# Documento de Requisitos TRATEME

### Versión 1.0

### Fecha 21/08/2018

### Preparado para:

[Raúl Iván Mazo](" \l "ORG-0001" \o "Especialización en desarrollo de software)

### Preparado por:

Richard Ruiz

Jhon Medina

Jean Paul Galeano

Andrés López

Tras realizar un análisis detallado de la LP (Linea de Productos) salen a la luz diferentes condiciones que se deben de cumplir para que exista un correcto funcionamiento en cada aplicación dado el contexto y categoría del software que se desea implementar. En esta línea de negocio en particular se hace necesario que cada cliente cuente con unas características específicas que le permitirán hacer uso de las aplicaciones como lo es una conexión a internet estable y una dirección IP publica y estática la cual será agregada a una White-List previo al uso de la aplicación que desee utilizar, además de esto se hace necesario que este cliente a nivel interno pueda transar con el estilo arquitectónico REST, ya que las aplicaciones de esta LP son *aplicaciones orientadas a servicios* las cuales entregan un bajo acoplamiento y una serie de características que dado el contexto de negocio facilitaran realizar el acceso a una la plataforma remota donde las aplicaciones expondrán las funcionalidades requeridas en formato json que proveerá la comunicación a través de canales http, los cuales por medio de tokens jwt facilitaran la gestión de la seguridad brindando autenticación y validación de los usuarios de cada sistema.

Para lograr cumplir con este objetivo se usará tanto el estilo arquitectónico orientado a microservicios que facilita la construcción de servicios independientes lo que permitirá que cada uno sea desarrollados en el lenguaje de programación más apto para generar un mejor desempeño permitiendo que estos componentes hagan parte de un sistema completamente funcional y el estilo arquitectónico REST (REpresentational State Transfer) esta tecnología funciona con HTTP lo cual facilita la comunicación entre lenguajes lo que las hace mantenibles y escalables.

Teniendo en cuenta lo anterior se platea realizar un despliegue distribuido basado en una estrategia *Cliente-Servidor* ya que cada cliente tendrá su aplicación stand-alone las cuales consumirán los servicios almacenados en la nube dentro de un servidor web.

Para los aspectos técnicos de la aplicación es importante tener en cuenta una serie de factores que pueden afectar el correcto funcionamiento de las Apis, es por este que su método de construcciones debe de facilitar la contenerización, la administración de dicha contenerización y el escalamiento horizontal. Para lograr estas metas se a optado por desarrollar las aplicaciones en lenguajes orientaos a objetos tales como java, .Net y node js que permiten la extensión del funcionamiento de los microservicios, esto a su vez permitirá un alojamiento en múltiples sistemas operativos dado las ventajas entregadas por este tipo de aplicaciones.

Modelo de variabilidad



------------------------------------------------------ Texto de Ejemplo-----------------------------------------

Variabilidad en las aplicaciones.

1. Identificar la variabilidad en los tipos de aplicaciones

Es importante que las aplicaciones externas que hagan uso de las apis expuestas tengan haciendo uso de canales de intenet.

Las aplicaciones deben poder transar con el estilo arquitectónico REST

Deben de contar con una ip estatica que pueda suscribirse en una White list

Capacidad de generación de jwt

La mensajería de los servicios estará basada en el formato json

Los componentes de dominio estarán orientadas a familias de aplicaciones orientadas a servicios diseñadas para soportar comunicaciones entre componentes poco acoplados

1. Identificar la variabilidad en el estilo arquitectónico

Se usara el estilo arquitectónico orientado a Microservicios combinado con el estilo arquitectónico REST (REpresentational State Transfer)

La comunicación esta orientada a aplicaciones

Las apis estarán expuestas remotamente en Internet

1. Identificar la variabilidad en las estrategias de despliegue

La estrategia de despliegue es distribuida

Nota: Afecta la latencia de respuesta hacia la aplicación final, pero existen estrategias que ayudan a mejorar por ejemplo Arquitecturas orientadas a Cache.

Tolerable a fallos totales

Grandes volúmenes de transacciones , facilidad, flexibilidad, elasticidad

Pilas es despliegue de aplicación

Despliegue cliente – servidor. Capa Cliente Servidor web, Capa de Aplicación Servidor de Aplicaciones

El alojamiento de las apis estarán on cloud

1. Identificar la variabilidad en los aspectos técnicos

Poder transar con servicios rest

Tener canal de comunicación (Red, Puertos, Formato, etc ) habilitado para poder transar contra los componenter de domino

La tecnología de implementación de las apis deben tener las capacidades suficientes para cubrir

Capacidad de contenerizarse y escalamiento horizontal, despliegue automatico, y administrador de contenedores

Tales como .Net, J2EE, NODEJS

OS: Multi - sistema operativo

1. Construir el modelo de variabilidad de la arquitectura de referencia
2. Documentar la(s) arquitectura(s) de referencia