

**DISEÑO E IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA
TECNOLOGICA CON FINES EDUCATIVOS SOBRE SISTEMAS
DISTRIBUIDOS UTILIZANDO RASPBERRY PI.**

Tutor
Carlos Daniel Barroeta Mata

Revisor
David Hernández Valencia

Tesista
Héctor Félix Sam Colina

PLANTEAMIENTO DEL PROBLEMA

Desde el año 2008 la Escuela de Ingeniería Informática de la Universidad Católica Andrés Bello dicta la cátedra de Sistemas Distribuidos, anteriormente llamada Sistemas de Operación II, en ella, se profundizan temas en relación a diferentes nodos conectados entre sí para proporcionar una funcionalidad específica. Es una materia teórico – práctica que posee proyecto y se necesita obtener un mínimo de 4.75 puntos en cada parte para ser aprobada.

Muchas materias de Ingeniería Informática cuentan con prácticas para reforzar los conocimientos adquiridos en la teoría, estas son realizadas dentro de un laboratorio, donde los estudiantes forman equipos de trabajo y logran realizar una serie de actividades que permiten asociar los conocimientos de la materia de forma directa. Normalmente estas prácticas poseen una duración de dos horas y en algunos casos se entregan informes que validan que los estudiantes concluyeron con los objetivos de la clase.

Hoy en día la materia de sistemas distribuidos se dicta con 4 horas semanales, dónde los estudiantes aprenden todo lo relacionado con procesos e hilos, comunicación, sincronización, replicación, sistemas de archivos, servicios de nombre, objetos distribuidos y servicios web. Los estudiantes atienden a la explicación de los profesores para un tema particular. Algunos temas pueden tardar más tiempo que otros debido a su complejidad. Por lo que a veces no pueden ser captados por los estudiantes de forma continua, es decir, en una misma semana de clases. Los profesores emiten los proyectos de la materia con la finalidad de que los estudiantes puedan aplicar los conocimientos teóricos adquiridos en clases para resolver el problema planteado en el enunciado.

Para realizar el proyecto, los alumnos se organizan en equipos, dos o tres personas como máximo. Dependiendo de las exigencias del proyecto, ellos deben buscar información acerca de los temas que se relacionan con el problema a resolver. Los apuntes de cada estudiante y las láminas por el profesor no siempre son suficientes para reforzar los conceptos, el uso del internet ha permitido que muchos estudiantes lo utilicen como su principal fuente de información para resolver dudas en relación a un concepto o tema que su significado no se encuentre muy claro.

Se propone implementar una infraestructura tecnológica con fines educativos que permita reforzar los conocimientos en relación a los sistemas distribuidos mediante aplicaciones orientadas a los tópicos de la cátedra. Cada aplicación será desplegadas en pequeñas computadoras llamadas *Raspberry Pi* para simular los nodos de un sistemas distribuido. Las aplicaciones contendrán parámetros de configuración que serán almacenados en una base de datos al igual que la información teórica asociada al tópico. Los usuarios controlarán los nodos a través de un módulo de ciclo de vida encargado de enviar instrucciones utilizando scripts de sistema operativo que se conecten a los nodos mediante el protocolo Secure Shell (SSH).

Cada nodo contendrá un agente de configuración que permite recolectar información sobre el estado del *Raspberry Pi* y las aplicaciones en ejecución, posteriormente se envía la información a un módulo de monitoreo que le permite al usuario visualizar los mensajes intercambiados entre los nodos. Los mensajes son enviados mediante una librería de registro de mensajes. La comunicación dentro de la infraestructura se realizará mediante sockets. Por último, los usuarios pueden crear sus propias aplicaciones con sus respectivos parámetros de configuración e información teórica y almacenarlas en la base de datos para utilizarlas consecutivamente dentro de la infraestructura tecnológica.

OBJETIVOS

General

Diseñar e implementar una infraestructura tecnológica con fines educativos que permita la consolidación de conocimientos sobre la cátedra de Sistemas Distribuidos mediante la simulación de nodos utilizando Raspberry Pi para estudiantes de la Escuela de Ingeniería Informática de la Universidad Católica Andrés Bello.

Específicos:

- Desarrollar un conjunto de aplicaciones educativas que permitan la simulación de nodos de un sistema distribuido en función a los tópicos de la cátedra.
- Desarrollar un agente de configuración que permita recolectar información del estado actual de los nodos que forman parte del sistema distribuido.
- Desarrollar un servicio de recepción capaz de procesar la información proveniente de los agentes de configuración instalados en los nodos.
- Desarrollar una librería que encapsule un componente para el registro de mensajes de entrada y/o salida que será incorporada en los componentes de la infraestructura y en las aplicaciones de sistemas distribuidos.
- Diseñar e implementar una base de datos que permita el almacenamiento de los parámetros de configuración y la información teórica correspondiente a las aplicaciones que forman parte de los tópicos del contenido programático de la cátedra de Sistemas Distribuidos.
- Crear scripts de sistema operativo que serán desplegados en los nodos para controlar aspectos como: parametrización, ejecución y detención de las aplicaciones de sistemas distribuidos.
- Desarrollar un módulo de administración que permita controlar el ciclo de vida de los nodos que forman parte de la infraestructura tecnológica.

- Implementar un módulo de monitoreo que permita visualizar el estado de cada nodo así como los mensajes capturados por la librería de gestión de mensajes.
- Desarrollar un módulo de gestión que permita la creación, modificación y eliminación de tópicos y preguntas así como la incorporación de aplicaciones pertenecientes a la cátedra de Sistemas Distribuidos.
- Integrar los módulos de ciclo de vida, monitoreo y gestión a través de una aplicación web que permita la interacción del usuario con la infraestructura tecnológica.
- Elaborar la documentación de los componentes desarrollados pertenecientes a la infraestructura tecnológica

ALCANCE

Las aplicaciones educativas sobre sistemas distribuidos se basarán en el contenido programático de la cátedra Sistemas Distribuidos de la Escuela de Ingeniería Informática de la UCAB, cada aplicación estará asociada a un tópico y permiten consolidar los conocimientos de acuerdo a escenarios definidos. Un escenario es un posible resultado que se puede obtener por un evento generado por el usuario. La siguiente tabla indica los tópicos que fueron seleccionados para desarrollarse como aplicaciones:

Tema	Aplicaciones
Introducción a los Sistemas Distribuidos	Características de los Sistemas Distribuidos Desafíos de los Sistemas Distribuidos Arquitecturas: Cliente / Servidor Punto a Punto (P2P)
Comunicación en Sistemas Distribuidos	Sockets RMI Comunicación en grupo
Sincronización en ambientes distribuidos	Relojes lógicos: Algoritmo de Lamport Relojes físicos: Algoritmo de Cristian Algoritmo de Berkeley Algoritmo con promedio (distribuido) Exclusión mutua: Algoritmo Centralizado Algoritmo Distribuido (Ricart y Agrawala) Algoritmos de selección: Grandulón Anillo
Replicación	Tipos de Fallas Fallas Bizantinas
Sistemas de archivos distribuidos	Arquitectura Cliente Servidor: Modelo de acceso remoto Modelo de carga y descarga

	Sistemas de archivos basados en Clúster
Servicios de nombre	Domain Name System (DNS) Lightweight Directory Access Protocol (LDAP)
Objetos Distribuidos	Remote Method Invocation (RMI) Enterprise JavaBeans (EJB)
Servicios Web	Simple Object Access Protocol (SOAP) (Servidor / Cliente) Representational State Transfer (REST) (Servidor / Cliente)

Tabla 1: Aplicaciones Sistemas Distribuidos, Elaborado por: Héctor Sam

Las aplicaciones se desarrollarán utilizando Java como lenguaje de programación, contarán con patrones de diseño para garantizar las buenas prácticas del desarrollo de software, pruebas unitarias que validen que el comportamiento de cada método es correcto y estándares de codificación para permitir que cualquier desarrollador pueda comprender el código sin problemas. Adicionalmente, poseen parámetros de configuración que son obtenidos desde una base de datos.

Los nodos se simularán utilizando *Raspberries Pi*, consisten en pequeñas computadoras que ejecutan el sistema operativo Linux y poseen la máquina virtual de Java. Se encontrarán conectadas a una red local para permitir la comunicación entre ellas y los elementos que conforman la infraestructura tecnológica.

La comunicación entre el usuario y las aplicaciones de sistemas distribuidos se realizará a través de un servidor central, quien será el encargado de enviar y recibir datos por parte de los nodos. Los ejecutables de cada aplicación se encontrarán alojados en un repositorio local dentro del servidor, permitiendo enviar las aplicaciones a cada nodo con sus respectivos parámetros en el momento que el usuario lo indique desde el módulo de ciclo de vida.

El agente de configuración consiste en una aplicación java que se encontrará alojada en todos los nodos *Raspberries*. Al encender el nodo, el agente se iniciará automáticamente y recopilará información del sistema operativo como por ejemplo: configuración de red, procesos en ejecución, memoria disponible, entre otros. Adicionalmente, indicará la aplicación activa que se estará ejecutando (Nombre de la aplicación, estado actual, número de nodo, información propia de la aplicación). El agente tendrá configurado la dirección de red donde se encuentra en ejecución el servidor central para el envío de la información recopilada.

El servicio de recepción será el encargado de procesar la información enviada por los agentes de configuración y por las aplicaciones de los nodos para posteriormente ser visualizada por los usuarios. El servicio será ejecutado en el servidor central, este recibe los mensajes enviados a través de la librería de recepción de mensajes y los redirige al módulo de monitoreo para que el usuario pueda observar la información correspondiente.

La librería para el registro de los mensajes de entrada y salida se desarrollará en Java. Tendrá atributos configurables dónde se indicará la dirección ip del remitente y de los destinatarios, el mensaje a enviar, fecha y hora del mensaje, entre otros. Se encontrará en todos los componentes de la infraestructura.

Se creará una base de datos que almacene la información de los tópicos como por ejemplo: definiciones, puntos a tratar, imágenes, preguntas, entre otros. Permitirá almacenar los parámetros de configuración necesarios para ejecutar cada aplicación en los nodos (dirección ip, puertos, parámetros propios

de las aplicaciones). Guardará las rutas del repositorio local (path) donde se encontrarán ubicados los ejecutables de las aplicaciones.

Los scripts de sistema operativo (Shell scripts) se encontrarán almacenados dentro del servidor central, permiten que el usuario pueda controlar las aplicaciones en cualquier momento desde el módulo de ciclo de vida. Cada script posee instrucciones de la acción a realizar (ejecutar la aplicación, detener un nodo particular, detener todos los nodos, entre otros). Mediante el protocolo ssh se accede a los nodos para realizar la acción deseada.

El módulo de administración de ciclo de vida le permite al usuario desplegar y controlar las aplicaciones sobre sistemas distribuidos en los nodos *Raspberry Pi*. Mediante un conjunto de botones se mostrarán los eventos disponibles para la aplicación actual (ejecutar, detener un nodo particular, detener todos los nodos, entre otros). El usuario selecciona un evento específico y el módulo envía un mensaje a través de la librería al servidor central para localizar el script correspondiente para su ejecución.

El módulo de monitoreo permite que el usuario pueda observar los eventos que ocurran en las aplicaciones distribuidas, los mensajes enviados entre cada nodo y su estado actual. Mediante la librería de mensajes se recibirá la información enviada por los nodos y por los agentes de configuración al servicio de recepción, este último reenvía la información al módulo de monitoreo separándola en dos tipos: información de los nodos (mensajes entre ellos), información del ambiente distribuido (cantidad de nodos, estado, aplicación activa, entre otros).

El módulo de gestión permitirá que se puedan agregar nuevos tópicos a la infraestructura con su debida aplicación, existirán formularios que permitan

introducir la información del tópico que se está creando, adicionalmente se tiene la opción de subir los ejecutables de las nuevas aplicaciones y un formulario de parámetros de configuración que posteriormente será guardado en la base de datos. Se requiere un registro previo del usuario para poder utilizar el módulo de gestión. Cuando un usuario almacene una aplicación dentro del repositorio local, tendrá la opción de colocar la cantidad de parámetros necesarios a través de un formulario dinámico. Donde cada uno tendrá la siguiente sintaxis: Parámetro1 = Valor1, Parámetro2 = Valor2.

El módulo de gestión permite generar preguntas acerca de un tópico, cada pregunta se encuentra relacionada con los conceptos teóricos o las aplicaciones educativas. Poseen el esquema de selección simple, donde se muestran varias opciones y solo una es la respuesta correcta. Existirán un conjunto de preguntas por cada tópico almacenadas en la base de datos y se podrán generar aleatoriamente, permitiendo que se puedan crear diferentes modelos de pruebas para un mismo tópico. Se creará un archivo en formato PDF por cada modelo para ser descargado por los usuarios de la infraestructura. El módulo permite realizar las operaciones de agregar, modificar o eliminar preguntas a la base de datos.

Los módulos desarrollados se integrarán mediante una aplicación web compuesta por una interfaz gráfica amigable, intuitiva y de fácil uso para todos los usuarios. Al acceder a la aplicación web, el usuario observa una lista con todos los temas de la materia y los tópicos asociados. Cuando se selecciona un tópico se muestra una descripción sobre el objetivo del mismo, las aplicaciones disponibles y la opción de generar preguntas. El usuario al hacer clic sobre la aplicación del tópico es redirigido a otra página donde se visualiza una pequeña introducción de la aplicación, las instrucciones de despliegue, los posibles

escenarios que pueden surgir al generarse un evento, los módulos de administración de ciclo de vida y de monitoreo.

Las aplicaciones sobre sistemas distribuidos tendrán asociadas unos escenarios de acuerdo a los eventos generados por el usuario. Cuando se finalice la ejecución de la aplicación, el usuario compara el resultado obtenido con los escenarios predefinidos para comprobar que se haya cumplido con el objetivo del tópico.

La documentación se encuentra relacionada con cada componente desarrollado, se realizará un manual de uso de la infraestructura que puedan orientar al usuario al utilizar los *Raspberries Pi*, de igual forma se crearán documentos de creación que indiquen como se realizó cada uno de los componentes, incluyendo diagramas de diseño (historias de usuario, entidad relación, de clases, entre otros). Por último, se creará una guía de configuración que permita instalar y configurar toda la infraestructura para su posterior funcionamiento, indicando los requerimientos de hardware, versiones del software, pasos para configurar el ambiente y como asociar aplicaciones realizadas por otros usuarios.

La siguiente ilustración muestra la arquitectura de la infraestructura planteada, donde se obtienen cuatro componentes: Una aplicación web a la que accede el usuario de la infraestructura, un servidor central, un servidor de base de datos y cuatro *Raspberries Pi* que funcionarán como nodos del sistema distribuido:

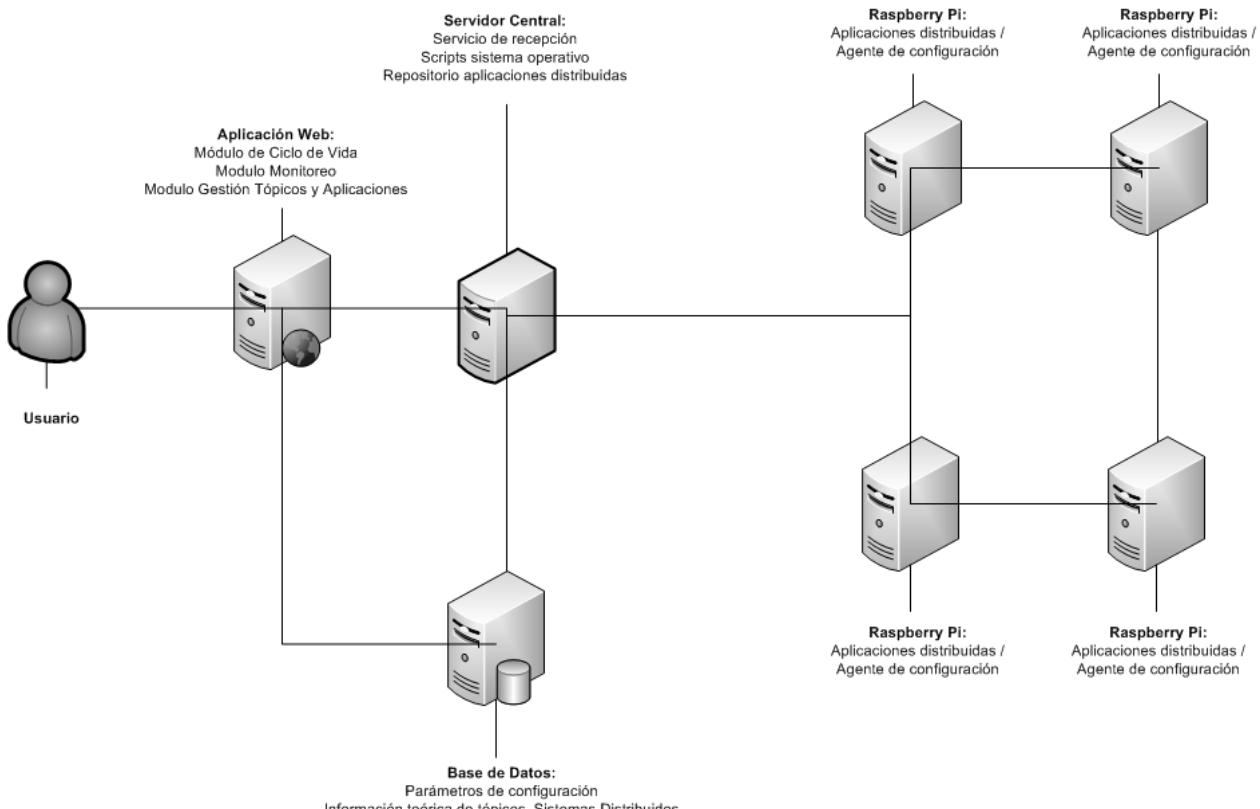


Ilustración 1: Arquitectura Infraestructura. Elaborado por: Héctor Sam

LIMITACIONES

- Los tópicos a desarrollar estarán basados únicamente en los temas del programa de sistemas distribuidos para estudiantes de ingeniería informática de la UCAB.
- Los scripts a realizar sólo podrán ser ejecutados bajo sistemas operativos Unix / Linux.
- Existirán un máximo de cuatro *Raspberries Pi* que funcionen como nodos.
- Las aplicaciones a desarrollar sobre sistemas distribuidos no poseen interfaz gráfica, por lo que todas las acciones que ocurran dentro de ellas solo podrán ser visualizadas por salidas enviadas por la librería desarrollada. En caso que se requiera enviar una información a un nodo particular el usuario lo podrá indicar desde el módulo de ciclo de vida.
- Las aplicaciones a desarrollar sobre sistemas distribuidos se enfocarán en resolver el objetivo del tópico, no poseen un alto grado de complejidad en términos de programación (líneas de código, cantidad de clases utilizadas, entre otros) más que el necesario para mantener las buenas prácticas del desarrollo de software.
- Existirá una base de diez preguntas relacionadas con cada tópico de la cátedra sistema distribuido en función a las aplicaciones a desarrollar, dejando a criterio del profesor la creación de nuevas preguntas.

JUSTIFICACIÓN

El motivo de realizar este trabajo especial de grado se debe a que muchos de los elementos descritos en la materia (arquitecturas, servicios web, sincronización, servicio de nombre, entre otros) son muy utilizados fuera del ámbito académico, conceptos que todo ingeniero informático debe manejar independientemente de la especialización que desea aspirar. En ingeniería se ha demostrado que las prácticas le permiten a los alumnos visualizar los conceptos que en teoría pueden parecer complicados o muy extensos. El hecho de que el estudiante pueda observar que está ocurriendo con los nodos cuando ocurra algún evento ayuda a comprender los conceptos teóricos de la cátedra.

Utilizar *Raspberries Pi* como nodos permite que estudiantes de ingeniería informática que no posean los recursos necesarios para comprar una computadora con al menos cuatro núcleos (tres nodos + servidor web y base de datos) puedan aprovecharlas debido a sus bajos precios, sólo 35 \$. Debido a que los *Raspberries Pi* utilizan Linux como sistema operativo, se pueden desarrollar no sólo aplicaciones Java como en este trabajo, sino también bajos otros lenguajes muy utilizados en la actualidad.

A través de la infraestructura planteada se obtiene un laboratorio portable, debido a que se requiere poco espacio físico para desplegar cada componente: *Raspberries Pi* y sus elementos (cables de red, de corriente, memoria sd), un Switch o Router para establecer la conexión entre cada nodo y una computadora central que funcione como controladora de toda la infraestructura. Los estudiantes puedan comprender los conceptos relacionados con los sistemas distribuidos mediante los resultados obtenidos por las aplicaciones desplegadas

y comparándolos con los escenarios planteados para cada aplicación, logrando cumplir con el objetivo del tópico.

Se permite que los usuarios de la infraestructura (Profesores o estudiantes) puedan generar sus propias aplicaciones, pudiéndole agregar mayor complejidad o enfocarlas desde otro punto de vista, almacenarlas en la infraestructura con sus parámetros de ejecución, instrucciones de despliegue, posibles escenarios y archivos ejecutables. La infraestructura permite generar preguntas de tipo examen que puedan servir como una guía de estudio para los estudiantes. Por otro parte, se permite que se puedan crear, modificar o eliminar tópicos a la infraestructura en caso de que el programa para Sistemas Distribuidos de la Escuela de Ingeniería Informática en la UCAB sea modificado.

MARCO REFERENCIAL

Sistemas Distribuidos:

Los sistemas distribuidos se pueden definir como “Aquel sistema en el que los componentes localizados en computadores conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes” [1]. La principal razón para la construcción de sistemas distribuidos es compartir recursos, un recurso puede estar dirigido a hardware: Discos duros, impresoras, entre otros o a software pudiendo incluir elementos como base de datos, archivos o programas [1].

Otra definición sobre sistema distribuidos es: ‘Una colección de computadoras independientes que dan al usuario la impresión de constituir un único sistema coherente’ [2]. De estas dos definiciones se puede resaltar dos términos importantes: computadoras y red. Es requisito fundamental poseer más de una computadora y una conexión entre ellas que determinen una relación de intercambios de mensajes, balanceo de carga o compartimiento de recursos. Por ello, una definición básica es: Sistema de cómputo conectado en una red de alta velocidad [3].

Un sistema distribuido está constituido por dos elementos principales: La red y los nodos, la red es por donde viaja la información, es el canal de comunicación que permite enviar los mensajes desde un computador a otro. Un nodo es un punto de conexión en una red, es cualquier elemento que posea una dirección única de red como IP o MAC.

En los sistemas distribuidos, un nodo puede ser un computador o un servidor.

Las principales ventajas de los sistemas distribuidos son [3]:

- Tienen una proporción precio / desempeño mucho mayor a la de un sistema centralizado.
- Varias computadoras trabajan de manera conjunta con un mismo propósito.
- Poseen una mayor confiabilidad, si falla una máquina el sistema puede seguir funcionando a pesar de tener una perdida, a diferencia de un sistema centralizado, donde si falla la máquina se pierde la operatividad del sistema por completo.
- El crecimiento por incremento es más satisfactorio, ya que al tener un sistema distribuido sólo se debe agregar nuevos nodos (procesadores) que cuando se tiene un sistema centralizado y se requiera agregar otra computadora, debido a que en esta última las aplicaciones deberían reconfigurarse o reprogramarse.
- Permite balancear carga dividiendo las peticiones de los usuarios realizadas hacia el sistema logrando distribuirla en varios computadores con la finalidad de no saturar los recursos de hardware de cada máquina.

Entre las desventajas de los sistemas distribuidos se obtiene [3]:

- Aplicaciones distribuidas: La complejidad de desarrollar aplicaciones distribuidas es mayor a las centralizadas, debido a que deben

funcionar en todos los nodos del sistema, algunos lenguajes de programación son más eficientes que otros.

- La red de comunicación entre los nodos del sistema se puede saturar de mucha información y puede generar pérdidas de datos. Si la red falla en un nodo, este queda aislado de la comunicación con el sistema distribuido.
- La seguridad es desventaja en algunos casos, debido a que pueden surgir escenarios donde los datos puedan ser vistos por otras personas sin necesidad de requerir permisos o autorización específica.

Sockets

Una definición de sockets es “Un punto de un enlace bidireccional entre dos aplicaciones que se encuentra en ejecución dentro de una red” [4]. El punto o *endpoint* es una combinación de dirección IP y puerto, cada conexión TCP está únicamente identificada por ambos elementos, de ahí se puede tener múltiples conexiones entre el host y el servidor.

El servidor ejecuta una aplicación en un puerto específico y se queda en estado de escucha, esperando por las peticiones de los clientes. Al conocer la dirección IP y el puerto por donde se encuentra escuchando el servidor, el cliente puede realizar las peticiones para conectarse; Se requiere que el cliente se identifique ante el servidor y para ello el sistema asigna un puerto de salida para lograr la comunicación [4].

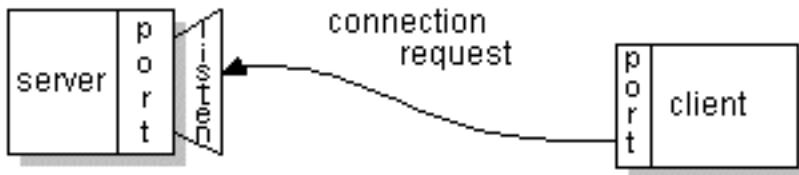


Ilustración 2: Petición Cliente – Servidor. Obtenido de:
<http://docs.oracle.com/javase/tutorial/figures/networking/5connect.gif>

Si el servidor acepta la conexión, este obtiene la dirección y puerto del cliente y redirige el socket local a otro puerto para poder seguir escuchando por nuevas peticiones.

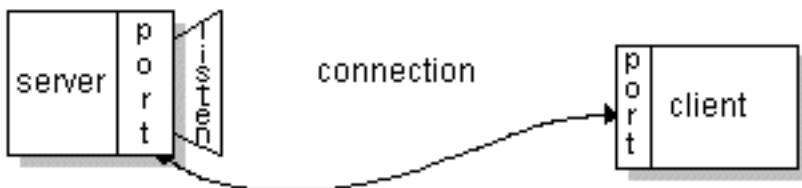


Ilustración 3: Respuesta Servidor – Cliente. Obtenida de:
<http://docs.oracle.com/javase/tutorial/figures/networking/6connect.gif>

En el cliente, si la conexión es aceptada, se crea el socket entre ambas partes y se puede empezar a intercambiar información a través del canal [4].

Raspberry Pi

Es una computadora del tamaño de un tarjeta de crédito que permite ejecutar las tareas básicas como navegar por internet, procesar documentos de textos, hojas de cálculos, juegos, programar, entre otros. El Raspberry Pi fue creado con la intención de enseñar a los niños a programar, tiene un costo de 35 \$ por lo que lo hace la computadora más

barata que se puede conseguir en el mercado permitiéndole a países en vías de desarrollo adquirir una computadora accesible a personas con bajos recursos. Existen dos modelos: A y B, el modelo B posee un procesador ARMv6 de 700 MHz, 512mb de memoria RAM, dos puertos USB 2.0, puerto Ethernet, entrada para audio, rca video y HDMI. No contienen disco duro, por lo que el sistema operativo es almacenado dentro de una memoria SD [5].

Entre los sistemas operativos disponibles para los Raspberries Pi se encuentra una variante de Debian llamada Raspbian que se encuentra compilado para trabajar con procesadores la versión 6 del ARM contenido en el Raspberry Pi [6]. Otros sistemas operativos Linux que pueden utilizar son: ArchLinux y Fedora. Como lenguajes de programación tiene soporte principalmente para Python el cual es el recomendado por la fundación Raspberry para la educación, Perl, C y Java.

RASPBERRY PI MODEL B

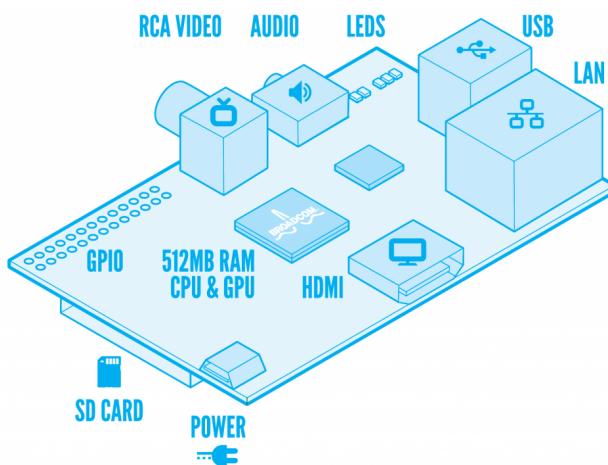


Ilustración 4: Componentes Raspberry Pi. Obtenido de: <http://www.raspberrypi.org/wp-content/uploads/2011/07/RaspiModelB-1024x902.png>

Agente

Un agente es un software que realiza acciones continuas y automáticas dentro de una computadora, se ejecutan en *background*, por lo que el usuario puede realizar su trabajo sin notar la presencia del agente [7]. Pueden realizar tareas dependiendo de su objetivo: recolectar información, monitorear una aplicación o proceso, enviar datos hacia otro computador o agentes, entre otros.

La idea de desarrollar agentes es generar un software que funcione como un ‘robot’ que permita realizar tareas dentro del sistema sin la interrupción del usuario. Entre los atributos que definen a un agente se tienen [8]:

- Reactividad: El agente posee la capacidad de actuar de forma selectiva
- Autonomía: Debido a que se inicia por sí mismo y posee un propósito.
- Comportamiento Colaborativo: Puede actuar con otros agentes para completar una meta en común.
- Capacidad inferencial: Puede actuar sobre tareas utilizando conocimiento previo de los objetivos generales y los métodos preferidos para lograr flexibilidad.
- Continuidad temporal: Persistencia de su estado durante largo periodos de tiempo.
- Movilidad: La capacidad de migrarlo de manera auto dirigida de un host a otro.

METODOLOGÍA

Para este trabajo especial de grado se utilizará una adaptación de la metodología eXtreme Programming (XP) en el desarrollo de los componentes de la infraestructura. XP es una metodología ágil que permite desarrollar software a través de historias de usuario, programación en parejas, la cooperación del cliente y el constante uso de pruebas para garantizar que el código generado cumpla con sus objetivos [9]. Se utilizan iteraciones para entregar pequeñas versiones del software que satisfacen un conjunto de requerimientos.

La siguiente ilustración muestra las etapas del ciclo de vida de un proyecto utilizando XP:

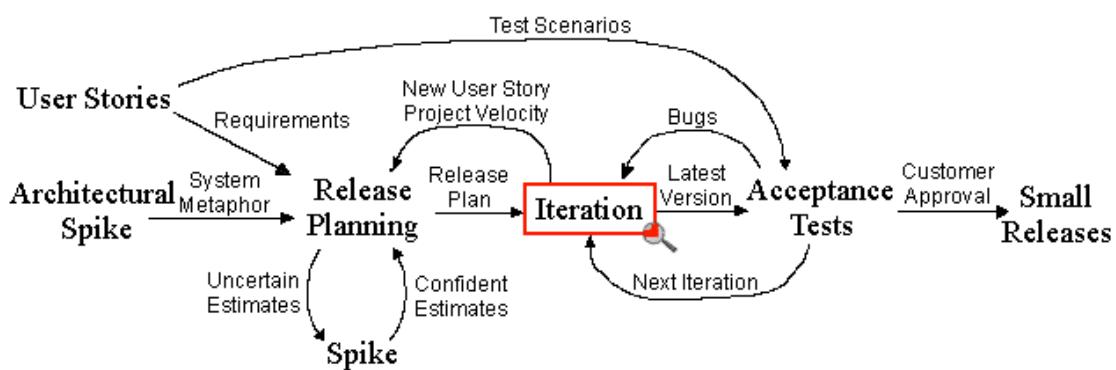


Ilustración 5: Ciclo de Vida en XP. Obtenido de:
<http://www.extremeprogramming.org/map/images/project.gif>

Se empieza con las historias de usuarios, estas permiten conocer los requerimientos necesarios para cumplir con los objetivos del proyecto. Una historia de usuario es un texto breve que indica una funcionalidad del proyecto, se escriben por parte del cliente con un lenguaje natural, sin necesidad de agregar sintaxis técnica. Las historias poseen una estimación de desarrollo que puede ser una, dos o tres semanas. Más de tres semanas indica que la historia es muy compleja y debe dividirse en sub-historias, menos de una semana indica

que la historia es muy simple y se debe combinar varias historias para lograr una estimación mayor. Las historias de usuario se enfocan en las necesidades del usuario, por ello, se debe evitar colocar algoritmos o especificaciones de alguna tecnología.

Al finalizar la elaboración de las historias de usuario, se crea el plan de iteraciones que divide el proyecto en partes, colocado en cada iteración un conjunto de historias de usuarios relacionadas. Normalmente las iteraciones poseen una duración de una a tres semanas y al finalizar la iteración se obtiene un entregable que el cliente debe aprobar.

En cada iteración se logran realizar las siguientes etapas:

- Diseño: Se elabora el diseño de cada una de las historias de usuario que fueron acordadas en la iteración actual. Se crean las tarjetas CRC (clase, responsabilidad y colaboración) que permiten conocer que clases interactuarán en el software, están compuestas por: Nombre de la clase, clases hijo (si aplica), responsabilidad (descripción de los métodos) y colaboración (relaciones con otras clases).
- Codificación: Se codifican las historias de usuarios de la iteración, respetando las siguientes normas: estándares de codificación, creación de las pruebas unitarias antes que el código, reutilización de código, cooperación del cliente.
- Pruebas Unitarias: Se ejecutan luego de codificar las historias de usuario y deben ser exitosas para culminar la iteración y liberar la siguiente versión del software.

Al terminar la iteración se realizan las pruebas de aceptación por parte del cliente, permiten verificar si el entregable obtenido logró cumplir con las especificaciones indicadas por las historias de usuario. En caso negativo, se debe realizar una nueva iteración para corregir errores o defectos que puedan

haber surgido en el desarrollo, por el contrario, si el software es aceptado por el cliente se habrá liberado una versión del software final y se continua con la siguiente iteración del plan de iteraciones [10].

La adaptación de la metodología consiste en primer lugar en establecer el plan de iteraciones de acuerdo a los componentes necesarios en la infraestructura: Cada iteración representa un software de la infraestructura (aplicaciones sistemas distribuidos, módulos, agente, librería), adicionalmente se utilizará un único programador para realizar la fase de codificación, por lo que se obtiene una mayor duración en el desarrollo de las historias de usuario debido a que una misma persona debe tener en cuenta la legibilidad, el uso de estándares y la optimización del código mientras se programa la historia de usuario.

La elección de la metodología XP se debe a que los objetivos en la elaboración de la infraestructura consisten en desarrollar un grupo de componentes. XP se enfoca específicamente en crear software que funcione más allá de elaborar documentación extensa que no es útil cuando el software no realiza su propósito correctamente. Utilizando historias de usuario se obtienen los requerimientos de cada componente de la infraestructura de forma precisa. Mediante iteraciones se logra dividir el proyecto en partes, cada iteración se encuentra dirigida a un componente de la infraestructura y permite desarrollarlos de manera independiente a excepción de las aplicaciones de sistemas distribuidos que debido a la cantidad de aplicaciones a desarrollar se dividieron en dos grupos de iteraciones. A través del uso de pruebas unitarias se garantiza que cada método de las aplicaciones a desarrollar funciona de forma correcta y sin errores. La reutilización de código estará presente en los componentes de la infraestructura pero principalmente en las aplicaciones educativas de sistemas distribuidos, donde en muchas existirán el esquema de servidor y el de cliente.

REFERENCIAS BIBLIOGRÁFICAS

Sistemas Distribuidos:

1. Coulouris, G. (2012). *Distributed Systems: Concepts and Design* (5ta ed.).
2. Tanenbaum, A., & Van Steen, M. (2008). *Sistemas Distribuidos: Principios y Paradigmas*.
3. Tanenbaum, A. (1995). *Sistemas Operativos Distribuidos*.

Sockets:

4. Oracle. (2013). *What is a socket?* Obtenido de <http://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>

Raspberry Pi:

5. The Raspberry Pi Foundation. (2013). *Raspberry Pi*. Obtenido de <http://www.raspberrypi.org/>
6. Raspbian. (n.d.). *Raspbian*. Obtenido de <http://www.raspbian.org/>

Agente:

7. Zwass, V. (2013). *Encyclopædia Britannica: Agent (Computer Science)*. Obtenido de <http://www.britannica.com/EBchecked/topic/705121/agent>
8. Bradshaw, J. M. (1997). *Software Agent: An introduction to software agents*. Obtenido de <http://agents.umbc.edu/introduction/01-Bradshaw.pdf>

Programación Extrema:

9. Jeffries, R. E. (1999 - 2013). *XProgramming: An agile software development resource*. Obtenido de <http://xprogramming.com/>
10. Wells, D. (2009). *Extreme Programming: A gentle introduction*. Obtenido de <http://www.extremeprogramming.org/>

CRONOGRAMA DE TRABAJO

<input type="checkbox"/> Diseño e Implementación de Infraestructura		80 days	Mon 10/06/13	Wed 02/10/13
Diseño e Implementación Base de Datos		5 days	Mon 10/06/13	Fri 14/06/13
Creación Scripts Sistema Operativo		2 days	Mon 17/06/13	Tue 18/06/13
<input type="checkbox"/> eXtreme Programming (XP)		68 days	Wed 19/06/13	Wed 25/09/13
Elaboración Historias de Usuario		10 days	Wed 19/06/13	Wed 03/07/13
<input type="checkbox"/> Desarrollo de agente de configuración (Iteración 1)		6 days	Thu 04/07/13	Fri 12/07/13
Diseño Historias de Usuario		1 day	Thu 04/07/13	Thu 04/07/13
Codificación Historias de Usuario		4 days	Mon 08/07/13	Thu 11/07/13
Pruebas		1 day	Fri 12/07/13	Fri 12/07/13
<input type="checkbox"/> Desarrollo de servicio de recepción (Iteración 2)		4 days	Mon 15/07/13	Thu 18/07/13
Diseño Historias de Usuario		1 day	Mon 15/07/13	Mon 15/07/13
Codificación Historias de Usuario		2 days	Tue 16/07/13	Wed 17/07/13
Pruebas		1 day	Thu 18/07/13	Thu 18/07/13
<input type="checkbox"/> Desarrollo librería de envío de mensajes (Iteración 3)		4 days	Fri 19/07/13	Thu 25/07/13
Diseño Historias de Usuario		1 day	Fri 19/07/13	Fri 19/07/13
Codificación Historias de Usuario		2 days	Mon 22/07/13	Tue 23/07/13
Pruebas		1 day	Thu 25/07/13	Thu 25/07/13
<input type="checkbox"/> Desarrollo de Aplicaciones Sistemas Distribuidos Parte 1 (Iteración 4)		15 days	Fri 26/07/13	Thu 15/08/13
Diseño Historias de Usuario		3 days	Fri 26/07/13	Tue 30/07/13
Codificación Historias de Usuario		10 days	Wed 31/07/13	Tue 13/08/13
Pruebas		2 days	Wed 14/08/13	Thu 15/08/13
<input type="checkbox"/> Desarrollo de Aplicaciones Sistemas Distribuidos Parte 2 (Iteración 5)		15 days	Fri 16/08/13	Thu 05/09/13
Diseño Historias de Usuario		3 days	Fri 16/08/13	Tue 20/08/13
Codificación Historias de Usuario		10 days	Wed 21/08/13	Tue 03/09/13
Pruebas		2 days	Wed 04/09/13	Thu 05/09/13
<input type="checkbox"/> Desarrollo Modulo Administración Ciclo de Vida de nodos (Iteración 6)		4 days	Fri 06/09/13	Wed 11/09/13
Diseño Historias de Usuario		1 day	Fri 06/09/13	Fri 06/09/13
Codificación Historias de Usuario		2 days	Mon 09/09/13	Tue 10/09/13
Pruebas		1 day	Wed 11/09/13	Wed 11/09/13
<input type="checkbox"/> Desarrollo Modulo Monitoreo (Iteración 6)		5 days	Thu 12/09/13	Wed 18/09/13
Diseño Historias de Usuario		1 day	Thu 12/09/13	Thu 12/09/13
Codificación Historias de Usuario		3 days	Fri 13/09/13	Tue 17/09/13
Pruebas		1 day	Wed 18/09/13	Wed 18/09/13
<input type="checkbox"/> Desarrollo Módulo de Gestión de Tópicos y Aplicaciones (Iteración 7)		5 days	Thu 19/09/13	Wed 25/09/13
Diseño Historias de Usuario		1 day	Thu 19/09/13	Thu 19/09/13
Codificación Historias de Usuario		3 days	Fri 20/09/13	Tue 24/09/13
Pruebas		1 day	Wed 25/09/13	Wed 25/09/13
<input type="checkbox"/> Otras actividades	Documentación del Sistema	5 days	Thu 26/09/13	Wed 02/10/13
		5 days	Thu 26/09/13	Wed 02/10/13