**OBJETIVO DEL DOCUMENTO**

La iniciativa que se presenta en este documento la compone esencialmente

evidenciar el proceso de la solicitud de Distribución de artículos desde centros de acopio y la propuesta que realizamos para cumplir dicho requerimiento.

## 1. CLASIFICACIÓN DE LA INICIATIVA

Requerimiento del negocio.

Nueva funcionalidad.

## 2. DESCRIPCIÓN DEL PROBLEMA

Transporters Inc necesita desarrollar un algoritmo que indique cuáles contenedores se deben seleccionar para su transporte de tal forma que podamos obtener el mayor KPI posible sin exceder el presupuesto establecido.

## 3. DESCRIPCIÓN DEL PROCESO ACTUAL

N/A.

## 4. ALCANCE DE LA INICIATIVA

Tener un proceso funcional el cual le permita a Transporters Inc mediante un

conjunto de pasos sencillos despachar sus contenedores con el mejor KPI.

## 5. REQUERIMIENTOS FUNCIONALES

* RF\_01: Crear una API REST.
* RF\_02: Crear un servicio “/containers” en donde se realice el despacho de contenedores.
* RF\_03: Crear un servicio “/stats” que devuelva las estadísticas de los despachos.
* RF\_04: Anexar una base de datos.
* RF\_05: Tener en cuenta que la API puede recibir fluctuaciones agresivas de tráfico (Entre 100 y 1 millón de peticiones por segundo).

## 6. CRITERIOS DE ACEPTACION

* RF\_01:
  + Crear un programa con alguno de los siguientes lenguajes: Java, Golang, .Net, NodeJS
  + Subir código fuente en un Repositorio Privado (Github, Gitlab, Bitbucket, etc) con acceso.
  + Desplegar esa API en un proveedor Cloud como Google Cloud Platform, Amazon AWS, etc..
* RF\_02: Crear servicio mediante un HTTP POST que solicite un Json el cual tenga el siguiente formato:

POST → / containers

{

  "budget": 1508.65,

  "data": [

    {

      "name": "C1",

      "transportCost": 571.4,

      "containerPrice": 4744.03

    },

    {

      "name": "C2",

      "transportCost": 537.33,

      "containerPrice": 3579.07

    },

    {

      "name": "C3",

      "transportCost": 434.66,

      "containerPrice": 1379.26

    },

    {

      "name": "C4",

      "transportCost": 347.28,

      "containerPrice": 1700.12

    },

    {

      "name": "C5",

      "transportCost": 264.54,

      "containerPrice": 1434.8

    }

  ]

}

* + En caso de verificar el mejor despacho, debería devolver un HTTP 200-OK, con la respuesta de los contenedores que se deben despachar
  + En caso contrario se debe mostrar el mensaje de error de la validación pertinente.
  + Si hay un problema interno en la aplicación devolverá un HTTP 500- Internal Server Error, con el mensaje de error.
* RF\_03: Crear servicio mediante un HTTP GET que devuelva un Json el cual tenga el siguiente formato:

GET → /stats

{

    "containersDispatched": 10023.22,

    "containersNotDispatched": 2814.06,

    "budgetUsed": 1508.65

}

* + Para mejorar la búsqueda se puede realizar por filtros de fecha o rango el cual no es obligatorio.
* RF\_04: Se deben guardar los despachos verificados con la API, en cualquier motor de Base de Datos.
* RF\_05: Pruebas unitarias con una cobertura > 80% con reporte.

## 7. REGLAS DE NEGOCIO

* RF\_02:
  + Los valores de los campos (Budget, transportCosty containerPrice) deben ser numéricos con valores decimales y no se puede enviar valores negativos.
  + El valor total de los contenedores enviados a despachar debe ser mayor o igual que el presupuesto enviado.
  + No puede existir contenedores a despachar repetidos.
  + Limitar el tamaño de la lista de contenedores a despachar.
  + No enviar contenedores ya despachados.
* RF\_03: Solo se debe mostrar un único registro con valores numéricos y con dos dígitos decimales.

## 8. BENEFICIOS ESPERADOS

Desarrolla el algoritmo de la manera más eficiente posible y tener en cuenta buenas prácticas de desarrollo, calidad de código y patrones.

## 9. IMPLEMENTACION

### 9.1. DIAGRAMAS

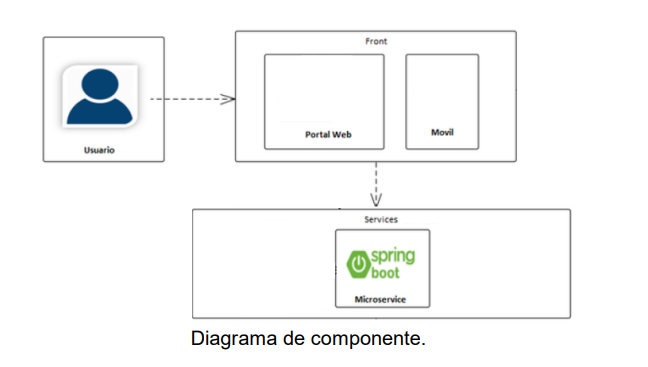


Diagrama de componente.

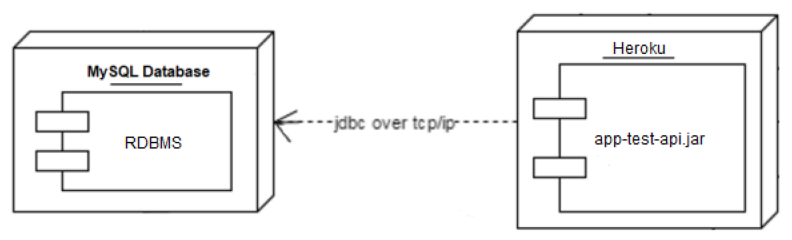
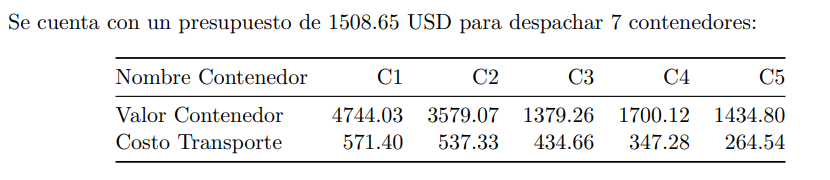


Diagrama de despliegue.

### 9.2. ILUSTRACIONES:

Para cumplir con el requerimiento funcional (RF\_01) y entender los criterios de aceptación y las reglas del negocio, se debe interpretar la siguiente ilustración:



### 9.3. RAZONAMIENTO:

Para cumplir con el objetivo el requerimiento, se plantea la siguiente implementación:

* Crear un nuevo repositorio en github y público.
* Crear un microservicio en el lenguaje JAVA utilizando como Framework Spring Boot.
* Crear un esquema, tabla y funciones con la estructura que se entrega en el archivo DDL.sql
* Crear el Api y el Controller con dos métodos que contengan las siguientes declaraciones:
  + containers:
    - RequestMapping = "/ containers ".
    - Method = POST.
    - RequestBody.
    - ResponseEntity.
  + Stats:
    - RequestMapping = "/stats".
    - method = GET.
    - RequestParam= startDate y endDate (campos no obligatorios).
    - ResponseEntity.
* Crear el Business de los métodos que consume el controller.
* Crear los DTO request y response necesarios.
* Crear el DAO para insertar y consultar la información en la base de datos.
* Crear una clase Validations en útil donde se realice las respectivas validaciones.

## 10.REQUERIMIENTOS NO FUNCIONALES

* Latencia
* Escalabilidad
* Alta disponibilidad
* Confiabilidad

## 11.ENTREGABLES

* Código fuente en el repositorio github.
* Instrucciones de cómo ejecutar el programa en local y cloud.
* Documento de pruebas unitarias
* DDL.