colaboraciones (acoplamiento):

El nivel de acoplamiento es bajo, apenas hay relaciones (no llama a ninguna clase, y tan solo devuelve un string).

Nombre de clase:	Attbash	
Responsabilidades:	Cifrar/descifrar con el algoritmo de attbash	
	Pre-condición	Parámetros correctos y validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	-

Ficha CRC: Cesar.

Responsabilidades (cohesión):

La clase redimensión tiene una cohesión alta. Esta clase se encarga de cifrar con el método de cesar un texto.

Colaboraciones (acoplamiento):

El nivel de acoplamiento es bajo, apenas hay relaciones (no llama a ninguna clase, y tan solo devuelve un string).

Nombre de clase:	Cesar	
Responsabilidades:	Cifrar/descifrar con el algoritmo de cesar	
	Pre-condición	Parámetros correctos y validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	-

Ficha CRC: Vigenere.

Responsabilidades (cohesión):

La clase rotar tiene una cohesión alta. Esta clase se encarga de cifrar con el método de vigenere un texto.

Colaboraciones (acoplamiento):

El nivel de acoplamiento es bajo, apenas hay relaciones (no llama a ninguna clase, y tan solo devuelve un string).

Nombre de clase:	Vigenere	
Responsabilidades:	Cifrar/descifrar con el algoritmo de vigenere	
	Pre-condición	Parámetros correctos y validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	-

Ficha CRC: Vernam.

Responsabilidades (cohesión):

La clase espejo tiene una cohesión alta. Esta clase se encarga de cifrar con el método de vernam un texto.

Colaboraciones (acoplamiento):

El nivel de acoplamiento es bajo, apenas hay relaciones (no llama a ninguna clase, y tan solo devuelve un string).

Nombre de clase:	Vernam	
Responsabilidades:	Cifrar/descifrar con el algoritmo de vernam	
	Pre-condición	Parámetros correctos y validados
	Post-condición	Texto cifrado/descifrado
	Colaboraciones	-

La única clase para la que no vamos a hacer ficha CRC es la clase ErrorExcepcion, ya que simplemente se usa para lanzar una excepción en la clase correspondiente.