

# INF-343 Sistemas Distribuidos

Tarea N°2: “Uso de servicios de Middleware en una aplicación de negocio de protección de archivos”

## Antecedentes generales

La tarea en cuestión tiene como objetivo:

- Aplicar de forma práctica **gRPC**, un framework lenguaje-independiente desarrollado por Google para implementar “llamadas a procedimientos remotos” (Remote Procedure Call).
- Aplicar de forma práctica **RabbitMQ**, un “bróker” de mensajería y de transmisión de datos.
- Aprender a incorporar integraciones a proyectos ya existentes.

## Fechas importantes

- **Fecha de ayudantía:** lunes 6 de mayo.
- **Fecha de entrega:** jueves 30 de mayo.

## Descripción del problema

La empresa “DiSis” está muy orgullosa de su sistema de protección de archivos (obviamente desarrollado por ellos, sin ayuda externa), debido a sus buenos resultados y su facilidad de uso. Sin embargo, con la empresa en constante crecimiento, el servidor en el que está alojado no da a vasto para atender todas las peticiones, por lo que se requiere una reinversión de la solución.

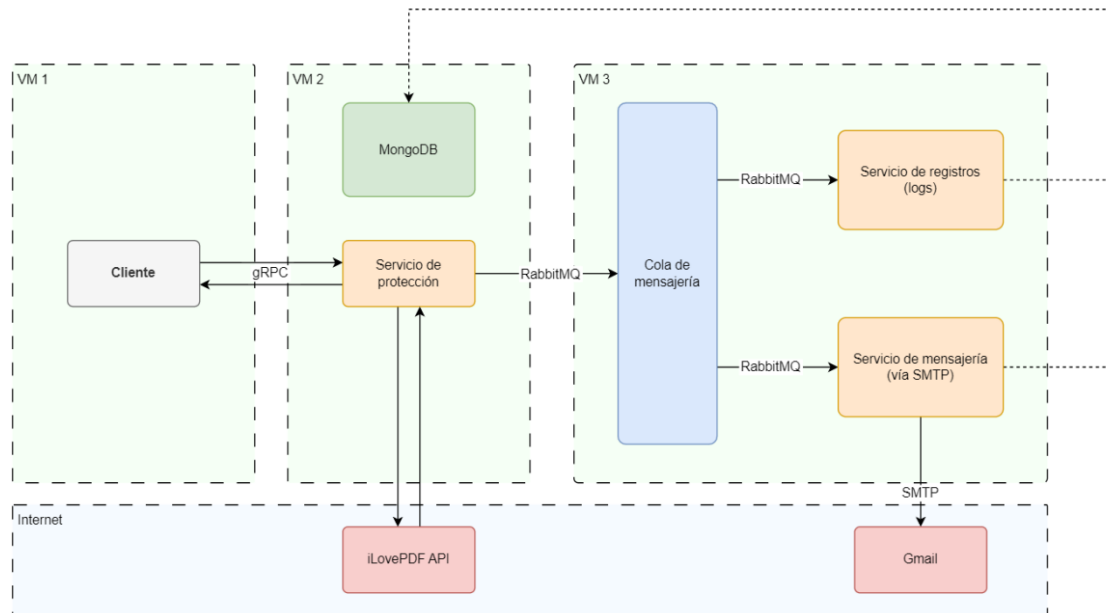
Es entonces cuando DiSis recuerda a cierto estudiante de Sistemas Distribuidos que podría ayudar a realizar una nueva implementación que tienen en mente.

## Descripción de la solución

Deberá utilizar sus conocimientos adquiridos en la tarea anterior para desarrollar un sistema distribuido que realice una protección de archivos PDF, pero implementado algunas funcionalidades adicionales que vienen de la mano con la implementación de RabbitMQ. En particular, el sistema que desarrollará deberá ser capaz de:

- Realizar la protección de archivos PDF desarrolladas en la tarea anterior.

- Guardar registros de las operaciones y sus resultados una vez hecho el procesamiento del archivo.
- Notificar al cliente para el cual el archivo está siendo protegido, a través de un mensaje por correo electrónico.



En la imagen se describe mejor el sistema en su totalidad. En este, cada máquina virtual implementa un protocolo de comunicación diferente entre ellas:

- La primera máquina (VM 1) alojará al cliente de la aplicación, el cual se encargará **únicamente de enviar los datos de cifrado del archivo, el correo de destino, y el archivo en cuestión**. En particular, deberá mandar:
  - El RUT del cliente. Este se utilizará para proteger el archivo PDF, **utilizando el RUT completo, sin guion, como contraseña**.
  - El correo electrónico del cliente o de quién desee ser notificado con el resultado de la operación.
  - El archivo que protegerá, únicamente **con formato PDF**.

Para realizar todo esto, deberá llamar al cliente **utilizando sólo los argumentos del programa** al correr el archivo.

- La segunda máquina (VM 2) contiene el servicio de protección en sí. Puede reutilizar el código que utilizó para comunicarse con la API de iLovePDF en la primera tarea. Además, el servicio ofrece también invocaciones remotas vía gRPC, en particular una función que reciba los datos de la VM 1 y luego obtener el resultado de la operación y la dirección en la que se guarda el archivo protegido.

- La tercera máquina (VM 3) es la responsable de:
  - Registrar cada operación relevante hecha por la VM 2 (tanto éxitos como errores), las cuales son: subir un archivo, inicializar un servidor de iLovePDF, subir el archivo al servidor de iLovePDF, realizar la protección del archivo y descargar el archivo desde el servidor de iLovePDF. Note que gran parte de las operaciones son las mismas que debe realizar para comunicarse con la API de iLovePDF, por lo que si su código anterior está bien organizado, trivializaría registrar cada paso.
  - Enviar una notificación al correo indicado en la primera máquina, indicando que su archivo ya está disponible.

## Algunas consideraciones

### Primera máquina virtual (VM 1)

Con gRPC, es capaz de mandar archivos, utilizando colecciones de bytes. De preferencia, utilice archivos pequeños. Recuerde definir correctamente las solicitudes y respuestas en sus archivos Proto Buffer para que esto no sea problema.

Sobre cómo es el input de los datos, usted deberá ejecutar `go run cliente <rut> <correo> <ruta>`, donde:

- `<rut>` es el RUT utilizado por el servicio para proteger el archivo.
- `<correo>` es el correo electrónico al que se notificará una vez terminada la operación.
- `<ruta>` es la ruta donde se encuentra el archivo a subir. Con esa ruta deberá obtener el archivo localmente y enviarlo por gRPC.

### Segunda máquina virtual (VM 2)

Por favor, tenga en cuenta que la dirección de correo electrónico será objeto de recibir mensajes, por lo que se le pide encarecidamente que utilice sólo correos los cuales sean de su propiedad, tenga permiso de su dueño y/o sean obtenidos desde un servicio de generación temporal de correos electrónicos.

### Tercera máquina virtual (VM 3)

El correo electrónico por enviar no necesita tener el archivo incrustado en él, sólo requiere notificar al cliente/persona de que el archivo fue protegido correctamente y **en que ruta de la VM 2 se encuentra** (no se complique demasiado con el mensaje, puede ser simplemente un texto con esa información). Dicho esto, tampoco es necesario mandar el archivo a través del bróker.

## Condiciones de entrega

Su entrega debe cumplir las siguientes condiciones:

- Su código debe estar escrito en el lenguaje Go, a excepción de los archivos Proto Buffer, que estarán escritos en el mismo IDL.
- Su tarea debe encontrarse en un repositorio de GitLab **distinto al de la tarea anterior**. En este deberán agregar al usuario “bsoto” como mantenedor.
- Su aplicación será evaluada únicamente con 4 ejecuciones:
  - `go run cliente`, el que contiene el cliente del sistema.
  - `go run proteccion`, que levantará el servicio de gRPC.
  - `go run registros`, que levantará el servicio de registros (logs).
  - `go run mensajería`, que levantará el servicio de mensajería vía SMTP.

La cantidad de ejecuciones no puede ser cambiada, *pero sí las instrucciones persé*; puede hacer que la ejecución sea, por ejemplo, `go run cliente.go`, siempre y cuando indique las nuevas instrucciones en un README.md.

- Deberá tener un archivo con las variables de entorno, llamado “.env”, dentro de su repositorio, con las siguientes variables:

Nombre	Descripción	Valor obligatorio (si aplica)
GRPC_HOST	Dirección IP del servicio de gRPC.	IP de la máquina correspondiente
GRPC_PORT	Puerto usado por el servicio de gRPC.	8080
RABBITMQ_HOST	Dirección IP del servicio de RabbitMQ.	IP de la máquina correspondiente
RABBITMQ_PORT	Puerto usado por el servicio de RabbitMQ.	5672
SENDER_EMAIL	Dirección de correo electrónico del emisor de la notificación (distinto al cliente notificado).	
SENDER_PASSWORD	Contraseña del correo electrónico del emisor de la notificación.	
MONGODB_URI	URL de conexión para el servicio de MongoDB.	mongodb://{user}:{pass}@{ip}:2707

PUBLIC_KEY	La clave pública otorgada en su panel principal del portal de desarrollador de iLovePDF API.	
------------	--	--

Note que “{user}”, “{pass}” y “{ip}” deben ser reemplazados por un usuario de la base de datos, la contraseña de este y la dirección IP de la máquina en la que está alojada.

La cantidad de variables puede aumentar a necesidad del estudiante.

- Su tarea debe encontrarse clonada en las tres máquinas, en las cuales cada una de ellas deberá correr los servicios que les corresponda.
- Si bien se sugiere un orden específico de las máquinas (VM 1, VM 2 y VM 3), usted puede utilizar el que más le acomode (sí, puede utilizar cómo VM 2 la máquina que ya tiene MongoDB configurado), siempre y cuando los cambios sean indicados en un README.md.
- Ante cualquier duda, comunicarse con el ayudante por los medios indicados y reiterados varias veces.