








ICE String Checker

1. Interfaz IDL-Slice

```
module Demo
{
    interface Checker
    {
        int checkString(string str,int number);
    }
}
```

En el archivo Checker.ice se especifica las interfaces que serán implementadas, en este caso será el Checker, el cual proporciona el servicio de verificar una cadena, recibiendo por parámetro una cadena y un número positivo mayor a 0.

2.

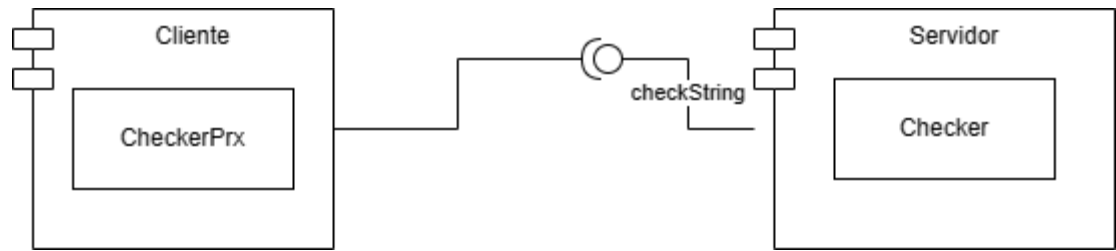
GRAPH	COMMIT MESSAGE
	Authors showing up
	Program working, maybe
	File added
	Server side functionality done
	Arguments checking working
	Project init
	Initial commit

Hubo un total de 7 versiones del proyecto, cada una con incrementos y desarrollo de las funcionalidades del programa. El commit inicial fue simplemente la inicialización del repositorio. El segundo fue donde se agregó la estructura básica del proyecto gracias al gradle init. En el tercero se implementó el algoritmo para verificar que los argumentos sean válidos, teniendo en cuenta las diferentes posibilidades. En el cuarto se desarrollo el lado del servidor sin ninguna funcionalidad. El quinto fue simplemente la adición del archivo de texto de pruebas. En la sexta versión se completó la funcionalidad para la verificación de la cadena junto con el algoritmo necesario para encontrar el siguiente número primo. El último commit simplemente se agregó una salida de texto con los autores en el caso que no se ingrese ningún argumento.

Diseño de componentes:

Como se especificaba en el problema los únicos componentes necesarios fueron el cliente y el servidor. La interfaz Checker permite la conexión entre el cliente y el servidor:

Mateo Rada
Santiago Arévalo



Pruebas:

Para probar la funcionalidad de los componentes se hicieron pruebas manuales comprobando que cumplieran los escenarios de uso mencionadas en el problema.

- Sin argumentos:

```
java -jar .\client\build\libs\client.jar
The system requires at least one positive integer
Mateo Rada Arias A00368693
Santiago Arevalo A00368494
PS C:\Universidad\7mo\Arquisoft\Checker>
```

- Un solo argumento:

```
PS C:\Universidad\7mo\Arquisoft\Checker> java -jar .\client\build\libs\client.jar 1
2
PS C:\Universidad\7mo\Arquisoft\Checker> java -jar .\server\build\libs\server.jar
129d3377-184e-4cdf-a582-b051a08d1822 VALID REQUEST
█
```

- Dos argumentos:

Archivo inexistente

```
PS C:\Universidad\7mo\Arquisoft\Checker> java -jar .\client\build\libs\client.jar 1 asd.txt
No such file/directory exists
```

Archivo existente

```
PS C:\Universidad\7mo\Arquisoft\Checker> java -jar .\client\build\libs\client.jar 1 prueba.
txt
1
PS C:\Universidad\7mo\Arquisoft\Checker> java -jar .\server\build\libs\server.jar
pollito-chicken INVALID REQUEST
█
```

Modificabilidad y portabilidad:

Para correr los componentes de manera remota en diferentes máquinas en primera instancia las máquinas donde se instalen deben estar conectadas a una red compartida (Ej: ZeroTier). Además se debe configurar el archivo de las máquinas en las cuales serán instaladas para poder reconocer los dispositivos en dicha red (Archivo local de hosts)

Por el lado del cliente se debe modificar el archivo config.client el cual lleva la configuración del cliente

```
#  
# The client reads this property to create the reference to the  
# "hello" object in the server.  
#  
Checker.Proxy=SimpleChecker:default -p 8000  
  
#  
# Uncomment to use the WebSocket transports instead.  
#  
#Hello.Proxy=hello:ws -p 8000  
  
# Only listen on the ZeroTier's LIASOn1 interface by default.  
#  
Ice.Default.Host= localhost
```

Modificando el Default.Host se puede poner la IP de la red compartida por ambos dispositivos.

Por el lado del servidor se debe hacer lo mismo en el server.config

```
#  
# The server creates one single object adapter with the name  
# "Hello". The following line sets the endpoints for this  
# adapter.  
#  
Checker.Endpoints= default -p 8000  
  
#  
# Only listen on the ZeroTier's LIASOn1 interface by default.  
#  
Ice.Default.Host= localhost
```

Claramente dependiendo de donde se instalen los componentes se debe poner la IP respectiva del dispositivo en el que se encuentran. Ambos servicios deben tener el mismo puerto en el apartado de Endpoints.

Código fuente:

[Mateo698/ICE-StringChecker \(github.com\)](https://github.com/Mateo698/ICE-StringChecker)