INF343 Sistemas Distribuidos

Tarea N°1: “Desarrollo de una aplicación distribuida simple para proteger documentos PDF usando servicios web”

Bastián Soto Iturra
26 de abril de 2024

Antecedentes generales

La tarea en cuestión tiene como objetivo:

- Aprender a programar con Go, el lenguaje concurrente desarrollado por Google.
- Aprender a desarrollar una API REST sencilla con **Gin**, un framework web para desarrollar APIs rápidamente.
- Familiarizarse con la base de datos no-relacional **MongoDB** y su uso en Go.
- Aprender a *consumir* servicios externos para la comunicación entre sistemas web.

Fechas importantes

- **Fecha de ayudantía:** lunes 8 de abril.
- **Fecha de entrega:** martes 7 de mayo.

Descripción del problema

Ha sido contratado por “DiSis”, una empresa de desarrollo de soluciones informáticas puntera del mercado, destacada por su eficiencia y una planificación y estructuración impecables.

Sin embargo, si la empresa tuviera que mencionar uno de sus puntos flojos, sin duda sería la *seguridad*. Y esa *seguridad* fue lo que les pasó factura; fueron atacados por un MITM (Man-in-the-middle), lo que dejó expuesto esta carencia en la empresa.

Para mantener la confidencialidad de los documentos que envían, DiSis ha optado por proteger sus PDFs con contraseña antes de enviarlos, y le han pedido a usted que haga un sistema que automatice este proceso. Como se ha inscrito hace poco a INF343, ha decidido utilizar las tecnologías y frameworks vistos en la asignatura, y crear una API REST para realizar este proceso automáticamente.

Descripción de la solución

Deberá construir una API REST, utilizando el lenguaje Go y el paquete *Gin*, un completo framework para construir APIs de forma rápida y sencilla. Además de esto, deberá utilizar el servicio de API entregado por iLovePDF para proteger estos documentos y así solucionar el problema.

Su API deberá ser capaz de:

- Registrar y consultar clientes de la empresa.
- Ingresar como usuario para obtener su ID.
- Consultar a la API de iLovePDF para proteger los archivos subidos.

Adicionalmente, también requerirá una aplicación cliente, que deberá conectarse a su API creada y realizar las llamadas correspondientes dependiendo de las opciones que elija el usuario que la utilice.

iLovePDF API

El servicio web conocido como iLovePDF facilita al usuario varias herramientas para manejar PDFs, entre ellas transformaciones a otros formatos (como PDF a Word), compresión, o –y lo que nos interesa en esta tarea– protección con contraseña, todas estas disponibles en su página web oficial, con una agradable interfaz visual.

Adicional a estas herramientas visuales, iLovePDF presenta una API para desarrolladores, que permite usar estos servicios a través de distintos lenguajes de programación, para los cuales tienen sus propios clientes, instalables como paquetes. El problema radica en que, entre estos lenguajes, no se encuentra Go, por lo que deberá utilizar la API convencional de iLovePDF.

Funcionamiento de la API de iLovePDF

La API de iLovePDF se compone de 5 endpoints, con un funcionamiento bien definido:

- **Authentication:** el primer paso para comunicarse con la API de iLovePDF es generar un token de autenticación. Si bien existe una manera de generar este token manualmente (lo que es recomendable para una aplicación de servidor, como sería nuestro caso), para facilitar las cosas utilizará el endpoint entregado por la API para generarlo.
- **Start:** siendo el primer endpoint del proceso como tal, Start utiliza el token para obtener un servidor y un ID del proceso de la tarea, donde podrá realizar el procesamiento de los archivos.

- **Upload:** ingresando el nuevo servidor como parte de la URI y el proceso de tarea generados en Start como parte del cuerpo de la petición, puede construir la petición completa para subir el archivo. Como resultado, obtendrá el nombre del archivo dentro del servidor.
- **Process:** una vez iniciado el servidor y subido el archivo, se puede realizar la tarea definida en la llamada a Start, con los parámetros que el usuario desee. Podrá ver el resultado del procesamiento en la respuesta.
- **Download:** cuando se termina la tarea, el endpoint deja a disposición del usuario el PDF, para poder guardarlo, si el navegador lo soporta, leer el archivo ingresando la contraseña definida antes.

Estos endpoints –al menos para el propósito de este curso– son utilizados de forma secuencial, por lo que la implementación dentro de su código deberá seguir este mismo patrón.

El diagrama anterior indica el paso a paso del proceso completo para utilizar las herramientas de la API, cuya referencia se encuentra en la web oficial de iLovePDF para desarrolladores, bastante bien explicada y con más detalles de cada endpoint y sus especificaciones.

El estudiante deberá crear una cuenta dentro de la plataforma para poder obtener su llave pública, que utilizará para generar los tokens.

Endpoints

Su solución deberá contemplar los siguientes endpoints listados a continuación. Cabe destacar que, si bien estos endpoints son necesarios, puede agregar más en caso de necesitarlo.

Endpoints de autenticación

La aplicación debe ser capaz de tener un sistema básico de autenticación; el usuario, miembro de DiSis, ingresa correo y contraseña, se revisa en la base de datos que este exista, y se retornan los datos del usuario (omitiendo, por supuesto, la contraseña). Además, el usuario tiene la posibilidad de darse de alta (registrase) en su aplicación, llenando los datos requeridos. Debe validar que el usuario ingresado no tenga ni un

correo ni RUT repetidos en la base de datos, sin embargo, ahórrese las validaciones complejas (como que el RUT exista -también llamado “módulo 11”-).

Login

Endpoint	127.0.0.1/login
Descripción	Ingresar al sistema como usuario de empresa.
Método	POST

Cuerpo	<pre>{
	“email”: “correo@gmail.com”,
	“password”: “contraseña123”
	}</pre>

Respuesta	**200:**
	<pre>{
	“data”: {
	“id”: “92823adss8234932”
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	}</pre>**404:**
	<pre>{
	“message”: “Usuario no encontrado”
	}</pre>

Observaciones	No complicarse con encriptar las contraseñas. Esto es una falta de seguridad gravísima, y no debería realizarse de esta manera. Sin embargo, no es el objetivo de la tarea la criptografía, por lo que podemos dejarlo pasar.

Registro

Endpoint	127.0.0.1/register
Descripción	Registra al usuario con los datos ingresados.
Método	POST

Cuerpo	{
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”,
	“password”: “contraseña123”
	}

Respuesta	**200:**
	{
	“data”: {
	“id”: “92823adss8234932”
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	}

Endpoints de clientes

Además de los endpoints para usuarios, deberá crear un CRUD básico para los clientes de la empresa.

Obtener los clientes de la empresa

Endpoint	127.0.0.1/api/clients
Descripción	Obtiene todos los clientes de la empresa.
Método	GET
Parámetros	?rut=12345678-9
Respuesta	**200 sin filtro de parámetros:** { “data”: [{ “id”: “21432asdasFSDGDSg” “name”: “Nombre”, “last_name”: “Apellido”, “rut”: “12345678-9”, “email”: “correo@gmail.com”

	}, {
	“id”: “2823asjsal203823”
	“name”: “Nombre 2”,
	“last_name”: “Apellido 2”,
	“rut”: “23456789-1”,
	“email”: “correo2@gmail.com”
	}
	]
	}
	200 con filtro de parámetros:
	{
	“data”: [
	{
	“id”: “21432asdasFSDGDSg”
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	]
	}

Observaciones	Note que se presentan parámetros. Estos son valores que sirven para filtrar los datos recopilados por la petición, y ese filtro deberá aplicarlo en el endpoint.

Obtener un cliente con un ID

Endpoint	127.0.0.1/api/clients/:id
Descripción	Obtiene un cliente dado un ID.
Método	GET

Respuesta	{
	“data”: {
	“id”: “21432asdasFSDGDSg”
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”

	} }

Crea un cliente

Endpoint	127.0.0.1/api/clients
Descripción	Registra un cliente con los datos indicados en el cuerpo.
Método	POST

Cuerpo	{
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}

Respuesta	**200:**
	{
	“data”: {
	“id”: “21432asdasFSDGDSg”
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	}

Actualizar un cliente, dado un ID

Endpoint	127.0.0.1/api/clients/:id
Descripción	Actualiza los datos de un cliente, con los datos enviados.
Método	PUT

Cuerpo	{
	“name”: “Nombre nuevo”,
	“last_name”: “Apellido nuevo”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}

Respuesta	**200:**
	{
	“data”: {
	“id”: “21432asdasFSDGDSg”,
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	}
	404:
	{
	“message”: “Usuario no encontrado”
	}

Observaciones	Debe considerar que: • La actualización puede ser parcial o completa. • Los datos devueltos son los datos previos a la actualización. • Es recomendable que el valor “:id” sea el generado por MongoDB.

Borrar un cliente

Endpoint	127.0.0.1/api/clients/:id
Descripción	Borra el cliente con el ID especificado.
Método	DELETE

Respuesta	**200:**
	{
	“data”: {
	“id”: “92823adss8234932”,
	“name”: “Nombre”,
	“last_name”: “Apellido”,
	“rut”: “12345678-9”,
	“email”: “correo@gmail.com”
	}
	}
	404:
	{

	`“message”: “Usuario no encontrado”` `}`
Observaciones	Debe considerar que: • La actualización puede ser parcial o completa. • Los datos devueltos son los datos previos a la actualización.

Procesamiento de PDF

Endpoint	127.0.0.1/api/protect
Descripción	Recibe un archivo y el ID de un cliente, y protege el mismo con los datos del usuario como contraseña.
Método	`POST`
Cuerpo	Un `form-data` que contiene los siguientes campos: **`id`** — `92823adss8234932` **`file`** — `Requerimientos.pdf`
Respuesta	El archivo protegido con contraseña.
Observaciones	Tenga en cuenta que: • El campo del form-data recibe un archivo, no un string o texto de la ruta. Para el manejo de este archivo, *se recomienda* guardar el archivo en una ruta especificada, y luego mandar el archivo. • Con respecto a la contraseña que debe utilizar, deberá usar el RUT del *cliente*, sin guion ni dígito verificador, como contraseña a la hora de utilizar la API de iLovePDF.

Algunas consideraciones sobre los endpoints

En general, todos los endpoints deben tener los campos indicados en cada uno, tanto en el cuerpo como en la respuesta, pero esto no implica que tenga que encasillarse a ese formato; puede incluir los campos que crea convenientes para lo que desee lograr luego en el cliente, siempre y cuando mantenga los que se muestran aquí como un mínimo.

Las estructuras mostradas en cada respuesta son un ejemplo de la respuesta recibida. Si bien tratan de ser consistentes entre sí, puede que encuentre inconsistencias entre librerías. Por ello, si bien los ejemplos tratan de simular la respuesta, puede que no sean totalmente fieles a los tipos de datos de las librerías utilizadas, como, por ejemplo, los IDs de MongoDB, o los parámetros de la URI en los endpoints correspondientes.

"DiSis" Client

Como último trabajo, se requiere de una aplicación cliente desarrollada en Go que se comunique con la API que han creado, y permita:

- Ingresar/registrarse en el sistema con una cuenta.
- Listar los distintos clientes de la empresa.
- Subir un documento al sistema y guardar el resultado en una copia protegida con contraseña.

Se sugiere utilizar la estructura mostrada en el ejemplo mostrado a continuación, pero puede modificarla para una mejor presentación.

Ejemplo de salida del cliente

```
go run main

Bienvenido al sistema de protección de archivos de DiSis.
Para utilizar la aplicación seleccione los números
correspondientes al menú.

Ingrese o regístrese
  1) Ingreso
  2) Registro
  3) Salir

> 2

Ingrese su nombre: Nombre
Ingrese su apellido: Apellido
Ingrese su RUT: 12345678-9
Ingrese su correo: correo@gmail.com
Ingrese su contraseña: contraseña123

¡Registro exitoso!
```

```
Ingrese o regístrese
   1) Ingreso
   2) Registro
   3) Salir

> 1

Ingrese el correo de su cuenta: correo@gmail.com
Ingrese su contraseña: contraseña123

¡Ingreso exitoso!

Menú principal
   1) Clientes
   2) Protección
   3) Salir

> 1

Menú clientes
   1) Listar los clientes registrados
   2) Obtener un cliente por ID
   3) Obtener un cliente por RUT
   4) Registrar un nuevo cliente
   5) Actualizar datos de un cliente
   6) Borrar un cliente por ID
   7) Volver

> 4

Ingrese nombre del cliente: Nombre
Ingrese apellido: Cliente
Ingrese RUT del cliente: 12345678-9
Ingrese el correo del cliente: empresa1@gmail.com

¡Cliente “Nombre” creado con éxito!

Menú clientes
   1) Listar los clientes registrados
```

```
   2) Obtener un cliente por ID
   3) Obtener un cliente por RUT
   4) Registrar un nuevo cliente
   5) Actualizar datos de un cliente
   6) Borrar un cliente por ID
   7) Volver

> 1

---
ID: 21432asdasFSDGDSg
Nombre: Nombre
Apellido: Cliente
RUT: 12345678-9
Email: empresa1@gmail.com
---

Menú clientes
   1) Listar los clientes registrados
   2) Obtener un cliente por ID
   3) Obtener un cliente por RUT
   4) Registrar un nuevo cliente
   5) Actualizar datos de un cliente
   6) Borrar un cliente por ID
   7) Volver


> 3

Ingrese el RUT a buscar: 12345678-9

---
ID: 21432asdasFSDGDSg
Nombre: Nombre
Apellido: Cliente
RUT: 12345678-9
Email: empresa1@gmail.com
---
```

```
Menú clientes
   1) Listar los clientes registrados
   2) Obtener un cliente por ID
   3) Obtener un cliente por RUT
   4) Registrar un nuevo cliente
   5) Actualizar datos de un cliente
   6) Borrar un cliente por ID
   7) Volver


> 7

Menú principal
   1) Clientes
   2) Protección
   3) Salir

> 2

Escriba el ID del cliente objetivo: 21432asdasFSDGDSg
Escriba la ruta donde se encuentra el archivo (incluya el
nombre): pdfs/archivo.pdf

¡Protección exitosa! Su archivo se encuentra en:
pdfs/archivo.pdf

Menú principal
   1) Clientes
   2) Protección
   3) Salir


> 3

¡Vuelve pronto!
```

Persistencia de datos

Como ya se ha mencionado en el documento, usted deberá utilizar MongoDB para persistir los datos de los usuarios y clientes, además *de los tokens de autenticación que deberán utilizar posteriormente con* ***Bearer*** para todas las peticiones de iLovePDF.

Luego de que termine el desarrollo de su aplicación y API, MongoDB debe ser instalada en la máquina virtual *de forma local* para comunicarse con esta. Para ello *sí* podrá utilizar módulos/paquetes a los documentos, como los drivers cliente que existen para Go en la documentación oficial.

Consideraciones generales

- Para simplificar los inputs, considere que siempre se envían datos correctos; es decir, no requiere que valide los inputs.
- Asuma que el RUT sigue la siguiente estructura: 12345678-9.
- La autenticación de la API que está creando es básica, en el sentido que debe buscar en la base de datos algún documento que coincida con los datos entregados y si se encuentran y coinciden las contraseñas, debería ser suficiente.

Condiciones de entrega

- Su código debe ser escrito, tanto para la API como el cliente, en Go. Puede utilizar las librerías que quiera, excepto aquellas que reemplacen el uso de Gin para crear la API.
- La tarea debe encontrarse en un repositorio del GitLab, para luego subir ese enlace al Aula de la asignatura. En ese repositorio deberán agregarme como usuarios para poder revisar su código.
- Su aplicación sólo se evaluará con dos ejecuciones, las cuales son:
    - `go run api`, módulo que contendrá el código de la API.
    - `go run main`, módulo que leerá el input del usuario y realizará las llamadas a la API a medida que lo solicite el usuario.
- Deberá ingresar todas las variables de entorno dentro de un archivo "`.env`", el cual debe contener, cómo mínimo:

Nombre	**Descripción**	**Valor obligatorio (si aplica)**
`HOST`	Dirección de la API creada.	127.0.0.1

PORT	Puerto donde se ejecuta la API creada.	5000
MONGODB_URI	URI que contiene las instrucciones de conexión de MongoDB.	mongodb://{user}:{pass}@127.0.0.1:27017
PUBLIC_KEY	La clave pública otorgada en su panel principal del portal de desarrollador de iLovePDF API.	

Note que "{user}" y "{pass}" deberá reemplazarlo por un usuario de la base de datos, *el cual es distinto al usuario de la aplicación*.

- Adicionalmente, su tarea debe estar subida en la máquina virtual entregada por el ayudante. En particular, debe estar presente en la carpeta raíz del usuario, también conocida como "home".
- Ante cualquier duda o consulta, puede comunicarse a través de Discord, o, en su defecto vía correo electrónico.

Sobre las máquinas virtuales

A cada grupo de tareas se le asigno un total de tres máquinas virtuales, *independientes entre sí*, ordenadas por una numeración en la primera columna de la tabla. Las contraseñas de estas se pueden encontrar en los mensajes enviados a cada uno de sus grupos.

Su grupo deberá clonar el repositorio de GitLab (o GitHub, dependiendo de su grupo y luego de consultar con el estudiante) en el directorio raíz ***de la primera máquina virtual***. Es decir, configurar la máquina con el menor ID y subir ese repositorio allí.

Además, considere lo siguiente.

- MongoDB sólo se configurará en la máquina donde ejecutará su código. Esto es, instalar MongoDB en la primera máquina de su lista.
- *No debe configurar MongoDB en el resto de las máquinas, sólo en la que se utilizará para esta tarea*. No seguir esta instrucción podría llevar a un descuento.
- Las otras máquinas no son necesarias de inicializar (esto ocurre cuando entras a una máquina nueva por primera vez), ya que sólo utilizarán la primera. De todas formas, si ya la inicializaron, no hay problema.