

**ESCUELA POLITÉCNICA NACIONAL**

**FACULTAD DE INGENIERÍA DE SISTEMAS**

**INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

**PERÍODO ACADÉMICO:** 2016 A

**ASIGNATURA:** COMPUTACION DISTRIBUIDA

**PROFESOR: Ing**. Carrera Izurieta Ivan Marcelo

**INTEGRANTES:** Mendoza Kevin

Vargas Marlon

Vivanco Katherine

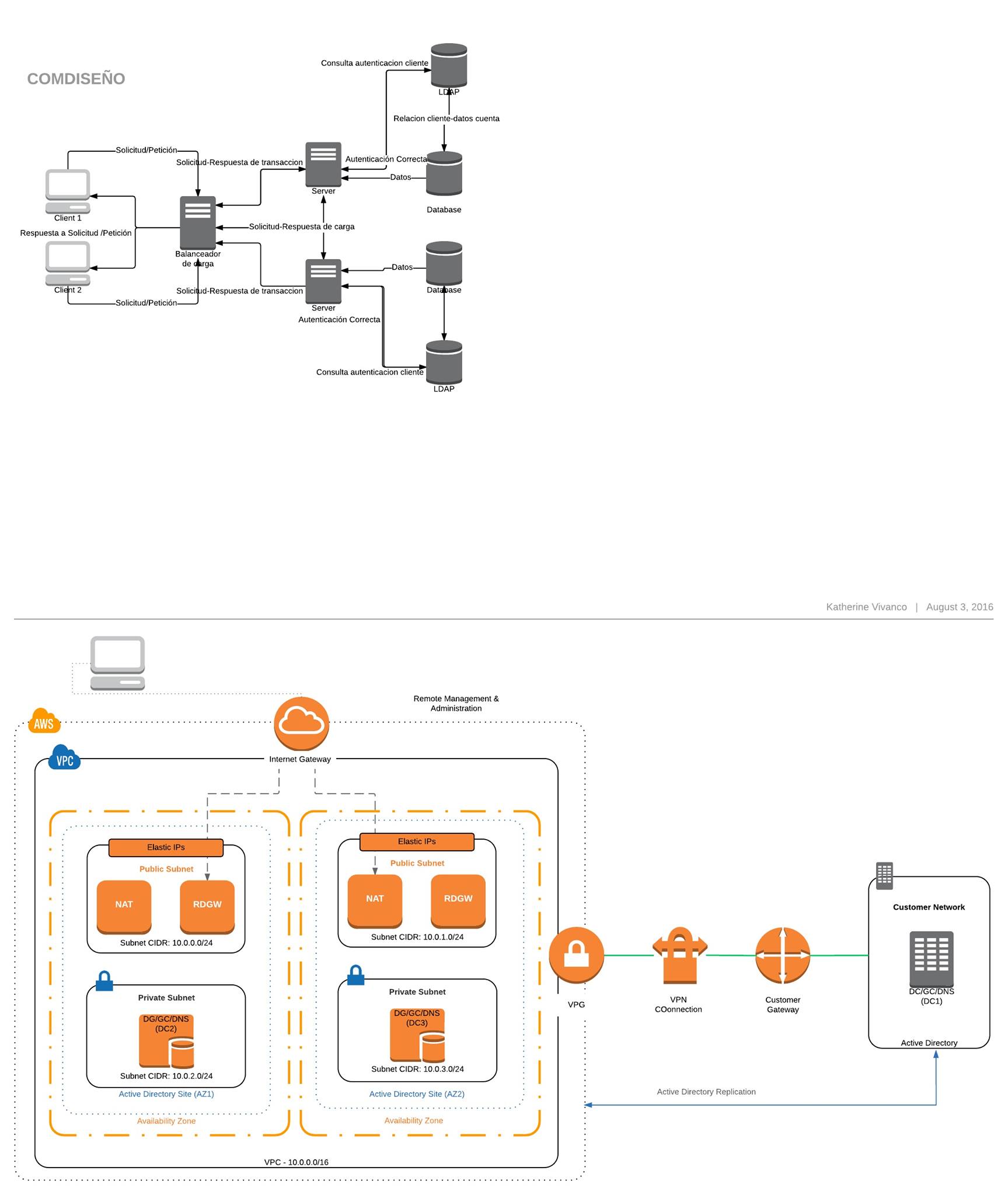
**TEMA:** PROYECTO

1. **Concepto y Diseño**

Se tiene la necesidad de implementar un sistema de transacciones bancarias, el cual opere sobre una arquitectura distribuida de varios servidores. Éstos deben proveer a los clientes del banco los servicios necesarios para que puedan realizarse transacciones como: consulta, depósito y retiro. Los datos relativos a las transacciones se deben almacenar en una base de datos distribuida replicada en cada servidor del sistema.

Los servidores del banco que realizaran las transacciones recopilara los datos de procesamiento, los cuales los almacenara en un archivo en tiempo real, y estos se enviaran al balanceador cuando se los solicite.

El balanceador de carga se encargara de recibir las peticiones de todos los clientes del banco, después este enviara una petición a todos los servidores del banco que solicitara su información de carga, dependiendo de estos datos se va a determinar el servidor que se va a usar y enviara la solicitud que viene del cliente.

****

1. **Algoritmo**

* **Pseudocódigo:**
  + Programa Cliente:

*Proceso Cliente*

*Crear socket*

*Abrir canal de comunicación*

*Conectar con servidor Balanceador*

*Hacer:*

*Enviar datos*

*Recibir datos*

*Hasta que Cerrar canal de comunicación*

*Fin Proceso Cliente*

* + Programa Servidor balanceador de carga:

*Proceso Servidor Balanceador*

*Crear socket*

*Abrir canal de comunicación*

*Escuchar Conexión Cliente*

*Hacer*

*Recibir solicitud cliente*

*Hacer*

*Enviar solicitud datos carga*

*Recibir datos*

*Hasta que se envíe a todos los servidores*

*Determinar servidor con menor carga*

*Conectar con servidor*

*Repetir*

*Enviar solicitud cliente*

*Recibir datos*

*Hasta que cerrar conexión*

*Enviar datos a cliente*

*Hasta que cerrar canal de comunicación*

*FinProceso*

* + Programa Servidor:

*Proceso Servidor*

*Crear socket*

*Abrir canal de comunicación*

*Escuchar Conexión Balanceador*

*Si petición = carga Entonces*

*enviar\_datos\_carga*

*Sino*

*Hacer*

*Recibir solicitudes balanceador*

*Procesar solicitud balanceador enviar datos*

*Hasta que cerrar conexión*

*Fin Si*

*Fin Proceso Servidor*

1. **Selección de Herramientas**

Sistema operativo: Windows 10-windows server 2012

ID: net beans

BD: postgres

**2. El resumen de los resultados de las Pruebas realizadas**

* Como resultados de las pruebas resultadas tenemos:
* A alta concurrencia de clientes al querer acceder al servidor, el alanceador de carga funcionó perfectamente al igual que la implementación de threads, y nunca se cayó el sistema.
* Las transacciones bancarias, funcionaron a la perfección, y al querer estresar al servidor con varias transacciones bancarias seguidas, este no se colgó.
* No se implementación el servidor ldap, por lo cual no se cumplió con el objetivo de que este guarde la información de los usuarios (psw, user), en fin, estos datos fueron ingresados en la bd y funcionaron perfectamente.

1. **Las Conclusiones y Recomendaciones obtenidas en el proceso de implementación del sistema**

**Conclusiones**

* Las herramientas elegidas en el primer informe, fueron correctas, no las tuvimos que cambiar.
* La implementación de ldap hubiese hecho de nuestro sistema algo más seguro, ya que las contraseñas y usuarios se hubiesen guardado en ello.
* El sistema cumple con su objetivo, no colgarse y realizar transacciones.

**Recomendaciones**

* Como buena práctica, se recomienda no guardar una contraseña en la base de datos y q esta sea completamente visible, sino que esta se guarde en un formato seguro.
* Se recomienda siempre tener un balanceador de carga, para que el sistema nunca se caiga.