Documento de descripción arquitectónica

Cliente Servidor

Lizzette Betancourt, Mishele Loján C

Versión 1.0

Mayo 2015

**Historial de revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 04/05/2015 | 1.0 | Versión inicial del documento de descripción arquitectónica Cliente-Servidor. | Lizzette Betancourt,  Mishele Loján |

Tabla de contenido

[**1.** **Introducción** 4](#_Toc421116680)

[1.1 Propósito 4](#_Toc421116681)

[1.2 Alcance 4](#_Toc421116682)

[1.3 Definiciones, siglas y abreviaturas 4](#_Toc421116683)

[1.4 Referencias 4](#_Toc421116684)

[1.4 Resumen 5](#_Toc421116685)

[**2.** **Representación arquitectónica** 5](#_Toc421116686)

[**Vista Lógica** 5](#_Toc421116687)

[**Vista de Procesos** 5](#_Toc421116688)

[**Vista de desarrollo** 5](#_Toc421116689)

[**Vista Física** 6](#_Toc421116690)

[**Vista de Escenarios (+1)** 6](#_Toc421116691)

[**3. Objetivos y limitaciones de la arquitectura** 6](#_Toc421116692)

[3.1 Plataforma técnica 6](#_Toc421116693)

[3.2 Transacciones 6](#_Toc421116694)

[3.3 Seguridad 6](#_Toc421116695)

[3.4 Disponibilidad 7](#_Toc421116696)

[3.5 Rendimiento 7](#_Toc421116697)

[**4. Vista de escenarios** 7](#_Toc421116698)

[4.1 Actores 8](#_Toc421116699)

[Servidor 8](#_Toc421116700)

[Cliente 8](#_Toc421116701)

[**5. Vista Lógica** 8](#_Toc421116702)

[**6. Vista de proceso** 8](#_Toc421116703)

[**7. Vista de desarrollo** 9](#_Toc421116704)

[**8. Vista de Física** 10](#_Toc421116705)

[**9. Tamaño y rendimiento** 10](#_Toc421116706)

[**10. Calidad** 10](#_Toc421116707)

# **Introducción**

El estilo arquitectónico Cliente-Servidor está compuesto por un cliente, un servidor y una red de conexión, este estilo describe cual es la relación entre el cliente y uno o varios servidores, el cliente puede efectuar varias solicitudes y el servidor podrá responder a cada una de ellas.

El presente documento nos proporciona el desarrollo de la arquitectura, que permite mostrar cual es la estructura de la aplicación en la que se va a efectuar el estilo arquitectónico cliente-servidor, para que esto se haga posible se utilizará el modelo de vistas 4+1 que describe la arquitectura del software usando vistas concurrentes que son: vista lógica, procesos, desarrollo y escenarios. El estilo arquitectónico del sistema de chats se basas en la arquitectura cliente servidor

## Propósito

El documento de arquitectura del software nos brinda una versión completa de la arquitectura cliente- servidor distribuido utilizando el modelo de vistas de Kruchten (4+1). Además permitirá conocer los elementos que van distribuidos en cada una de las vistas, ver y entender cómo se relacionan estos elementos de la arquitectura antes mencionada.

## Alcance

El alcance del presente documento de arquitectura del software es representar la arquitectura cliente servidor a través de modelo de vistas de Kruchten, para representar cada una de las vistas se utiliza el Lenguaje de modelado unificado (UML) para entender su funcionalidad.

## Definiciones, siglas y abreviaturas

* **MySQL** – sistema de gestión de base de datos relacional (RDBMS), para almacenar información de los usuarios.
* **SAD** - documento de arquitectura de Software
* **RUP** -Proceso racional unificado
* **UML** – Lenguaje unificado de modelado
* **Cliente** – es quien desea conectar con el servidor, para enviar y recibir respuestas del mismo.
* **Servidor –** es el encargado de responder cada una de las solicitudes del cliente.

## 1.4 Referencias

Kruchten, Philippe (s.f) Planos Arquitectónicos: El Modelo de “4+1” Vistas de la Arquitectura del Software∗. Recuperado el 15 de Mayo de 2015. Disponible en: <http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf>

Meier,J.(2009). Microsoft Application Architecture Guide. Disponible en: <https://msdn.microsoft.com/en-us/library/ff650706.aspx>

## Resumen

Con el fin de documentar plenamente todos los aspectos de la arquitectura, el Documento de Arquitectura de Software contiene las siguientes subsecciones.

Este documento de arquitectura de software está estructurado de la siguiente forma:

**Sección 2**: describe el uso de cada vista del modelo 4+1.

**Sección 3**: describe las limitaciones arquitectónicas de la aplicación.

**Sección 4:** Diagramas UML para representar la vista de casos de uso

**Sección 5:** Diagramas UML para representar la vista Lógica.

**Sección 6:** Diagramas UML para representar la vista de Procesos

**Sección 7:** Diagramas UML para representar la vista de desarrollo

**Sección 8:** Diagramas UML para representar la vista física.

**Sección 9:** Contendrá información acerca del rendimiento de la aplicación.

**Sección 10:** Describe algunas cuestiones y observaciones sobre la aplicación

# **Representación arquitectónica**

El modelo de arquitectura de software está representado por las siguientes vistas:

## **Vista Lógica**

**Audiencia:** Diseñadores.

**Área**: En esta vista representa la funcionalidad que el sistema proporciona a los usuarios finales. Es decir representa lo que el sistema va hacer y cada una de las funciones que ofrece.

**Artefactos relacionados:** Diagrama de clases, diagrama paquetes y diagrama de secuencia

## **Vista de Procesos**

**Audiencia**: Integrador de sistemas.

**Área:** Muestra los procesos del sistema así como la comunicación de estos y se centra en el comportamiento en tiempo de ejecución del mismo.

**Artefactos relacionados:** Diagrama de actividades.

## **Vista de desarrollo**

**Audiencia:** Programadores

**Área:** mostrar cómo está dividido el sistema software en componentes y las dependencias que hay entre esos componentes.

**Artefactos relacionados:** Diagrama de componentes y diagrama de paquetes.

## **Vista Física**

**Audiencia:** Ingeniero en Sistemas.

**Área:** Muestra todos los componentes físicos del sistema y las conexiones entre estos componentes que conforman las soluciones que incluyen en los sistemas.

**Artefactos relacionados:** Diagrama de despliegue

## **Vista de Escenarios (+1)**

**Audiencia:** Interesados y usuarios finales

**Área:** representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas, esto quiere decir que desde un caso de uso podemos ver cómo se van ligando las otras 4 vistas, con lo que tendremos una trazabilidad de componentes, clases, equipos, paquetes, etc., para realizar cada caso de uso**.**

**Artefactos relacionados:** Diagramas de casos de uso.

# **3. Objetivos y limitaciones de la arquitectura**

Esta sección describe aquellos requisitos y objetivos de software que se necesitan para desarrollar la arquitectura.

## 3.1 Plataforma técnica

La aplicación de la arquitectura cliente- servidor será desarrollada en un IDE denominado Netbeans 7.0.

## 3.2 Transacciones

La aplicación de la arquitectura cliente- servidor es transaccional ya que aprovecha las capacidades de la plataforma técnica.

## 3.3 Seguridad

La aplicación de la arquitectura cliente- servidor debe ser segura al momento de que el cliente interactúe con los datos. Además se tendrá la seguridad necesaria como por ejemplo:

**Confiable:** Para determinar que sea una conexión segura del cliente al servidor.

**Integridad de datos:** para la transmisión correcta de datos entre el cliente y el servidor.

**Auditoría:** para registrar cada acción que se realices.

## 3.4 Disponibilidad

La disponibilidad es un requisito clave, ya que la aplicación de la arquitectura cliente- servidor va a estar siempre activa. La disponibilidad dirigida será 24/7.

## 3.5 Rendimiento

La aplicación de la arquitectura cliente- servidor dará respuesta para un número indeterminado de clientes que lo soliciten.

# **4. Vista de escenarios**

La vista de escenario permite representar la funcionalidad que tiene el sistema mediante el diagrama de casos de uso.

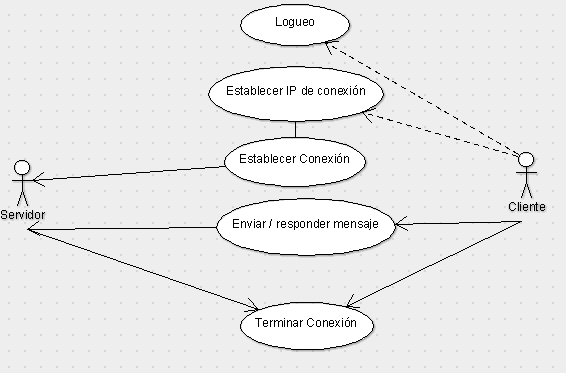


Fig 1. Diagrama de casos de uso

* **Logueo:** el cliente deberá loguearse o iniciar sesión para poder usar el aplicativo.
* **Establecer IP de conexión:** el cliente deberá ingresar la IP del servidor junto con el puerto que se va a realizar la conexión.
* **Establecer conexión:** El cliente deberá conectarse al programa servidor, en el cual se indicará si la conexión fue exitosa o no.
* **Enviar /responder mensaje:** Pueden existir múltiples clientes, los cuales puedes enviar mensajes ilimitadamente y el servidor así mismo enviará respuesta.
* **Terminar Conexión:** Se cierra la conexión por cualquiera de los actores.

## 4.1 Actores

Los actores en el sistema de chat son:

### Servidor

El servidor permitirá que sobre él se realicen múltiples conexiones, los cuales podrán recibir varias peticiones de cada cliente y resolver cada una de ellas.

### Cliente

Es quien solicita una conexión con el servidor para poder enviar solicitudes y recibir

# **5. Vista Lógica**

A la vista lógica se la representa mediante el diagrama de clases en el cual se puede observar cuales son las relaciones entre las clases, los métodos y operaciones que se involucran en el sistema

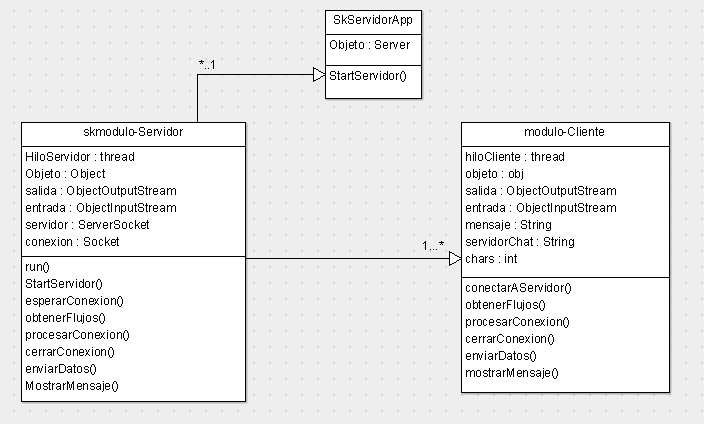


Fig 2. Diagrama de clases

# **6. Vista de proceso**

Para representar la vista de Procesos se utilizará del diagrama de actividad, en donde se mostrará el comportamiento dinámico del sistema y los flujos entre procesos del negocio.

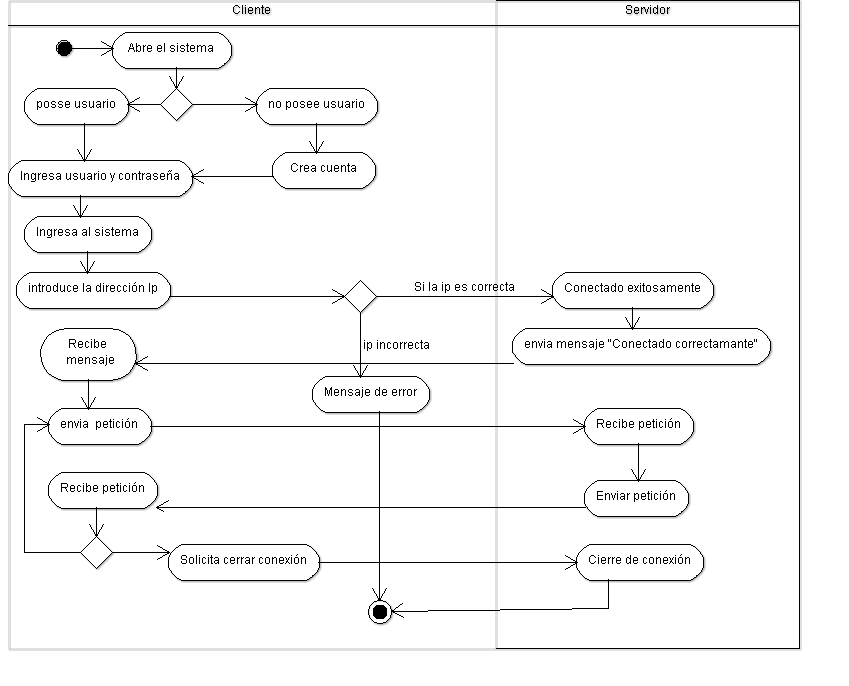


Fig 3. Diagrama de actividad

# **7. Vista de desarrollo**

Esta vista presenta la organización estática del sistema. En este caso presentará la distribución de la arquitectura cliente servidor en este aplicativo.

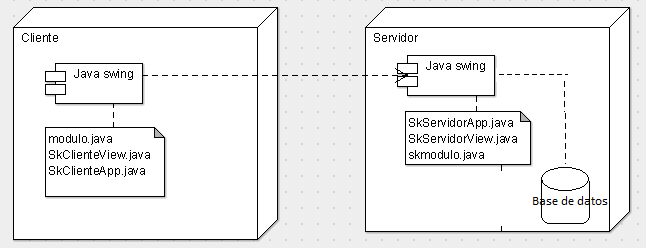


Fig 4. Diagrama de componentes

# **8. Vista de Física**

El aplicativo de interacción entre un cliente- servidor se encuentra compuesta por varias clases .java que permite el funcionamiento de dicho aplicativo. Para montar dicho aplicativo se necesita una máquina que tenga pre-instalado el ide de programación Netbeans 7.3 no importa el Sistema operativo que posea. El aplicativo de esta arquitectura puede funcionar en una sola maquina o en dos.

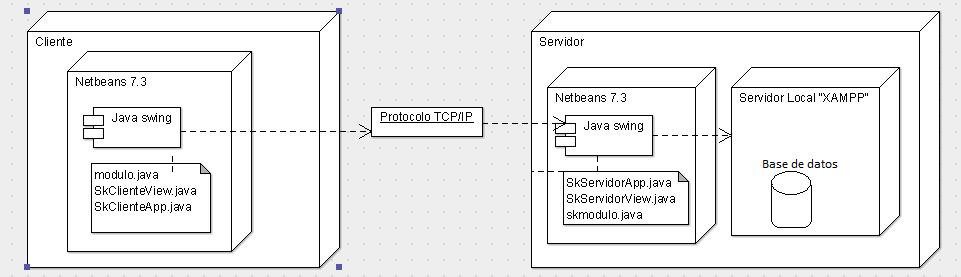


Fig 5. Diagrama de despliegue

# **9. Tamaño y rendimiento**

Volúmenes:

• Estimación de las conexiones simultáneas: 5 conexiones

• Envío de mensajes por cliente: dependerá de la velocidad de rede de cada usuario

Rendimiento:

• Tiempo que se ocupe para procesar el envío y recepción de los datos

# **10. Calidad**

Escalabilidad:

* La reacción del sistema cuando el usuario exige aumento

Fiabilidad, Disponibilidad:

* Mecanismo de conmutación por error transparente , tiempo medio entre fallos

Portabilidad:

* Capacidad para ser reutilizado en otro entorno

Seguridad:

* Autenticación y autorización mecanismos