**Desarrollo Base:**

Se desea implementar una aplicación para el registro de empresas de transporte, vehículos y conductores. El administrador requiere registrar nuevas empresas. El usuario de la empresa de transporte requiere afiliar y desafilar vehículos a la empresa, así como llevar el registro de los conductores vinculados a los vehículos. Los vehículos pueden afiliarse y desafilarse varias veces a la empresa. Los conductores pueden vincularse a más de un vehículo.

De cada empresa se desea registrar la siguiente información: tipo y número de documento de identificación, el nombre completo, dirección, ciudad, departamento, país y teléfono. Adicional se requiere la siguiente información del representante legal: tipo y número de documento de identificación, el nombre completo, dirección, ciudad, departamento, país y teléfono.

De cada vehículo se desea registrar la información de placa, motor, chasis, modelo, fecha de matrícula, pasajeros sentados y de pie, peso seco, peso bruto, cantidad de puertas, marca y línea.

De los conductores se requiere registrar el tipo y número de documento de identificación, el nombre completo, dirección, ciudad, departamento, país y teléfono.

**Integraciones y restricciones:**

Al final del día se requiere reportar estos registros a un tercero, este tercero expone servicios soap, los cuales responden con una latencia superior a un minuto

La información en cuestión se suma como complemento a un modelo de bodegas de datos para hacer inteligencia de negocios.

Aunque inicialmente se espera entregar la consulta sin ningún tipo de restricción, eventualmente será necesario en la medida en que los registros aumenten, realizar algún tipo de optimización en pro de que la consulta siempre visualice la información como máximo en un segundo.

**ENTREGABLES**

1. Definir diseño y arquitectura para esta solución:

* Atributos de calidad relevante (Mínimo 5), con su debida resolución.
* Acuerdos de nivel de servicio.
* Diagramación que ilustre la arquitectura solución (4+1 más los que se requieran).
* Supuestos
* Tecnologías sugeridas para resolver la solución (On premise y cloud).

1. Codifique con base al contexto dado, la creación de una nueva empresa de transporte utilizando registros en memoria, asociando conductores, tras hacer la creación confirmar al usuario la operación exitosa (tanto en front, como el back comunicándolo a través de api).

**Instrucciones:**

* Para realizar el punto 3 se puede usar el lenguaje de su preferencia, junto con el IDE de su preferencia.
* Crear un repositorio GIT personal y adjuntar el link a dicho repositorio, junto con el código crear un archivo readme.md que explique cómo poner a funcionar el entregable, dejando claro que debe funcionar, con la documentación requerida del punto 1

**SOLUCION DEL PLANTEAMIENTO**

**CARGO: Arquitecto de software Junior**

**CANDIDATO: Antonio Renteria Diaz**

**Atributos de calidad**

**Integridad**

El 100% de la información debe mantenerse íntegra ya que van a ser datos sensibles de clientes, por lo tanto, el sistema debe custodiar y garantizar que no sea posible cambiarse o eliminarse por ningún motivo.

**Disponibilidad**

La aplicación debe contar con una disponibilidad de 99.999%. Garantizar que tanto la actividad que se realicen en las empresas, conductores y vehículos debe estar habilitada para realizar cambios en cualquier momento, buscando que las horas de mantenimiento no afecte la operación.

**Elasticidad (Escalabilidad)**

Los servicios deben permitir configuraciones de elasticidad (aumento de recursos de hardware) por instancia de forma horizontal:

**Interoperabilidad**

Los servicios deben permitir la integración de diferentes componentes como se menciona en el requerimiento en cuanto a los terceros y además debe contar un 95% de eficiencia en el intercambio de información entre los servicios de integración, donde la duración de ésta no debe tardar más de 5 segundos contando con la disponibilidad de los servicios origen y destino.

**Desempeño**

La aplicación dispondrá de un desempeño de un 90%, el tiempo de respuesta de las API para los ingresen y máximo de 5 segundo para las peticiones. Se debe garantizar la eficiencia de los recursos destinados para la aplicación siendo necesario fortalecerlos en momentos de alta concurrencia utilizando la elasticidad promovida por la orquestación de contenedores.

**Acuerdos de nivel de servicio**

* Tiempos medios de solución a los inconvenientes que se presenten con la funcionalidad.
* Rango de horas de soporte para la solución de los errores.
* Niveles de para atender los inconvenientes.
  + **Nivel 1:** Atención de dudas, aclaraciones de las funcionalidades de la herramienta que se puedan resolver.
  + **Nivel 2:** Atención de errores, comportamientos diferentes en la aplicación, que puede ser necesario el acompañamiento del técnico.
  + **Nivel 3:** Atención de afectaciones masivas, temas de comunicación entre servicios, seguridad, caídas de algún servicio.
* Planes de contingencia y continuidad
  + Plan preventivo
* Realizar la gestión del riesgo, detectar acciones de mejora para los de nivel alto y medio y seguimiento para los de bajo.
* Contar con aplicaciones de monitoreo a los servicios.
* Respaldo a la base de datos, realizando pruebas de que no estén corruptos y puedan ser usados en cualquier momento.
* Respaldo de la configuración de máquinas virtuales y políticas de seguridad.
  + Plan de emergencia
* Contar con el apoyo de técnicos del api y servidores para realizar el monitoreo y estabilidad de los servicios lo más rápido posible.
* Contar con página de estado de los servicios donde se pueda mostrar si están funcionando correctamente.
  + Plan de recuperación
* Hacer uso de los respaldos almacenados, tanto de datos como código para restablecer el servicio.
* Reemplazo de componentes de ser necesario si existen daños.

**Diagramación que ilustre la arquitectura solución (4+1 más los que se requieran).**

**Vista Global de la Arquitectura**

**Diagrama

Descripción generada automáticamente**

En la anterior vista podemos apreciar la arquitectura global propuesta con la cual se espera mostrar cómo es la comunicación de los diferentes componentes, servicios y la bases de datos planteada, donde se propone que desde la aplicación web se pueda acceder a cada uno de los componentes pero antes pasar por un balanceador de carga que permita equilibrar las solicitudes que se realicen, posteriormente se llega a los diferentes componentes que consumen los servicios API los cuales tiene cada uno su propio esquema de base de datos permitiendo así mantener un desacoplamiento de los mismos sin que se afecte la funcionalidad de los mismos.

Finalmente se plantea un DWH la cual consulta la información de cada esquema para que luego se puedan generar informes que permitan analizar mas en detalle la información.

**Vista lógica**

**Diagrama

Descripción generada automáticamente**

La anterior vista plantea la presentación lógica de los componentes iniciando por la presentación de la aplicación y mostrando el uso que esta realiza sobre las APIs del modelo y al mismo tiempo nos muestra la comunicación de las APIs con la base de datos.

**Vista Clases**

**Diagrama

Descripción generada automáticamente**

La propuesta de la anterior vista plantea la relación de las diferentes clases y sus atributos que pueden llegar a estar implementadas dentro de la arquitectura para así dar solución al requerimiento definido.

**Vista de desarrollo**

**Diagrama

Descripción generada automáticamente**

En la anterior vista se muestra relación de los diferentes componentes desarrollados iniciando desde los asociados al “**front end**” pasando por el “**back end**” y finalizando en la base de datos a utilizar. Adicional también nos muestra la iteración de los servicios en la parte lógica con los módulos externos mediante el uso de un balanceador de carga.

**Vista de procesos**

**Diagrama

Descripción generada automáticamente**

Con esta vista se establece mostrar la comunicación de los distintos procesos dentro de la solución planteada desde como el componente web debe pasar por un balanceados de servicios que luego este direccionara a la API correspondiente que a subes usan sus respectivas bases de datos.

**Vista física**

**Diagrama

Descripción generada automáticamente**

Finalmente se establece la relación de los servidores físicos necesarios y las responsabilidades que cada uno de estos tienen dentro de toda la arquitectura, en esto podemos identificar como se plantea tener un servidor de base de datos transaccional el cual almacenara toda la información gestionada desde los servicios, también se establece que esta base de datos transaccional se comunicara con una segunda base de datos la cual llamaremos DWH la cual será la encargada de gestionar todo el modelo de hechos y dimensiones necesarios para solventar la problemática asociada a los informes de información a analizar, finalmente tenemos un servidor virtualizado en el cual se puede desplegar cada uno de os servicios necesarios para soportar la aplicación WEB que hará el consumo de estos.

**Supuestos**

* Se espera contar con mínimo dos servidores que permitas la admiración de la base de datos y uno que permita el despliegue de servicios REST necesarios para exponer las funcionalidades del requerimiento.
* Los servicios deben ser desplegado con herramientas que permitan la orquestación de contenedores de forma que sea fácilmente escalables horizontalmente.
* Se debe contar con un visualizador de reportes que permita realizar las consultas necesarias sobre la bodega de datos para analizar mejor los datos.
* Los servicios no serán tan concurridos inicialmente lo cual permitirá que estos se puedan gestionar desde una arquitectura “on-premise”
* Cada empresa puede registrar la información en el sistema sin que se deba pasar por un registro de usuario.
* A cada vehículo se le puede asignar su o sus respectivos conductores sin que sea necesario tener un registro previo de usuario y acceso.

**Tecnologías sugeridas para resolver la solución (On premise y cloud).**

* Teniendo presente el tamaño de la aplicación, se sugiere que la implementación se realice inicialmente On-Primese una ves esta tenga un auge considerablemente elevado se puede entrar a valorar la migración de esta a una tecnología cloud, esto tenido presente la reducción en costos iniciales.
* Se sugiere implementar tecnologías contenerizadas como Docker y orquestador de contenedores como kubernetes, permitiendo esto trabajar con replicas de pod’s que permitan gestionar el auto escalado horizontal.
* El manejo de servidores virtualizados lo cual facilita el proceso de migración a una tecnología cloud.

**Planteamiento de la solución para los servicios y la aplicación**

A continuación, se plantea la forma correcta como se debe ejecutar la aplicación y los servicios de forma que se pueda ver su funcionamiento.

**Herramientas necesarias**

1. Angular (aplicacion web)
2. Eclipse (servicios rest)

**Estructura de los proyectos (servicios rest)**

**service-conductores**:servicio expuesto por el puerto 8003

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**service-vehiculos**: servicio expuesto por el puerto 8002

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**service-empresas**: servicio expuesto por el puerto 8001

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Todos los servicios cuentan con las siguientes capacidades:

|  |  |
| --- | --- |
| **Capacidad** | **Descripcion** |
| /listar | Lista toda la información de los diferentes servicios desplegados |
| /ver/{id} | Consulta un registro puntual a partir del ID dentro de todos los registros que se tienen en cada servicio |
| /crear | Crea un registro asociado al servicio se corresponda |
| /editar/{id} | Edita el registro a partir del ID para el servicio concreto. |
| /eliminar/{id} | Elimina el registro a partir del ID para el servicio concreto. |

Para que los servicios puedan funcionar correctamente se debe ejecutar cada servicio antes de correr aplicación, estos servicios se ejecutaran desde el IDE eclipse.

Cada servicio se compone de un script que contiene las inserciones necesarias para validar las capacidades de estos, por lo que al ejecutar la capacidad de listar podemos ver la información con la que podemos trabajar.

Si se desea validar el funcionamiento de cada uno de estos servicios una vez ejecutado el servicio se puede llamar la siguiente URL: <http://localhost:8001/listar> para listar la información de las empresas, de esta misma manera se puede hacer para los otros servicios cambiando únicamente los puertos correspondientes.