**Universidad Nacional de San Juan**

****

**Facultad de Ciencias Exactas, Físicas y Naturales**

**Licenciatura en Ciencias de la Computación.**

**Informe del Proyecto Integrador.**

Asignatura: Lógica y Optimización Aplicada.

Alumnos:

García Sebastián, Registro: 21434

Jofré Santiago, Registro: 21260

**Índice:**

1. Explicación breve del problema a resolver
2. Explicación breve del modelo del problema planteado en CPLEX Studio

2.1 Variable de decisión.

2.2 Función objetivo y restricciones.

2.3 Resultado tras ejecutar el modelo.

1. Implementación del problema

3.1 Lenguajes, dependencias y APIs utilizadas.

3.2 Detallamiento de los componentes del sistema.

3.3 Ejecución normal del sistema.

3.4 Diagramas del sistema.

3.4.1 Casos de uso.

3.4.2 Diagramas de secuencia.

3.5 Primer prototipo de prueba

En el presente proyecto se creó un prototipo de un sistema el cual recibe datos de entrada para el problema de la mochila y se envían a IBM Cloud para resolverlo utilizando Cloud Object Storage y Wattson Machine Learning.

# 1. Explicación breve del problema a resolver

El ejemplo de problema utilizado para el prototipo es un caso del Problema de la Mochila, que busca maximizar el valor total de los objetos que se pueden guardar, restricto a la capacidad máxima de los recursos con los que cuenta la mochila.

En el problema se plantean K recursos (de los cuales uno por ejemplo puede ser peso) y N ítems los cuales tienen asociados la cantidad que consumen de cada recurso junto con el valor que aportan. Como resultado el sistema muestra la cantidad a cargar para cada uno de los ítems para maximizar el valor total de la mochila (En este prototipo no realizamos chequeo de errores en los datos de entrada).

# 2. Explicación breve del modelo del problema planteado en CPLEX Studio

## 2.1 Variable de decisión

La variable de decisión para este modelo es la que se muestra a continuación:



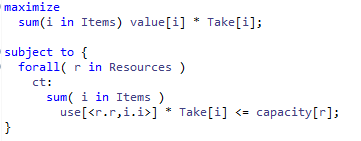
Donde Take representa la cantidad que se tomará para cada uno de los ítems u objetos. Para este modelo, se presentan 12 ítems, cada uno con un peso determinado por cada unidad de ese ítem.



‘MaxValue’ es una variable que indica el máximo entre las capacidades de los recursos de la mochila (esto se detalla en el siguiente apartado).

## 2.2 Función objetivo y restricciones

La función objetivo y sus restricciones serán:

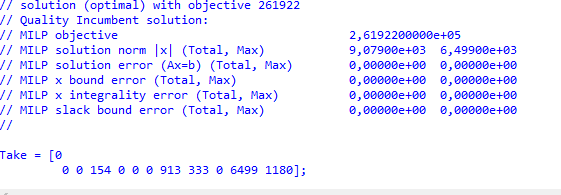


Dónde ‘value’ indica el peso/valor que tiene asociado ese ítem por unidad; ‘‘use’ indica el uso que hacen estos 12 ítems por cada recurso. En este modelo se presenta una cantidad de 7 recursos (bolsillos) de la mochila, cada uno con una capacidad máxima definida, tal como se muestra en la variable ‘capacity’.

Se observa que la función objetivo será maximizar, para cada ítem, la cantidad máxima que se puede tomar de esos ítems y colocarlos en la mochila, teniendo en cuenta la restricción de que dicha cantidad no sobrepase la capacidad máxima para cada uno de los recursos, teniendo en cuenta la variable ‘use’.

## 2.3 Resultados tras ejecutar el modelo

El resultado es la matriz Take, donde indica la cantidad máxima que se puede tomar para cada ítem sin que sobrepase las capacidades de los recursos de la mochila.



# 3. Implementación del problema

## 3.1 Lenguajes, dependencias y APIs utilizadas

* Para el apartado de backend (conexión con las APIs de IBM y tratamiento de datos), se utilizó el lenguaje JavaScript con Node.js
* Para el apartado de frontend (formularios de carga de datos) se empleó el lenguaje HTML.
* El módulo Express para la creación de puntos de entrada, como también para servir los archivos del frontend.
* ibm-cos-sdk como módulo suministrado por IBM para el manejo de la API de COS (Cloud Object Storage).
* El módulo Axios para realizar peticiones a la API de Wattson Machine Learning.

## 3.2 Detallamiento de los componentes principales del sistema

El servidor está compuesto de tres componentes principales: El módulo Controller, el módulo Service y la carpeta “public”, donde dicha carpeta contiene los archivos asociados al front-end de los formularios, como los ficheros .html, la hoja de estilos (.css) y los scripts de JavaScript asociados a la lógica de los formularios.

El módulo Controller (controlador) es el que se encarga de recibir las peticiones para cargar datos a COS y para realizar la resolución del problema, así como también servir los archivos del front-end que se encuentran dentro de la carpeta ”public”.

El módulo Service (servicio) es aquel que se comunica con las APIs de COS y Wattson Machine Learning, para enviarles los datos a cargar o leer de COS o para ejecutar los Jobs y resolver el problema.

## 3.3 Ejecución normal del sistema

La ejecución normal del sistema consiste en que el usuario ingresa primero el número de ítems a considerar y el número de recursos con los que contará la mochila (todo este proceso se lleva a cabo a través del formulario de index.html, que forma parte del apartado front-end del sistema). Luego, ingresará el valor/peso para cada ítem, la capacidad máxima de cada recurso y el consumo de cada recurso por cada ítem. Posteriormente, se envían al módulo service y se envían a COS utilizando la función putObject de ibm-cos-sdk.

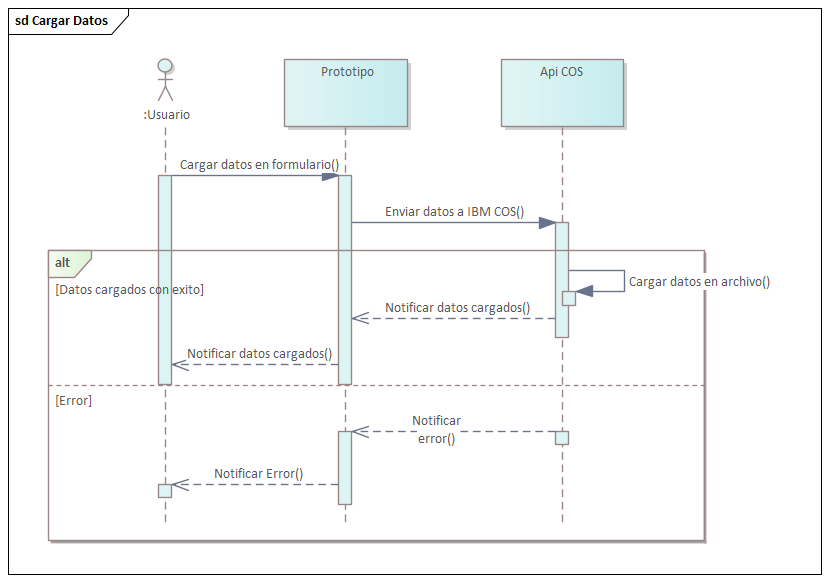
Luego se ejecuta un nuevo Job en Wattson Machine Learning a través de una petición de axios (nota: uno de los headers es un token de acceso generado por ibm, no se implementó en el prototipo un refresco automático del token, por lo que cada hora hay que generar uno nuevo a mano) el sistema se queda a la espera de que finalice el Job, lo cual comprueba cada 5 segundos (mediante un timer). Finalizada la ejecución los resultados se cargan dentro de un archivo solución en COS, donde finalmente son leídos y expuestos al usuario.

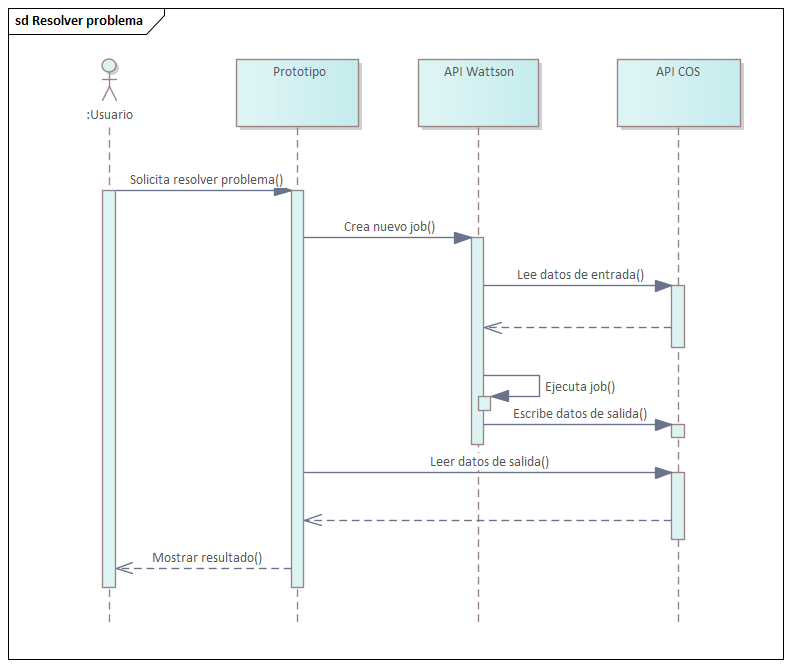
La creación del deploy y configuración del mismo dentro de Wattson se realizó de forma manual, de modo que el sistema solo interactúa con los archivos dentro de COS y ejecuta nuevos jobs. En caso de que sea imposible encontrar una solución con los datos de entrada expuestos se le avisará al usuario.

## 3.4 Diagramas del sistema

### 3.4.1 Casos de uso

### 3.4.2 Diagramas de secuencia





## 3.5 Primer prototipo de prueba

Para el testeo de las conexiones de IBM con la API, funcionamiento de los formularios y resolución de problemas, se tomó como modelo de pruebas el Problema de las Bebidas, el cual fue presentado durante el cursado de la asignatura, debido a su simplicidad y por la poca cantidad de datos que se requieren para hacer funcionar el modelo.