PROTOTIPO DE SISTEMA DE GESTIÓN DE INVENTARIO EMPLEANDO DISPOSITIVOS MÓVILES.

Galindo Rodríguez Gabriel Adolfo
2128265
Maldonado Ramos Julián David
2128329

Dario Segura Profesor Tutor

UNIVERSIDAD SANTO TOMAS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2017

TABLA DE CONTENIDO

1. PROBLEMA

En la actualidad las tecnologías de la información y las comunicaciones (TIC's) son muy importantes para la competitividad empresarial, en cuanto se presente una mejora en la articulación de las TIC's en una compañía, seguramente esto mejorará la gestión empresarial y en muchos casos optimiza el valor agregado de una organización.

En Orange Business Services, se maneja un sistema de inventario que permite ubicar un equipo según sea su orden de compra (conocida cómo PO) o serial. Asimismo el inventario cuenta con información extra de cada equipo como lo es el cliente, nombre del proyecto y marca. Todo este inventario se maneja por medio de un formato en Excel y un registro en una plataforma virtual llamada Camaleón. Dado que las limitaciones de manejar una base de datos por medio de un formato convencional en Excel en una red local cuyo alcance es sumamente protegido por los protocolos de seguridad de Orange, puede conllevar a limitaciones tan simples cómo no poder abrir el formato en un equipo con características específicas cómo lo sería tener el software de Microsoft office o simplemente sólo puede ser observado por una persona que tenga acceso a la red, se plantea crear el prototipo de una herramienta que pueda mejorar el manejo de la información que contiene el inventario, con un ambiente interactivo con el usuario, y fácil de usar para que funcione por medio de un sistema conformado por una red LAN y WAN, que a su vez permita alojar una base de datos y que esta base de datos se pueda acceder por medio de un servicio desde una App móvil de naturaleza de Android, también se pueda acceder la base de datos desde un computador del sistema operativo (SO) Windows.

2. ANTECEDENTES

A continuación, se muestran algunas herramientas digitales que funcionan similarmente al sistema propuesto, algunas de ellas se relacionan con los servicios a los dispositivos móviles con sistemas operativos (SO) Android. Entre las herramientas que se citan después de este párrafo, se encuentran proyectos en los cuales se diseñaron redes para las empresas (en su mayoría redes WAN y LAN), herramientas que asocian redes con aplicaciones móviles y el uso de bases de datos en un servidor con una aplicación móvil.

- **Diseño de una aplicación Android, que recolecta información de tráfico y su comunicación a un servidor:** En este proyecto se desarrolla una herramienta que es una aplicación móvil para los dispositivos Android. La App posee la capacidad de establecer un enlace cliente – servidor, en la red se permite compartir una base de datos en MySQL, esta base de datos se almacena en un servidor. De manera que este antecedente aporta información en cuanto al desarrollo de la aplicación Android y manejo de MySQL. (Godoy, 2015)

- **Desarrollo de una aplicación móvil Android para control remoto de un servicio web:** El objetivo de esta herramienta consiste en el desarrollo de una aplicación para el sistema operativo Android, que permita obtener una interfaz de usuario conectada a un servicio web. Donde resulta un vínculo con el proyecto, en vista al desarrollo e información de la aplicación Android. (Lozano Pérez, 2012)

- **Diseño de una red WAN para una compañía nacional (España):** Este es un proyecto que pertenece al área de Integración de Redes Telemáticas. Se trata del diseño y de una red WAN desarrollada para una empresa española que interconectará a las diferentes sedes, repartidas por toda la península. De modo que aporte al proyecto una visión en cuanto a la topología de la red que se desea diseñar (Red WAN) (Pavón, 2012)

- **Aplicación Android para la empresa Travelling-Service:** Esta fue una herramienta desarrollada con la finalidad de asociar una página web de la

empresa Travelling Service mediante el uso de servicios de telecomunicaciones y bases de datos. De ahí se puede referenciar unos parámetros básicos para tener en cuenta en el momento de diseñar una herramienta móvil para una empresa. (Matesanz, 2014)

- **Herramienta Wake On Lan:** Esta es una herramienta de naturaleza de escritorio remoto, es decir es una aplicación que permite controlar un ordenador de forma remota. La herramienta “Wake On Lan” es de uso exclusivo para el SO Android. Esta herramienta se asocia cómo un antecedente del proyecto, dado que asocia una red LAN con una aplicación móvil del SO Android. (El Androide Libre, 2015)

- **Implementación de aplicación web con acceso a base de datos para manejo de inventario de la empresa Orange Business Services Colombia S.A, por el autor Juan F. Gómez para la Facultad de Ingeniería Electrónica de la Universidad Santo Tomás:** .Esta aplicación es reciente a la propuesta del proyecto, en ella se realiza una página web que pública la base de datos y es protegida por medio del uso de usuario y contraseña. En este trabajo se describe un proceso completamente distinto, se propone tener la base de datos en un computador, que permita compartir la información de la misma, sin la necesidad de volver pública la base de datos. Asimismo, en este trabajo no se hace referencia al desarrollo web, más bien, al uso de web services, esto diferenciará totalmente el desarrollo del proyecto, pues en el trabajo de la aplicación web se realiza la conexión de manera directa a la base de datos por el uso de la página, en el trabajo propuesto se implementarán los servicios web para acceder a la información, protegiendo de mejor manera la información de la base de datos y a su vez, proponiendo un reto a los desarrolladores del proyecto, que es implementar distintos métodos de conexión con una base de datos, como lo son los drivers de MySQL, para poder comunicarse con los programas (de naturaleza de Java) que deseen ver la información. Por otra parte, la aplicación web no necesita de un desarrollo en la configuración de la red a implementar, en el prototipo propuesto, sí se realizará el diseño de la red sobre la cual se va a implementar, tema que no se abordó en el trabajo. También, el prototipo va a contar con un desarrollo de aplicación móvil, el desarrollo va a ser en el lenguaje nativo y oficial de la empresa Google para Android. El trabajo del desarrollo web aún no ha sido publicado a la fecha de entrega de este documento.

3. JUSTIFICACIÓN

En la industria es frecuente divisar nuevas implementaciones y cambios tecnológicos de manera acelerada con respecto a épocas anteriores, dado que existen diferentes software y herramientas que generan un nuevo espacio de trabajo, que optimizan el acceso a la información necesaria para realizar alguna operación logística, técnica o del quehacer diario en el entorno laboral de cualquier organización; esto permite brindar calidad en el desarrollo e innovación de los servicios en el campo del almacenamiento. Asimismo, la tecnología permite acentuar un sistema adecuado en cualquier área empresarial, agilizando y facilitando la manera de buscar información de una manera más rápida y sencilla.

Según sea el cambio tecnológico, es importante actualizar cada sistema partiendo de la tecnología que se pueda desarrollar o aplicar. Dada la capacidad tecnológica que se puede brindar a la fecha (2016), se propone modificar la forma en que se controla el inventario de la compañía Orange Business Services, diseñando una herramienta que sea amigable con el usuario y permita controlar de forma remota el inventario de la empresa.

Con base en a las ventajas que brinda la tecnología a las organizaciones, proponer y diseñar una herramienta que permita verificar de forma eficiente la información, es siempre de gran ayuda para una organización, por ello se plantea crear una base de datos para las plataformas móviles (Android) o en sistemas operativos convencionales (Windows) con base a una red WAN y LAN (la red se diseña con el fin de que sea una herramienta que pueda acceder al inventario desde distintos puntos). Con la única finalidad de optimizar los procesos logísticos de la empresa.

4. OBJETIVOS

4.1 OBJETIVO GENERAL

- Desarrollar un prototipo de un sistema que permita gestionar la información de inventario en base de datos, con un dispositivo móvil a través de redes WAN y LAN para agilizar el proceso del manejo de inventario en una empresa como Orange Business Services S.A.

4.2 OBJETIVOS ESPECÍFICOS

- Describir el proceso del manejo de inventario que se desea sistematizar para poder extraer los requerimientos funcionales del prototipo a implementar.
- Definir los componentes del sistema con sus funcionalidades para planear el desarrollo del prototipo y medir el avance del desarrollo.
- Diseñar el modelo de datos del sistema basado en los requerimientos para garantizar la disponibilidad de la información en el sistema a implementar.
- Desarrollar una interfaz en java para dar facilidad de manejo en la base de datos a múltiples usuarios que desconocen el manejo de bases de datos en MySQL.
- Diseñar la infraestructura de redes necesaria para acceder la información del sistema desde un dispositivo móvil.
- Simular la infraestructura diseñada para emplearla como canal de comunicación entre un dispositivo móvil y la información de inventario.
- Desarrollar los servicios web necesarios para soportar los requerimientos extraídos y sirva como interface entre los dispositivos móviles y la información en la base de datos.
- Desarrollar una aplicación en un dispositivo móvil que consuma los servicios web desarrollados para que el sistema conformado cumpla con los requerimientos extraídos.

5. ALCANCES Y LIMITACIONES

Se plantea utilizar una red LAN y una red WAN, estas redes tienen como finalidad permitir acceder a una base de datos en MySQL desde distintos equipos, mediante una interfaz de java. Asimismo, un dispositivo móvil puede acceder a la base de datos por medio de la red, pero teniendo en cuenta que puede ingresar y consultar información a la base de datos por medio de servicios web en una aplicación móvil.

Empero, se debe tener en cuenta que existe una limitación en la implementación de una red WAN dada a la imposibilidad de poseer información de la topología de la red de Orange Business Services, pues la susodicha información pondría en riesgo muchos servicios que provee la empresa.

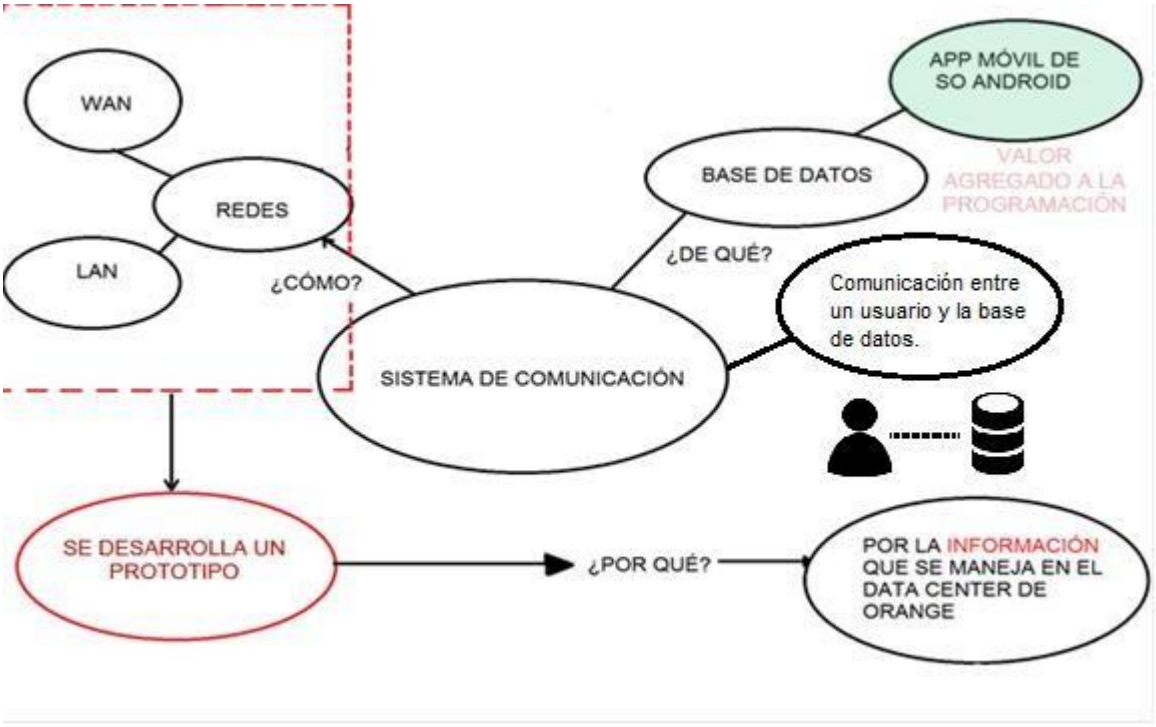


Figura 1. Mapa conceptual del proyecto y su dependencia de la información que NO puede proveer Orange.

Una limitación del proyecto se encuentra explícita en la Figura 1. En la región punteada de color rojo, la limitación es, que en cuanto a alguna posible implementación del sistema, la red no puede ser implementada por cuestiones de seguridad de la empresa (Orange Business Services), debido a que los protocolos

de seguridad de la empresa no permiten ninguna implementación que pueda comprometer la información que se encuentra en la región conocida cómo DATA CENTER.

En un principio, se plantea diseñar un sistema de comunicación entre un usuario y una base de datos. Un sistema de comunicación tiene un medio para transmitir, el medio de transmisión son las redes, esto se puede verificar en la Figura 1 en la región punteada. A su vez, cómo cualquier sistema de comunicación debe poseer un mensaje a transferir, el mensaje que se comparte es la información de la base de datos, esta base de datos se implementa por medio de un motor de bases de datos, los motores de bases de datos permiten ser compuestos por interfaces programadas por algún programador en lenguajes convencionales, así que se plantea implementar una interfaz en java y una aplicación móvil (del SO Android), definiendo dicho desarrollo como un valor agregado del sistema. Este valor agregado permite al usuario manipular la base de datos, sin importar que el usuario desconozca del manejo de SQL.

Asimismo, se cuenta con una serie de equipos que permiten el desarrollo de una red LAN.

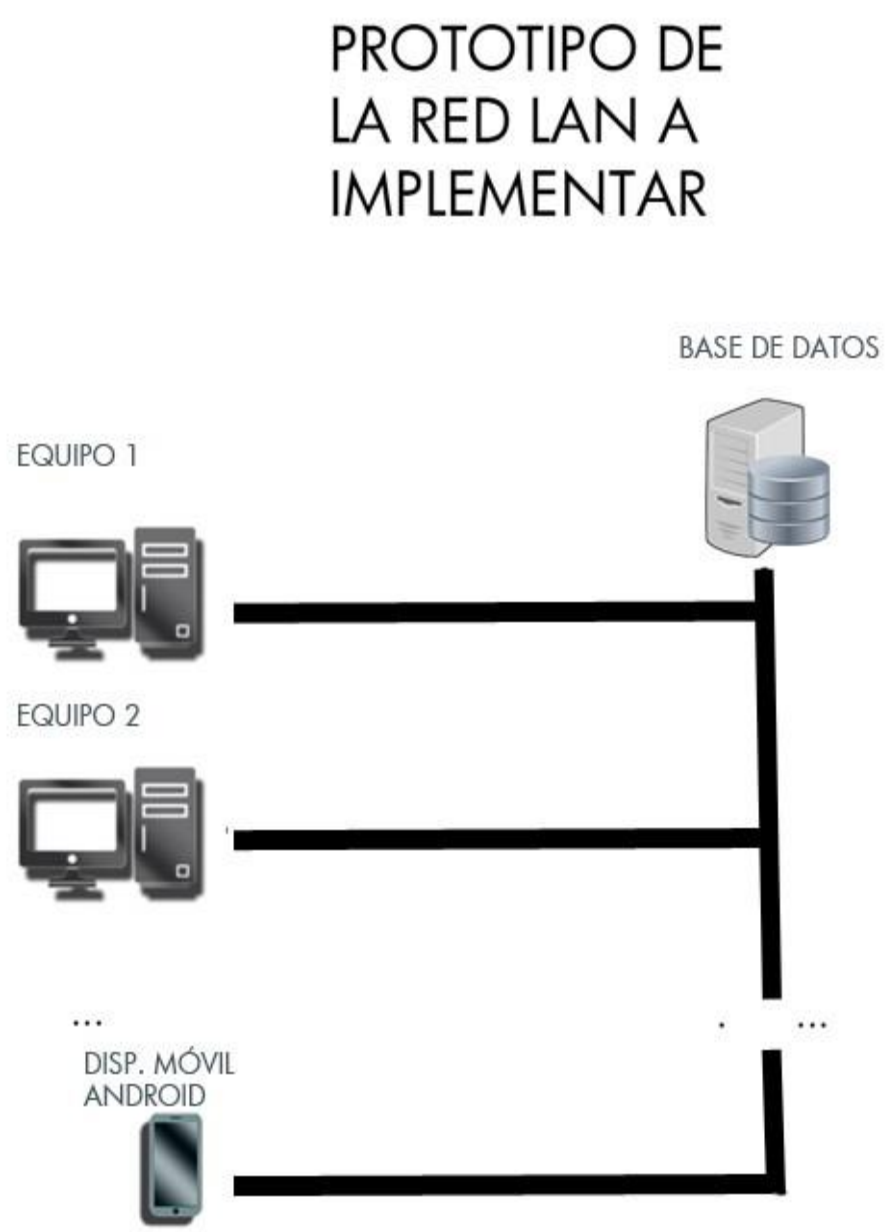


Figura 2. Topología de la red LAN a implementar en el prototipo.

En la Figura 2 se muestra la topología de la red LAN que se desea implementar en el prototipo compuesto por dos hosts, en uno de los hosts se almacenara la base de datos, de tal manera que esta se pueda compartir en ambos hosts, adicionalmente, se añade un dispositivo móvil a la red para que de igual forma tenga acceso a la base de datos.

La red WAN se planea diseñar con unas redes LAN que pueden estar en dos lugares distantes, a nivel metropolitano o más allá. La Figura 3 muestra la topología de la red WAN que se planea diseñar.

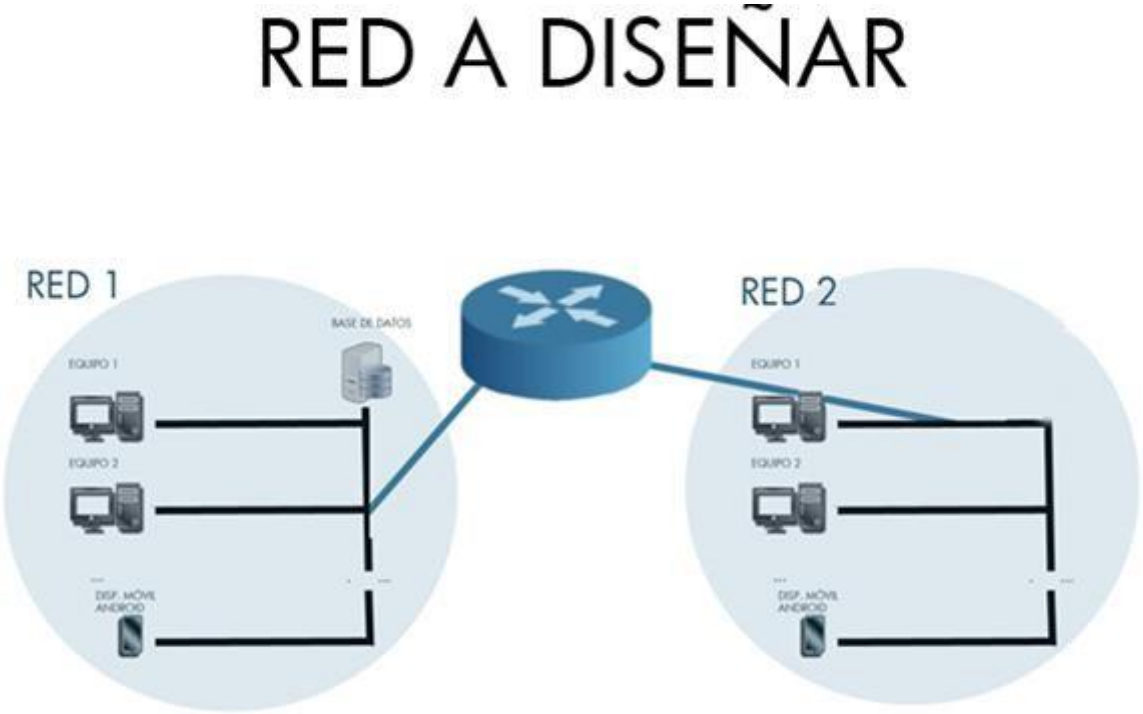


Figura 3. Topología del diseño de la red WAN.

Como se había explicado anteriormente, se encuentra una restricción en cuanto a la seguridad de la compañía y de las múltiples empresas que poseen contrato con la misma. Por tal motivo se también imposibilita la implementación de la red WAN.

Por último, en cuanto a la programación, se plantea hacer uso de sentencias SQL para la creación de la base de datos, el motor de la base de datos es el motor MySQL, manejado por una interfaz programada en JAVA. Para el dispositivo móvil se debe hacer uso de JAVA y XML orientado a la programación de dispositivos Android. Todos los lenguajes son en su mayoría programación orientada a objetos.

6. FACTIBILIDAD

Un sistema que emplea redes y alojamiento de bases de datos es de bastante utilidad para cualquier compañía a nivel mundial, éste sistema brinda algunas ventajas como lo son el orden, control y un fácil manejo de los archivos o datos que se deseen manipular. Sin embargo, un error habitual es no innovar las herramientas cómo lo son las redes, los equipos, entre otros. A su vez, la base de datos se presenta sin ningún valor agregado, complicando el manejo de la información a los usuarios. Por esto se puede desarrollar una herramienta completa con el uso de redes y bases de datos, para facilitar el acceso de los usuarios a esta información.

El sistema que se va desarrollar, que es un valor agregado para los usuarios, requiere más desarrollo por parte del ingeniero, pero dará más ventajas a los usuarios del sistema.

Dando así una mayor competitividad a un servicio de redes, como el que presta OBS, permitiendo acceder a bases de datos, y que este servicio pueda funcionar de manera adecuada en equipos de cómputo o en tecnología móvil.

Un sistema de inventario es importante para las compañías, los inventarios influyen en la toma de decisiones, estas decisiones son en su mayoría de operaciones logísticas. En Orange Business Services, el inventario toma un rol muy importante, porque es dependiendo del inventario, que se toma la decisión de entregar un equipo en específico, un equipo que se acomode a las exigencias del cliente. Asimismo, influye en la decisión de las tarjetas, cables, entre otros objetos que se vayan a entregar al cliente. Por otra parte, el inventario de OBS se debe manejar de forma eficaz, dado que las acciones tales como: inconvenientes o reventas al cliente, se deben realizar de forma inmediata. Por último, el inventario de OBS debe mantenerse actualizado lo más rápido posible, pues cómo Orange es una empresa de carácter internacional, sus dueños de Francia TELECOM, deben poseer información al día de cada movimiento de los activos fijos de la compañía, con el fin de realizar un seguimiento de cada activo para no sufrir pérdidas.

Aunque la empresa Orange Business Services no dispone el espacio para instalar el prototipo del sistema propuesto, una base del prototipo se puede alojar en dos equipos, configurando debidamente su red y desarrollando las interfaces anteriormente descritas.

7. MARCO TEÓRICO

7.1 Bases de Datos

7.1.1 Introducción a las bases de datos

Las bases de datos son un componente esencial del día a día de la sociedad moderna, por ejemplo, al instante de realizar alguna reserva vía internet, los bancos deben poseer una base de datos del cliente, dicho cliente es el encargado de realizar la reserva. Asimismo, para realizar la operación de venta o la reserva vía internet, se debe tener una base de datos o generar una base de datos sobre el cliente que procedió con el pago vía internet.

Este tipo de operaciones, como realizar un pago vía internet, se conocen como las aplicaciones tradicionales de las bases de datos, en las cuales la mayor parte de la información es guardada y accedida por los usuarios. Las bases de datos multimedia ya pueden contener imágenes, videos y sonidos de mensajes. Los sistemas de información geográfica (conocidos por sus siglas en ingles GIS) pueden contener y analizar mapas, datos climáticos e imágenes satelitales. (Elmasri & Navathe, 2003)

7.1.2 Introducción al modelado de datos

A razón del manejo excesivo de datos, es necesario aplicar algunas técnicas o herramientas para la compresión conceptual de una base de datos, que se debe desarrollar. Una técnica es el modelado de datos, que funciona en pro de la organización y de la documentación de los datos en el sistema. El resultado de un modelo de datos permite crear de forma rápida y sencilla una base de datos, que cumpla con las necesidades de almacenamiento de datos. (Alarcón, 2006)

Existen varias formas de realizar un modelado de datos, que en sí define cómo se relacionan todos los elementos, conjuntos y subconjuntos, en una base de datos. Básicamente existen tres tipos de modelado de datos. (Aguilar) Que son:

- Conceptual: muy general y abstracto, visión general entre las relaciones de cada elemento que influyen en los datos.
- Lógico: versión completa que incluye todos los detalles acerca de los datos.

- Físico: esquema que se implementará desde un manejador de bases de datos (DBMS).

Existen diversas formas de representar los datos, que necesita un sistema de información, pero el más utilizado y extendido es el diagrama entidad-relación, que se basa en la definición de entidades y de relaciones entre los datos.

Un diagrama de entidad-relación (DER) es una herramienta de modelado de datos, que describe las asociaciones que existen entre las diferentes categorías de datos dentro de un sistema de información. Los diagramas de datos sirven para representar la manera en que se almacenan los datos en un sistema de información, también permite representar gráficamente el sistema de información.

7.1.3 Modelo relacional

Se debe entender el modelo relacional como aquel tipo de modelo de organización y gestión de bases de datos, que consiste en el almacenamiento de datos en tablas compuestas por filas y columnas o campos. Se distingue de otros modelos, como el modelo jerárquico, por ser más comprensible para el usuario inexperto, y por basarse en la lógica de predicados para establecer relaciones entre distintos datos. Surge como solución a la creciente variedad de los datos que integran las *data warehouses* y se puede resumir el concepto como una colección de tablas (relaciones).

Un elemento que se debe entender, es la palabra **entidad** como cualquier ente o cosa, real o abstracta, de la cual se desea almacenar información. Asimismo, se debe entender que una entidad puede ser una abstracción de un conjunto de objetos similares que se pueden describir con los mismos tipos de datos. (Alarcón, 2006)

Otro elemento que influye en el análisis de un modelo entidad –relación, son los **atributos**, que define como las características, cualidades, rasgos principales o peculiaridades esenciales de una entidad, toman un valor especifico en una instancia particular.

Por último, se debe comprender la **relación**, que se define como el vínculo que permite definir una dependencia entre los conjuntos de dos o más entidades.

En la Figura 4 se ilustra un esquema de cómo es un modelo entidad-relación para un proyecto estudiantil.

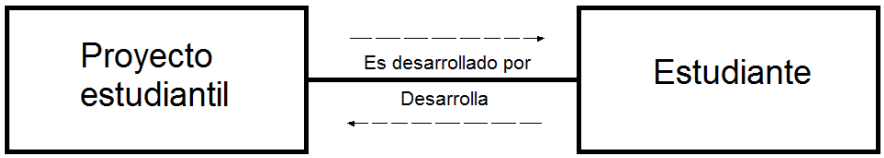


Figura 4. Modelo entidad-relación entre un estudiante y un proyecto estudiantil.

En cuanto se refiere a bases de datos, es común encontrarse con el término de “modelo relacional” para identificar al conjunto de conceptos, herramientas y métodos que se utilizan para el diseño de bases de datos relacionales.

En el presente trabajo, se discute sólo el modelo relacional porque los demás tienen únicamente interés histórico. Por excepción del enfoque de bases de datos orientado a objetos. (René C Martínez Baños, 2005)

7.1.4 Motores de bases de datos

Un motor de base de datos se define como un servicio que se utiliza para almacenar y procesar datos en formato relacional (tabular) o como documentos XML. (Microsoft)

Los motores de bases de datos deben cumplir con unas tareas principales, para poder satisfacer las necesidades del cliente. Las tareas principales son:

- “Diseñar y crear una base de datos que contenga las tablas relacionales o los documentos XML que el sistema necesita.”
- “Implementar sistemas para obtener acceso y cambiar los datos almacenados en la base de datos, lo que incluye implementar los sitios Web o las aplicaciones que funcionan con los datos, así como crear procedimientos que utilicen las herramientas y utilidades de SQL Server para trabajar con los datos.”
- “Aplicar los sistemas implementados en la organización o en los clientes”
- “Proporcionar soporte técnico administrativo diario para optimizar el rendimiento de la base de datos.” (Microsoft)

Entre los motores de bases de datos más reconocidos están MySQL, que es de la compañía ORACLE, asimismo, está el motor de base de datos de Microsoft denominado cómo Microsoft SQL Server, está PostgreSQL ampliamente reconocido en la seguridad informática (PostgreSQL, 2016).

La mayoría de motores de bases de datos, deben tener las mismas sentencias de SQL, con variaciones muy despreciables de una sentencia de un motor a otro, esto se debe a qué SQL hace referencia al acrónimo lenguaje de consulta estructurado (en inglés Structured Query Language), así lo determina el estándar de SQL que nació en el Instituto Nacional de Estándares y Tecnología (NIST). (Deutsch, 2013)

A continuación se presentan varias alternativas de motores de bases de datos.

7.1.4.1 MySQL

Es uno de los motores más usados por los programadores. Páginas cómo YouTube, Paypal, Google, Facebook, Twitter, ebay, Cisco, LinkedIn y Adobe cuentan con el motor MySQL, para el manejo respectivo de bases de datos y poder operar en la web y ofrecer distintos servicios que implican el manejo de bases de datos. Asimismo, MySQL es el motor de bases de datos de código abierto más importante del mercado. (MySQL, 2016)

“MySQL cuenta con varias presentaciones, una de ellas es MySQL Enterprise, que incluye el conjunto más completo de funciones avanzadas, herramientas de administración y soporte técnico para alcanzar los niveles más altos de escalabilidad, seguridad, fiabilidad y tiempo de actividad de MySQL. Reduce el riesgo, el costo y la complejidad en el desarrollo, implementación y administración de aplicaciones MySQL”. (MySQL, 2016)

Por otra parte, el motor MySQL es un motor de bases de datos de la compañía ORACLE. Por esta relación de MySQL con ORACLE se permite realizar grandes enlaces con herramientas cómo Netbeans.

7.2 Servicios Web

7.2.1 Introducción a los servicios Web

Se define como servicio Web: “… es un sistema de software diseñado para soportar la interoperabilidad máquina - máquina sobre una red…” (W3C, 2004),

entendiendo que la interoperabilidad como la comunicación por medio de protocolos o estándares tales como: SOAP, XML, UDDI y WSDL. Los servidores web también permiten mostrar y hacer disponibles funcionalidades que retornan un valor o resultado. Como lo son los servicios Web con la información del aeropuerto, es un web service que recopila toda la información de los aeropuertos por países, se puede ver un ejemplo de este servicio web en (WEBSERVICEX.NET, 2017). Cada servicio Web se hace responsable de realizar un conjunto de funciones concretas y definirlas de forma correcta, para que sus componentes independientes se puedan integrar, con la finalidad de desarrollar distintos sistemas distribuidos y complejos.

El protocolo más conocido es el protocolo simple de acceso a objetos (SOAP). Este protocolo recopila los mensajes para que puedan ser entregados a la red por medio de un protocolo de transporte, como lo puede ser HTTP, IIOP, SMTP, entre otros. (IBM, Introducción a SOA y servicios web)

El término de servicios Web específica una manera de integrar aplicaciones Web por medio del uso de XML, SOAP, UDDI y WSDL, para el intercambio de información, y de este modo presentar información dinámica al usuario. Diversas aplicaciones de software desarrolladas en cualquier lenguaje de programación y ejecutadas en distintas plataformas pueden hacer uso de servicios Web, para el intercambio de información.

De este modo los problemas que tratan de solventar los servicios Web son los temas de integración de datos y aplicaciones, estos permiten que las compañías puedan almacenar archivos de un sitio y transmitirlos por internet para que de este modo los usuarios puedan acceder a esta información y se puedan introducir funciones administrativas o de operaciones, como cuentas, seguridad, inventarios, nomina, etc. Para la creación de un servicio Web se puede hacer mediante una arquitectura llamada REST, ayudado de protocolos como lo es HTTPS o SOAP.

Los servicios basados en RESET trabajan sobre el protocolo HTTP, este es utilizado como medio de transporte entre cliente y servidor, en cuanto al formato de los datos transmitidos, lo más habitual actualmente es intercambiar la información en formato XML o JSON. (sgoliver.net, 2012) (Alicante, 2014)

7.2.2 Tipos de protocolos empleados en servicios Web

7.2.2.1 SOAP

El Protocolo de acceso a objetos simple (SOAP), es un protocolo de intercambio de información, un protocolo de mensajería generalmente construido en XML. Este protocolo permite que programas que funcionan en distintos sistemas operativos se logren comunicar, esta comunicación se realiza mediante los mensajes SOAP. Los mensajes SOAP, son independientes del sistema operativo que se maneje y por tal motivo se pueden transportar por diversos protocolos de internet.

Este protocolo es el encargado de realizar la codificación y comunicación entre tres aspectos importantes, que son, el interesado en los servicios, el proveedor del servicio y el intermediario de servicios. De esta manera se utiliza un mensaje SOAP para hacer la petición del servicio que se desee (Center, 2017).

“El mensaje SOAP tiene una infraestructura definida para describir lo que se encuentra en el mensaje y como procesarlo. Un mensaje SOAP contiene cabeceras con exactamente un cuerpo. El Mensaje es un documento XML, que proporciona toda la información para el usuario como lo es la dirección del mensaje, el control de la información y el mensaje propiamente dicho. Las cabeceras son las encargadas de la información de control, el cuerpo es el que contiene la información del mensaje y todos sus parámetros” (IBM, SOAP Y HTTPS).

7.2.2.2 HTTPS

La mayoría de las aplicaciones incluyen funcionalidades HTTP, el protocolo seguro de transferencia de hipertexto (HTTPS), este protocolo es un protocolo de comunicaciones utilizado para la comunicación segura. HTTPS se usa frecuentemente para mensajes basados en XML, pero también se puede usar para archivos binarios o sin formato.

Cuando se emplea el protocolo HTTPS, se puede observar el icono de un candado en la barra del navegador que se esté usando, este protocolo de seguridad empezó a ser usado por distintas entidades que trabajan con información personal e intransferible. Si en una página Web se usa HTTPS, esta página se protege por medio de una codificación con certificado digital SSL, creando así un canal más seguro para el transporte de datos del usuario o cliente (Culturacion, 2012) (Artificial, 2008).

“Para los servicios web de IBM, HTTPS es un protocolo no estándar, siendo SOAP el protocolo estándar.” (IBM, SOAP Y HTTPS)

Sin embargo, los Web Services basados en REST (conocido por sus siglas en inglés Representational State Transfer). Son considerados como Web services. La organización de Netbeans considera qué "… REST es una nueva forma de crear y comunicarse con los servicios web. En REST, los recursos tienen URIs y se manipulan a través de operaciones de encabezado HTTP." (Netbeans, 2017)

7.2.3 Librería Volley para Android

Volley es una librería HTTP que facilita la comunicación sobre redes para las aplicaciones de Android. Volley está disponible a través del repositorio AOSP abierto.

Volley ofrece las siguientes ventajas puntuales frente a la programación de servicios Web de manera directa con Java, JavaScript o XML. A continuación, se muestran las ventajas anteriormente nombradas:

- "Programación automática de las solicitudes de red."
- "Múltiples conexiones de red simultáneas."
- "Memoria caché de disco transparente y memoria con coherencia de caché HTTP estándar."
- "API de solicitud de cancelación. Puede cancelar una sola solicitud o puede establecer bloques o alcances de solicitudes para cancelar."
- "Facilidad de personalización, por ejemplo, para reintento y retroceso."
- "Herramientas de depuración y rastreo."

La librería Volley puede soportar desde imágenes, hasta mensajería soportada por JSON. (Android, 2016)

7.3 Aplicaciones móviles

7.3.1 Introducción a las aplicaciones móviles

El software se define convencionalmente como un conjunto de instrucciones lógicas, que permite ejecutar un algoritmo, el software se usa siempre para cumplir uno o más objetivos por un usuario. De acuerdo al estándar 729 de la IEEE el "Software es el conjunto de los programas de cómputo, procedimientos, reglas, documentación o documentos y datos asociados, que forman parte de las operaciones de un sistema de computación" (IEEE, 1983)

En referencia a la Aplicación Móvil, se puede determinar que una aplicación móvil, es la asociación del software orientado a los dispositivos móviles, cómo lo son celulares, tablets (o tabletas), entro otros dispositivos coparticipes con SO Android, cómo lo son relojes inteligentes y demás; teniendo en cuenta las definiciones anteriormente presentadas, y enfocándolo en el área de la computación móvil, las aplicaciones móviles son los conjuntos de instrucciones lógicas, procedimientos, reglas, documentación o documentos y datos asociados, que funcionan específicamente en dispositivos móviles, como por ejemplo teléfonos inteligentes, televisores inteligentes, tablets, reloj, entre otros. (Villegas, 2013)

En el desarrollo de aplicaciones móviles es muy frecuente relacionarse con el término API (siglas de 'Application Programming Interface'), que es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de intermediador entre programas distintos, de la misma manera en que la interfaz de usuario facilita la interacción humano-software. (Merino, 2014) Las APIs en el desarrollo de aplicaciones móviles, permiten al programador relacionarse con otras aplicaciones o en su defecto con funciones específicas del dispositivo o del sistema operativo (SO).

Para la programación en Android se maneja el lenguaje de programación JAVA para la parte lógica de la aplicación, por otra parte se maneja XML para la programación de la interfaz gráfica y requerimientos especiales (cómo solicitud de permisos al usuario o configuración de rutas lógicas).

7.3.2 Plataformas de desarrollo

Para el desarrollo de aplicaciones móviles, existen distintas plataformas de desarrollo. En los siguientes capítulos se muestran las plataformas de desarrollo más usadas para el desarrollo de aplicaciones Android.

7.3.2.1 NetBeans

El entorno de desarrollo integrado (conocido por sus siglas en inglés como IDE) NetBeans, es uno de los más usados por los programadores, sobre todo para el manejo de PHP, JavaFx, C/C++ y JavaScript. NetBeans es un proyecto de código abierto, pensado para trabajar con eficiencia y eficacia en el desarrollo de proyectos de programación. La página oficial de Netbeans ha registrado alrededor de 18 millones de descargas del mismo. (Netbeans, 2017)

Para el desarrollo de aplicaciones móviles, Netbeans cuenta con una extensión llamada NBAndroid. NBAndroid provee al IDE NetBeans del Android SDK (Software development kit o software de desarrollo). Para mayor información sobre NBAndroid

consultar (NBAndroid).

7.3.2.2 Android Studio

El IDE de Android Studio, es el entorno desarrollado por la empresa Google, Android Studio fue anunciado por primera vez en el año 2013 en la conferencia de Google I/O (android, 2016), como el primer IDE estable para el desarrollo de aplicaciones móviles (neoteo, 2014). Android Studio es el IDE oficial para desarrollo de aplicaciones Android. (Android D., 2017) Android Studio cuenta con grandes ventajas, entre las más destacadas, está la de recibir automáticamente las actualizaciones de nuevas APIs, como la de recibir anticipadamente las actualizaciones del SO Android. Asimismo, "Android Studio proporciona las herramientas más rápidas para la creación de aplicaciones en todos los tipos de dispositivos Android." (Android D., 2017) Android Studio permite desarrollar para todos los dispositivos Android, desde los *Smartwatch* hasta televisores inteligentes con el SO Android. (Android D., 2017)

7.3.2.3 Xamarin

En el 2011 nace una compañía de nombre Xamarin, organizada con un objetivo, que consiste en la implementación libre de la plataforma de desarrollo .Net para dispositivos Android, iOS y GNU/Linux. (Androide, 2014) La plataforma Xamarin ofrece la ventaja de desarrollar aplicaciones de Android, iOS y Windows Phone, de manera simultánea, generando aplicaciones para múltiples plataformas en una base de código en C #. Compañías como 3M y National Instruments han optado por usar Xamarin como plataforma de desarrollo.

A pesar de ofrecer la ventaja de no tener que hacer uso de Java para el desarrollo de aplicaciones en Android, Xamarin tiene la complicación de no poder desarrollar código reutilizable de una plataforma a otra en la interfaz gráfica, asimismo generando aplicaciones que ocupan mayor espacio que las aplicaciones nativas. (Androide, 2014)

7.3.2.4 Cordova

Cordova es un IDE pensado para el desarrollo de aplicaciones móviles, basado en HTML5, CSS y JavaScript. Lo que hace el IDE de Cordova es ejecutar el código HTML5, CSS y JavaScript como una plantilla en Cordova Apache, pero se puede transformar como una app móvil. Es decir, Cordova se basa del estándar web para crear la aplicación móvil, evitando que el programador tenga que desarrollar la aplicación móvil en el lenguaje nativo en el que se desea trabajar. (CORDOVA,

2017)

7.4 Redes

7.4.1 Tipos de redes

Según sea el funcionamiento, su cobertura geográfica y su configuración, se puede clasificar el tipo de red a manejar o implementar.

Los siguientes capítulos (7.4.2., 7.4.3. y 7.4.4.) se definen las redes LAN, MAN y WAN.

7.4.2 Redes LAN

Red de Área Local (LAN) (Véase la Figura 5), se trata de una interconexión de uno o varios equipos, creando una red que cubre una extensión pequeña, es una red de propiedad privada, como puede ser en una casa, oficina, universidad, etc. De esta manera genera una conexión rápida, sin inconvenientes, donde los usuarios tienen acceso a la información dentro de esta red de manera sencilla, otra característica de las LAN es que la transmisión de datos entre sí es de alta velocidad, sin embargo, las distancias son limitadas en cuanto usar una red LAN.

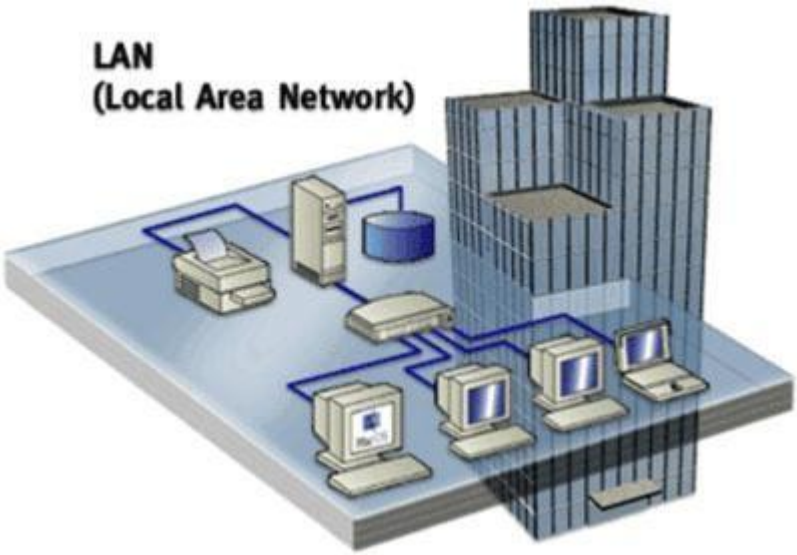


Figura 5. Red LAN (Red de Área Local).

Tomada de: http://cetis109.com/redes/redes02.html

En la configuración de una red de área local, es común tener un servidor en donde se almacena todo el software de control de la red y del mismo modo el software que comparte los datos con los demás computadores que se encuentren en la red. La mayoría de las redes LAN se conectan por medio de cables y tarjetas de red, una en cada equipo. (CISCO, 2017)

7.4.3 Redes MAN

Red de Área Metropolitana, este tipo de red en cuanto a cobertura geográfica tiene mayor alcance que las redes LAN, pero menos que las WAN, es por este motivo que estas redes se usan en ámbitos como ciudades, pueblos, etc., en la Figura 6 se puede observar un esquema de este tipo de redes. Las redes MAN se implementan normalmente por su conexión con fibra óptica, lo que genera alta velocidad y permite tener una tasa de errores y retrasos mucho más bajas que otras redes compuestas con otro tipo de conductores. La fibra óptica es un medio seguro porque la única manera de acceder a la información que transporta la fibra óptica es interrumpiéndolo físicamente.

Las redes de área metropolitana suelen ser ideales para muchos clientes de la red porque es una red de tamaño mediano, estas redes se suelen usar en empresas con varios locales, institutos como universidades con varias sedes. Además, las MAN conectan LAN cercanas entre sí a altas velocidades. (Hoy, 2015) (Research, 2015)

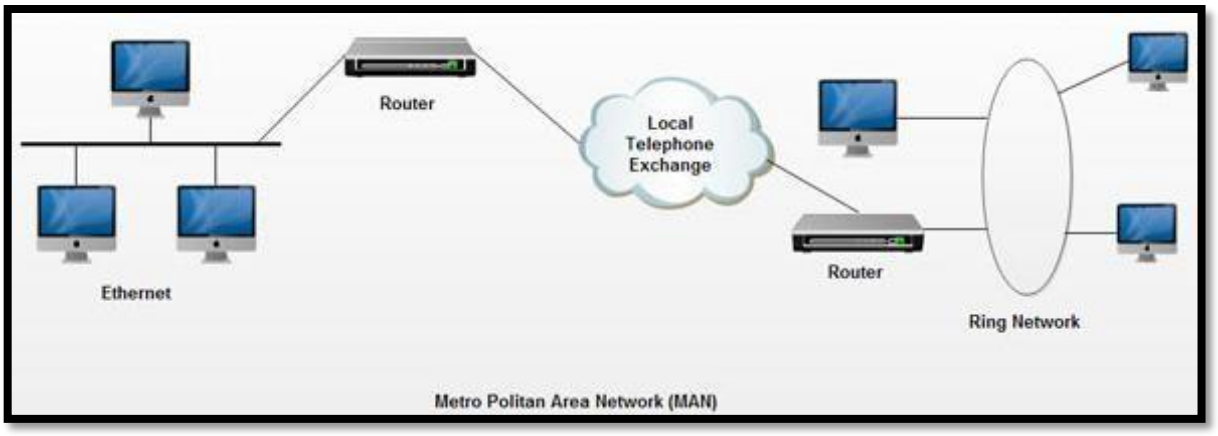


Figura 6. Red tipo MAN (Red de Área Metropolitana).

Tomada de: http://ecomputernotes.com/images/Metropolitan-Area-Network.jpg

7.4.4 Redes WAN

Red de Área Amplia, son redes que se extienden sobre un área geográfica extensa, cuando varias redes LAN se conectan entre sí mismas se conoce como una red WAN (Véase la Figura 7), estas conexiones son frecuentes en líneas telefónicas y satélites, estas redes son usadas con mayor frecuencia por grandes compañías para su propio uso (privadas), facilitando el intercambio de datos entre ellas.

Las WAN también son construidas por proveedores de servicios de Internet (ISP) para proporcionar conexiones desde las LAN de sus clientes a Internet. El Internet,

es una red mundial de redes informáticas interconectadas, por tanto, se puede definir como una red WAN, y por tal motivo es la red WAN más grande en existencia.

Las WAN son de punto a punto, esto implica que haya una conexión directa entre dos sitios o más, a través de redes de conmutación de paquetes (Véase la Figura 7), también pueden permitir a los usuarios de sitios remotos acceder e ingresar datos en una base de datos de un sitio central, como actualizar instantáneamente los registros que se desee. (Project, 2006) (CISCO, 2017)

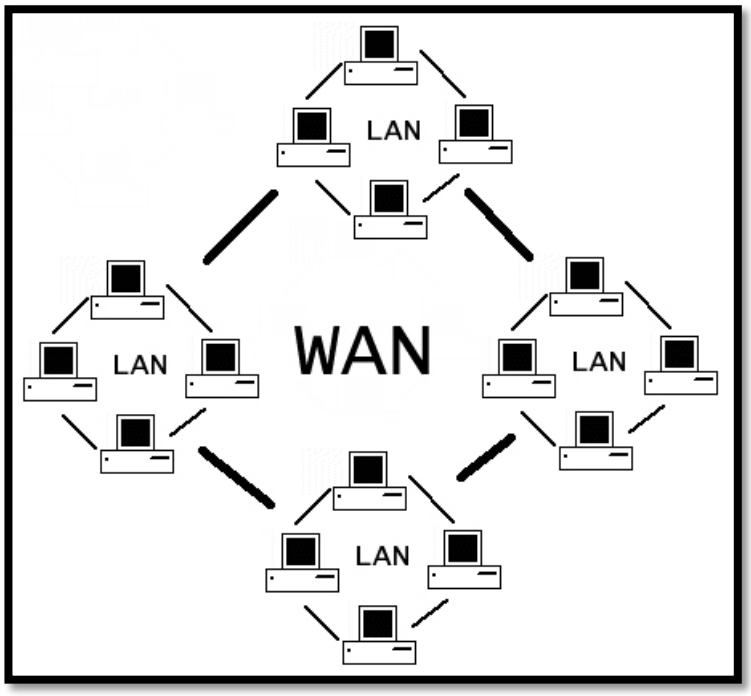


Figura 7. Red tipo WAN.

Tomada de: https://www.adslfaqs.com.ar/que-es-una-wan/.

8. DETALLES TÉCNICOS PARA LA CONEXIÓN DE LA BASE DE DATOS EN MYSQL CON JAVA

Para el desarrollo de un ejemplo práctico de bases de datos se planteó desarrollar una base de datos en MySQL, bajo un entorno que permite implementar la plataforma XAMPP, que trae consigo, el gestor de base de datos de MySQL, el Servidor Web Apache, que permite alojar la base de datos en un “localhost”, dado

que da viabilidad a un entorno necesario para desarrollar un enlace por medio de una interfaz creada en JAVA.

A continuación, en la Figura 8 se observa la apariencia del “XAMPP Control Panel”, se procede a activar Apache y MySQL. En la Figura 8 se puede ver que al frente de las palabras Apache y MySQL se encuentran unos botones con la palabra “Start”, para activar sus servicios se debe hacer clic sobre el botón, lo ideal es que en el compilador de XAMPP (la región en azul de la Figura 8) no debe aparecer ningún error (qué normalmente aparece con caracteres de color rojo). Si llega ocurrir algún error, lo más conveniente es reinstalar la herramienta XAMPP.

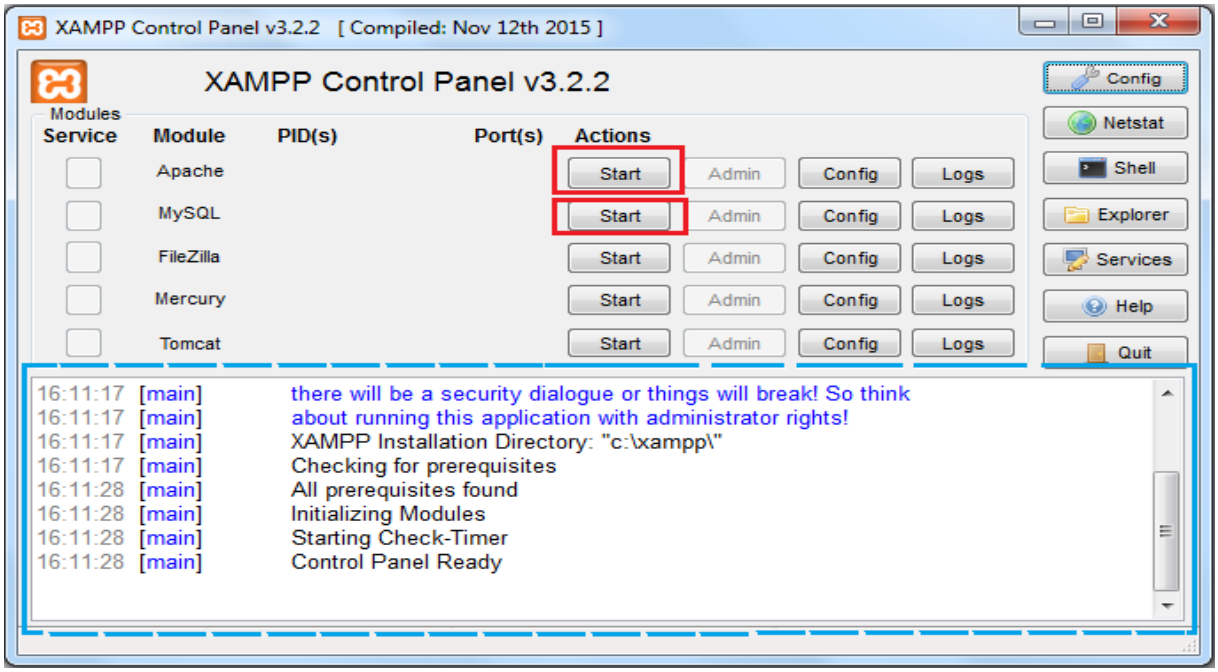


Figura 8. Interfaz de usuario de XAMPP.

Si todo está funcionando de la forma correcta, la interfaz del XAMPP debe lucir de forma parecida a la Figura 9.

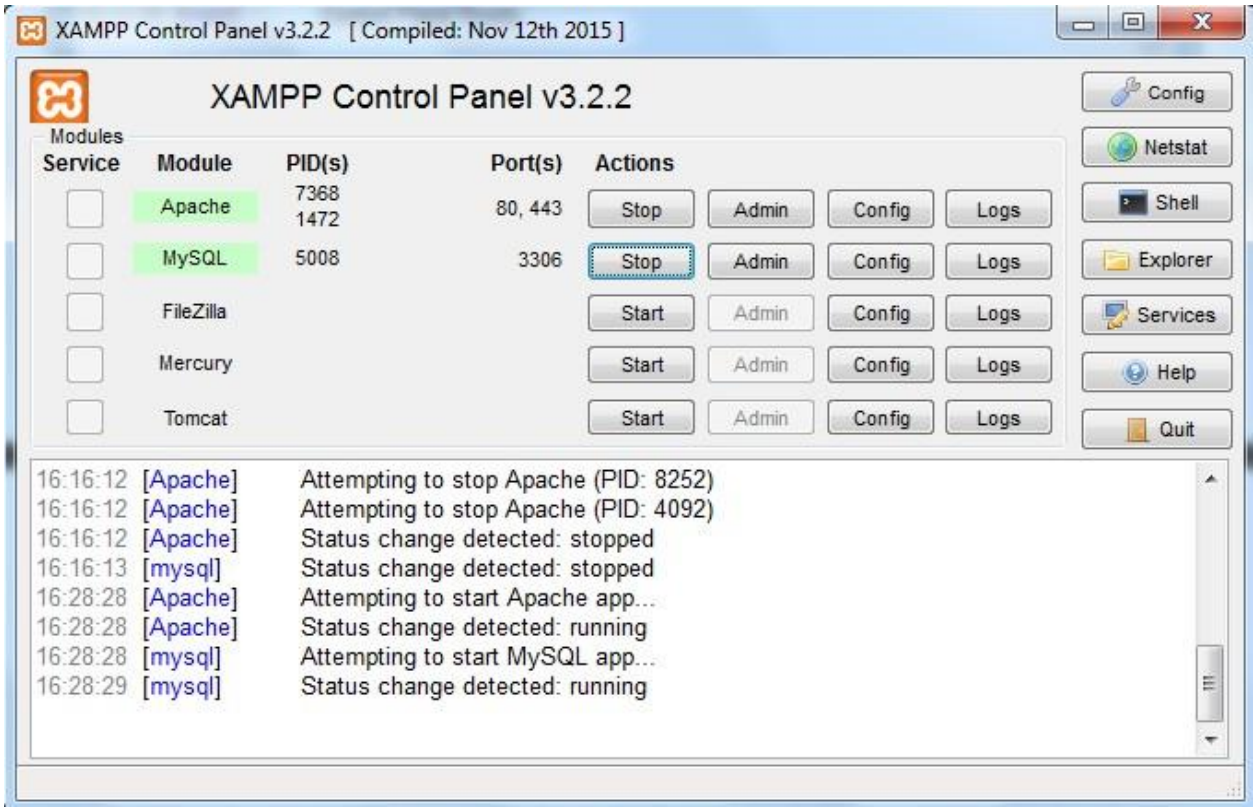


Figura 9. Interfaz del XAMPP cuando ha iniciado de forma correcta los servicios del Apache y MySQL.

Al iniciar la interfaz del Apache se debe proceder a abrir el explorador enlazado por defecto al XAMPP y abrir bajo la dirección de “http://localhost/phpmyadmin/”. Otra opción es hacer clic sobre el botón de “Admin” de MySQL mediante la interfaz del XAMPP. En la Figura 10 se puede visualizar el entorno de la interfaz de MyPhpAdmin. La plataforma XAMPP configura por defecto la herramienta MyPhpAdmin sin contraseña, también configura el nombre de usuario, que es “root”. Para mayor información sobre phpMyAdmin, revisar la cita (phpMyAdmin, 2016).

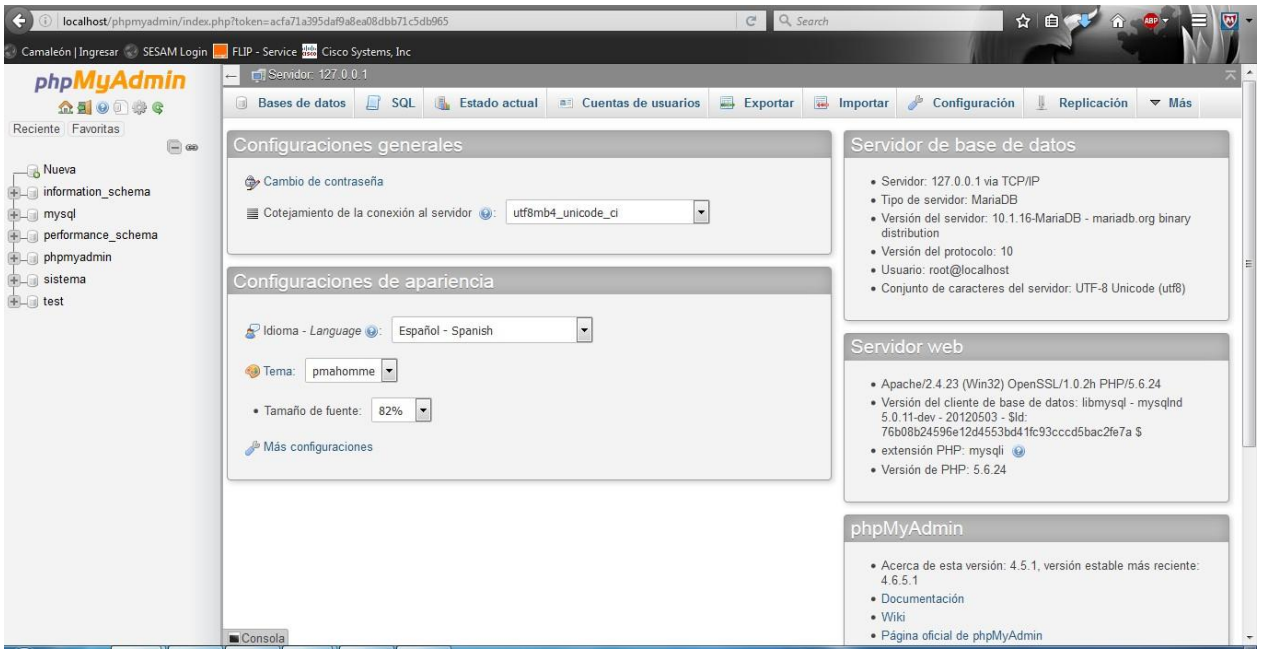


Figura 10. Interfaz inicial de phpMyAdmin.

Luego se procede a crear una nueva base de datos bajo la interfaz de PhpMyAdmin, que se muestra en la Figura 10, para ello se debe entrar a la pestaña que dice “SQL”. Luego en el espacio que aparece en pantalla (Figura 11), se escriben las sentencias “CREATE DATABASE” seguido por el nombre que desea dar a la base de datos (por ejemplo, la palabra sistema). Después se debe pulsar el botón de “Continuar”. En seguida de haber creado la base de datos, se pulsa en la pestaña de la base de datos creada y nuevamente se pulsa la pestaña con nombre “SQL” (Véase la Figura 10). En el espacio en blanco, se crea la tabla que va contener la información deseada, para crear esta tabla se escriben las siguientes sentencias:

“CREATE TABLE Nombredelatabla (
Id INT(Número de espacios a usar, por ejemplo 10) NOT NULL AUTO_INCREMENT,
Atributo1 VARCHAR(45)
Atributo2 VARCHAR(10)
...
AtributoN VARCHAR(N)
PRIMARY KEY (Id)
)”

Teniendo en cuenta que INT hace referencia a un número entero, VARCHAR a una cadena de caracteres, NOT NULL significa que el campo nunca puede tomar el valor de NULL o vacío, AUTO_INCREMENT le da la propiedad al atributo Id que se auto incremente cada vez que se ingresa un registro a la tabla. PRIMARY KEY es

una obligación de SQL que hace referencia al atributo de la tabla que será el identificador de cada fila, en este caso es el atributo “Id” que toma sus valores de forma automática, en valores tipo entero.

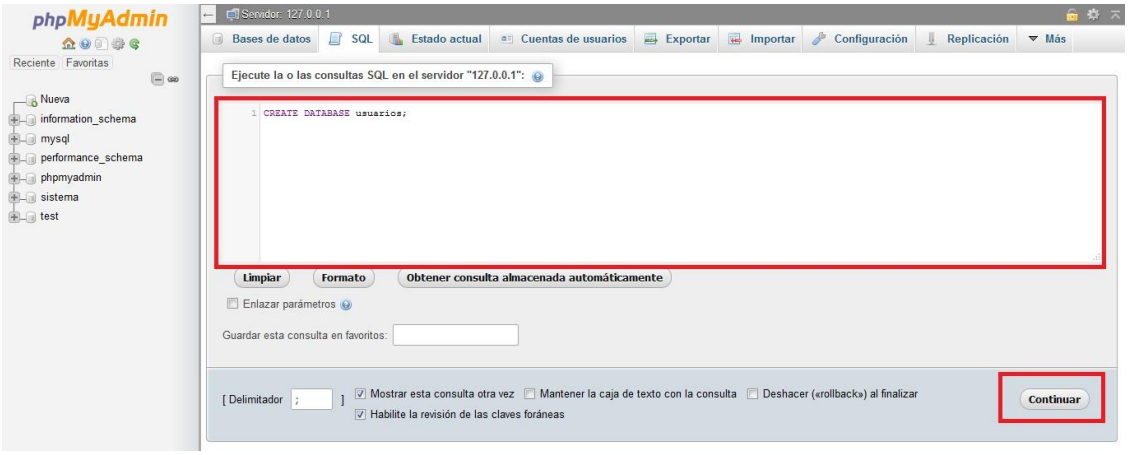


Figura 11. Creando la base de datos.

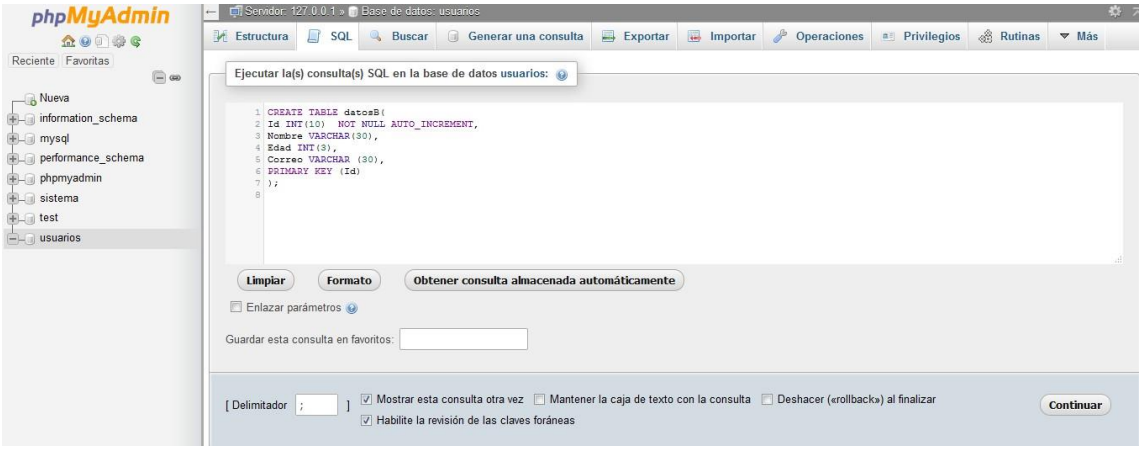


Figura 12. Creando la tabla de la base de datos.

En los **ANEXOS (Sección 11.2.1.)** de este documento se pueden ver los respectivos códigos programados por los autores del documento, allí se encuentra explicito el código que permite ejecutar la interfaz entre el usuario y la base de datos. La interfaz se compone de dos archivos, denominados JFrame1.java, ConexionBD.java.

La conexiónBD es una clase en java, que contiene el siguiente código para su método “conexion”:

```
try {
        Class.forName("com.mysql.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost/usuarios","root","");
        System.out.println("conecto");
        a1=true;
} catch(Exception e){
            System.out.println(e.getMessage());
}
return cn;
```

La variable “Connection cn = null;” expresa qué la clase en un inicio no va a tener ninguna conexión con la base de datos. La sentencia Driver Manager contiene la dirección lógica dónde está alojada la base de datos, seguido por su usuario (que para este caso es “root”) y su contraseña (que para este caso es ninguna).

En la Figura 13 se ve como agregar la librería MySQL JDBC Driver para el IDE Netbeans. Para llegar a la ventana de la Figura 13, se debe presionar clic derecho sobre el proyecto, en la opción “Properties” y después en la opción “Libraries”.

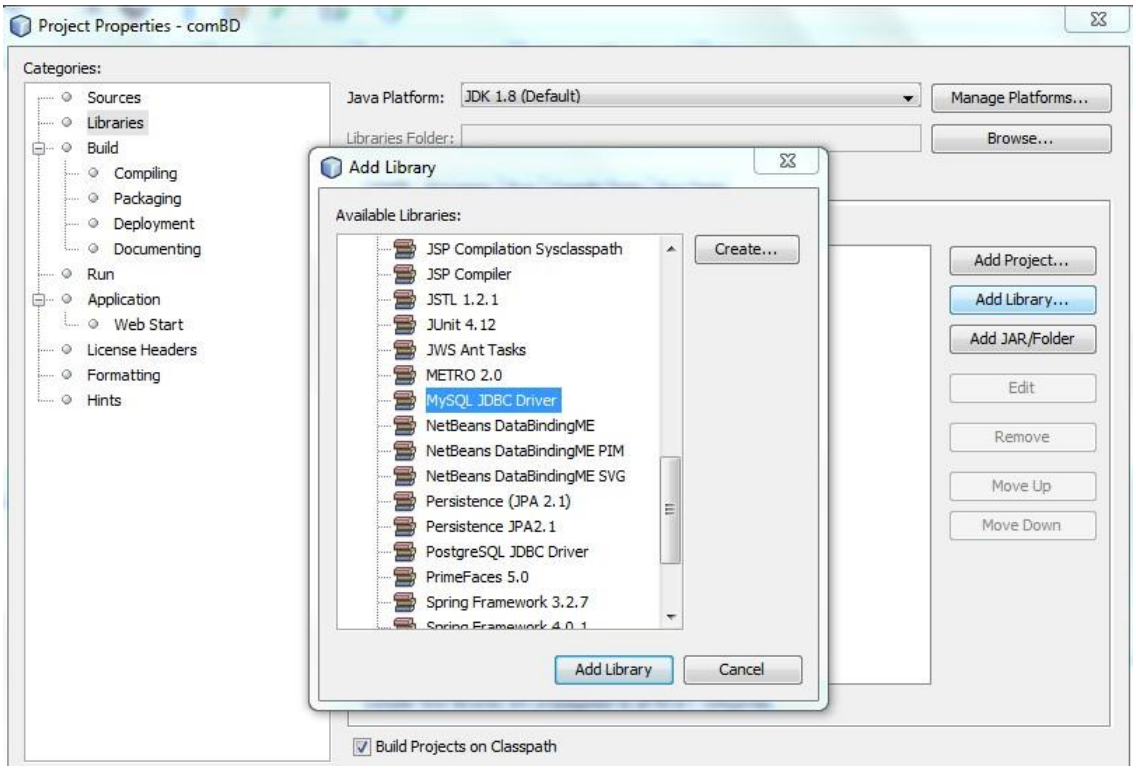


Figura 13. Agregando la librería MySQL JDBC Driver.

Luego de haber agregado la librería, se procede a crear un JFrame. Para crear un JFrame se procede a mirar la referencia (ORACLE, 2015). El JFrame de este ejemplo consta de tres “TextField” y dos “JButtons”. El “JFrame” obtenido se puede ver en la Figura 14, donde el botón conectar va a permitir crear el enlace entre el programa en Java y la base de datos en MySQL. El botón Ingresar va a permitir insertar información en la tabla de la base de datos.

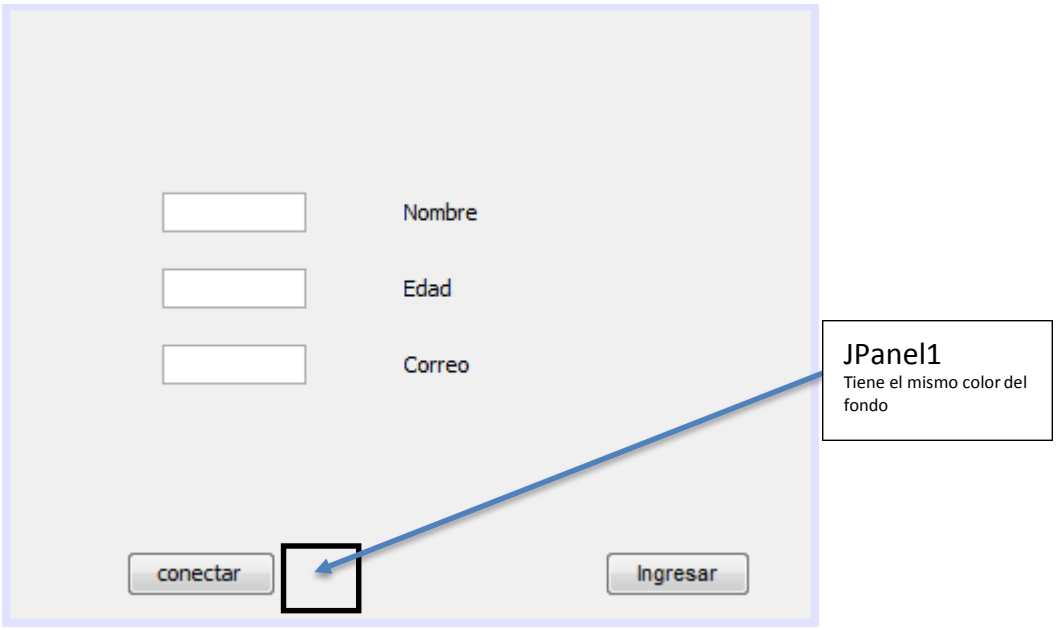


Figura 14. El panel “JFrame” desarrollado.

Entonces el botón conectar tiene la siguiente funcionalidad:

```
cn=con1.conexion();
 boolean A1=con1.geta1();
 if(A1)
 {
        jPanel1.setBackground(Color.green);

}
else
{
        jPanel1.setBackground(Color.red);

}
```

Se llama al método “conexion” de la clase “conexionBD”, que va a permitir realizar el enlace entre la base de datos y la interfaz en Java. Luego, si el resultado de la conexión es exitoso, el JFrame pinta de color verde al jPanel1, en caso contrario, lo pinta de color rojo.

El botón Ingresar tiene el siguiente código:

```
int as= Integer.parseInt(jTextField2.getText());
try {
        PreparedStatement pps = cn.prepareStatement(" INSERT INTO
                datosb(Nombre,Edad,Correo) VALUES(?,?,?)");
        pps.setString(1, jTextField1.getText());
        pps.setInt(2, as);
        pps.setString(3, jTextField3.getText());
        pps.executeUpdate();

        jPanel1.setBackground(Color.green);
}               catch               (SQLException               ex)               {
      Logger.getLogger(JFrame1.class.getName()).log(Level.SEVERE, null, ex);
}
```

La funcionalidad del código de ingresar, depende de un “PreparedStatement”, que es un objeto que representa una instrucción SQL. El “PreparedStatement” consta de dos variables tipo String y una de tipo entero, sin embargo, se deben enviar en orden según aparecen la sentencia SQL del objeto, para el caso en específico se debe enviar una variable de tipo String (Nombre), seguido por la variable tipo entero (Edad), seguido por la variable tipo String (Correo). Pues la sentencia real de SQL es: “INSERT INTO datosb (Nombre,Edad,Correo) VALUES(?,?,?)”.

Ulteriormente de haber creado el código, cada vez que el usuario ingrese a la interfaz desarrollada en java, podrá insertar valores a la tabla de la base de datos.

Después de todos los pasos nombrados, se puede visualizar la tabla creada en la base de datos, y sus registros mediante la interfaz de MyPhpAdmin.

9. DISEÑO METODOLÓGICO

Debido a que la información existente acerca del servicio sobre una base de datos está enfocada hacia el campo informático para la preservación, organización, publicación y distribución de los datos en diferentes ámbitos de programación, pero los datos ya están recolectados, es una información puntual de los equipos que contienen Orange Business Services. No es necesario realizar una exploración que implique una recolección de datos. Para el diseño metodológico se aplicará los pasos definidos en el anteproyecto del presente proyecto. A continuación se muestran las directrices de la metodología que se aplicó para ejecutar el proyecto, que fue aprobado por el comité de grado.

Nombre de la directriz metodológica	Objetivo que cumple
10.1. Levantamiento de requerimientos	Describir el proceso del manejo de inventario que se desea sistematizar para poder extraer los requerimientos funcionales del prototipo a implementar.
10.2 Definición de casos de uso	Definir los componentes del sistema con sus funcionalidades para planear el desarrollo del prototipo y medir el avance del desarrollo.
10.3. Diseño del modelo de datos	Diseñar el modelo de datos del sistema basado en los requerimientos para garantizar la disponibilidad de la información en el sistema a implementar.
10.4. Diseño de las interfaces de usuario	Desarrollar una interfaz en java para dar viabilidad de manejar la base de datos a múltiples usuarios que desconocen el manejo de bases de datos en MySQL.
10.5. Diseño de la infraestructura de comunicación necesaria (Redes)	Diseñar la infraestructura de redes necesaria para acceder la información del sistema desde un dispositivo móvil.
10.5.2.1 Diseño de la Topología y simulación	Simular la infraestructura diseñada para emplearla como canal de comunicación entre un dispositivo móvil y la información de inventario.
10.7. Desarrollo de los servicios web	Desarrollar los servicios web necesarios para soportar los requerimientos extraídos y sirva como interface entre los dispositivos móviles y la información en la base de datos.

10.8. Desarrollo de la aplicación de Android	Desarrollar una aplicación en un dispositivo móvil que consuma los servicios web desarrollados para que el sistema conformado cumpla con los requerimientos extraídos.

Tabla 1. Resumen de objetivos desarrollados en la ejecución del proyecto.

10. EJECUCIÓN

En el presente capitulo se presenta la ejecución de cada directriz metodológica. En cada subcapítulo se describe todo el proceso que se desarrolló, se muestra tanto en los conceptos de diseño como en los detalles técnicos. De este modo se culminó el desarrollo del sistema.

10.1 Levantamiento de requerimientos

TABLA DE LEVANTAMIENTO DE REQUERIMIENTOS	
PERFIL DEL USUARIO AFECTADO	
Nombre de la compañía	Orange Business Services
Puesto	Practicante
Principales responsabilidades	Tarea 1: Limpieza Lógica y Física de Routers y Switches. Tarea 2: Entrega de equipos a clientes. Tarea 3: Registrar la entrada y salida de equipos en el punto WTC. Tarea 4: Completar la base de datos de soporte técnico, instalación y desinstalación de los ingenieros de la empresa (Trilliums). Tarea 5: Inventario en bodega nueva y antigua.
EVALUACIÓN DEL PROBLEMA	
Problemas identificados	Existe un problema en el manejo de inventarios, el manejo normal impide que se pueda manipular el inventario en unos computadores distintos a los asignados para los practicantes.
¿Por qué existe este problema?	Porque el inventario se maneja con un formulario simple de Excel, este se encuentra en una carpeta compartida en la red de la empresa
¿Cómo resolver el problema?	Se propone diseñar un programa que pueda compartir al inventario como una tabla de una base de datos, sin embargo, esta base de datos estará alojada en un equipo y los demás deberán conectarse al equipo para no exponer la información que contiene la base de datos.

AMBIENTE Y NATURALEZA DEL USUARIO	
¿Quiénes son los usuarios?	Practicantes / Jefe de operaciones
Nivel educativo del usuario	Universitario
¿Qué plataformas se utilizan para estas aplicaciones?	Office Excel y plataforma web de Camaleon.
Expectativas con respecto a la facilidad de uso de esta aplicación	Poder modificar el inventario en distintos equipos y poder visualizarlos en los dispositivos móviles.

Tabla 2.Tabla de levantamiento de requerimientos.

Para el desarrollo del levantamiento de requerimientos, es necesario explicar las principales tareas que desarrollan los practicantes. En la tabla de requerimientos

La intención principal del prototipo desarrollado, es facilitar el registro de los equipos en el inventario y mostrar el mismo en distintos casos. Porque, como está explicitó en la problemática, el sistema busca facilitar el acceso y modificación de la información del inventario, a pesar de sus limitaciones en el sistema que se maneja actualmente en Orange Business Services. Dado ello, el sistema afectará directamente el desarrollo de la tarea 3 de la lista anteriormente mostrada. En seguida se explica con detalle el desarrollo de la tarea 3.

10.1.1. Explicación de la tarea de registrar la entrada y salida de equipos en el punto WTC

Orange Business Services Colombia S.A (OBS), como comercializadora de equipos y servicios en el área de telecomunicaciones, tiene un esquema muy detallado para controlar la manipulación de cualquier equipo. Las actas de entrada y de salida de equipos de la empresa son sustanciales ya que permiten el orden y control de los equipos que están disponibles en las bodegas, gracias a la ayuda de la plataforma Camaleón para la generación de actas correspondientes. Una de las actividades de los practicantes es recibir los equipos que lleguen al punto WTC, del mismo modo verificar su correspondiente serial y orden de compra, revisar el estado de los mismos y verificar las características del equipo que está llegando en físico al punto y estar atento que coincidan con las características estipuladas en los documentos internos de la empresa y de aquellos quienes transportan los equipos. Al recibir en buenas condiciones toda la carga se procede a realizar su correspondiente acta de entrada de la misma orden de compra y posteriormente se ubican en bodega, de igual manera hay que actualizar los datos de la bodega en una base de datos que se tiene estipulada en una carpeta compartida para el área.

Para la salida de equipos el procedimiento es similar, se realiza una búsqueda de la respectiva orden de compra en el sistema y posteriormente se busca el equipo

en físico, verificando las especificaciones, se elabora un acta de salida con las características del equipo. Por último, el equipo se permite entregar de varias maneras, puede ser entregado al ingeniero quien realizará la instalación correspondiente en donde se ubique el cliente, puede ser entregado directamente al cliente o se puede enviar a diferentes partes de Colombia por medio de DHL, solicitando el servicio.

10.2. Definición de casos de uso.

En el sistema se identifican dos actores, los practicantes y en si la base de datos. A continuación se muestran los principales casos de uso por los practicantes, se puede ver en la Figura 15, en la figura anteriormente nombrada, se puede identificar a los practicantes como el dibujo de la persona y la base de datos es el dibujo que está a la derecha. Los casos se basan en la problemática hallada en el inventario, de este modo, se realiza una abstracción de las tareas que realizan los practicantes con respecto al inventario y se resumen en tres acciones principales, la primera es ingresar los equipos, la segunda es registrar el egreso de los equipos y por último la ubicación o reubicación de los equipos. Las acciones anteriormente nombradas en este párrafo constituyen los casos de uso del usuario de la aplicación de Java para escritorio, esto se puede ver en la Figura15.

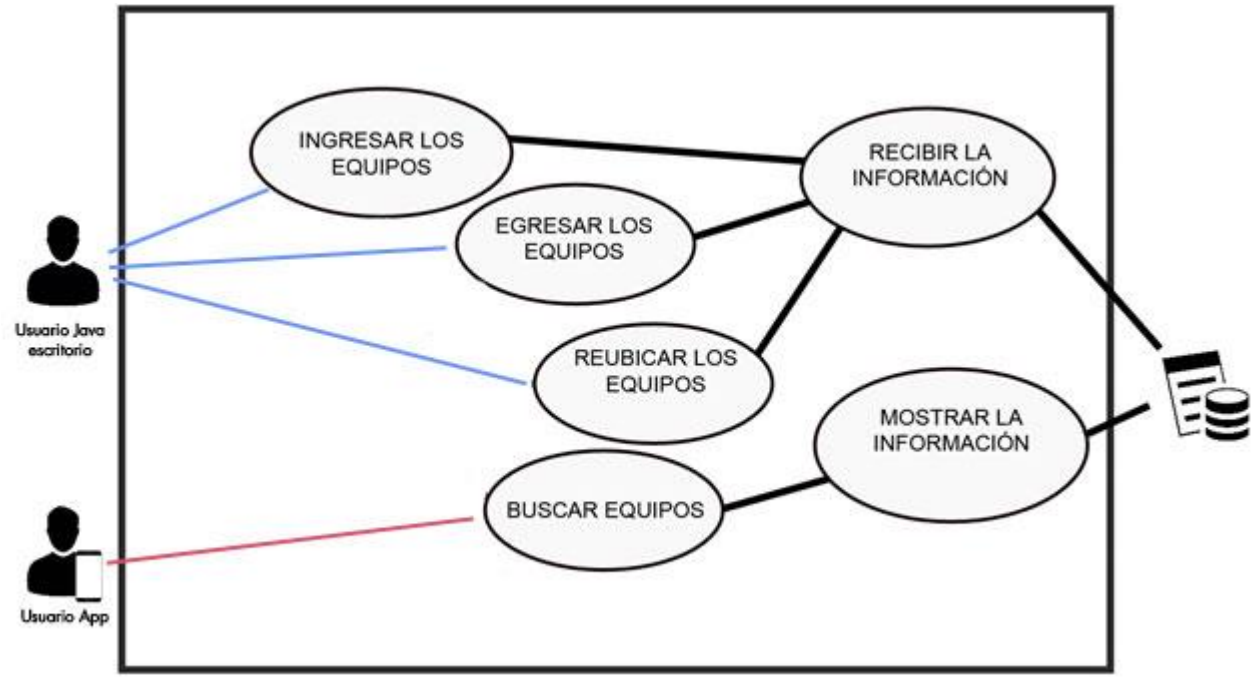


Figura 15. Casos de uso del prototipo desarrollado.

En la Figura 15 se puede observar que los principales casos de uso son:

- Ingresando los equipos.

- Egresando equipos, que sería lo mismo de mover en los inventarios,
- Reubicando los equipos.
- Buscando los equipos en el inventario.

Para el caso de la aplicación móvil, se limita el caso de uso de la base de datos, por dos motivos, el primero es porque es necesario formalizar la modificación de la base de datos por una oficina de Orange Business Services, pues en un móvil da la libertad de modificarlo en cualquier parte en que se encuentren los actores. El segundo motivo es que el móvil puede generar una gran vulnerabilidad a la información del sistema, por ello se opta que sólo se puede acceder a mostrar la información del inventario más no modificarla. Además, la aplicación móvil es pensada para facilitar el acceso a la información a distintos actores que en sí no modifican nunca el sistema pero les convendría acceder a la información, por ejemplo, a la jefe de operaciones, a los vendedores de la empresa e incluso a los técnicos de la misma, para que la herramienta sea parte de la comunicación interna de la empresa.

La arquitectura del sistema que se va a implementar, consta de dos elementos, uno es una aplicación en JAVA y el otro es la programación nativa del SO Android.

Las funcionalidades del programa en JAVA son:

- Conectarse a la base de datos (que contiene el inventario) por medio de una aplicación en JAVA.
- Poder ingresar un nuevo equipo al inventario.
- Poder borrar un equipo del inventario.
- Poder ubicar un equipo en el inventario.

Estas funcionalidades de la aplicación en JAVA para escritorio, son fundamentadas en la problemática vista en OBS (véase capítulo sobre problemática), se requiere poder ubicar el equipo según sea su orden de compra y serial. Dado esto, se genera la necesidad de tres funcionalidades, que son el ingreso, ubicación y capacidad de borrar del inventario cualquier equipo. Es importante detallar, que el manejo del inventario en OBS, es aparte del manejo logístico del mismo, pues en OBS, la herramienta logística por defecto es Camaleón. Por ello las funcionalidades del sistema se definen en el manejo exclusivo del inventario.

Teniendo en cuenta lo dicho en el anterior párrafo, las funcionalidades para la aplicación en Android se definen aún más sencillas que la aplicación para escritorio de JAVA, pues no es recomendable definir una funcionalidad de ingreso o de borrar del inventario un equipo, desde un dispositivo móvil, tiene la única necesidad de ubicar un equipo en el inventario, porque para el ingreso o egreso de un equipo, primero debe pasar por la aprobación física del director de operaciones de OBS,

mientras que la necesidad principal de la aplicación móvil, es poder ubicar un equipo del inventario, para solicitar el mismo por medio de los protocolos requeridos en OBS. Entonces, basándose en la necesidad principal de la aplicación móvil y en la necesidad de respetar los protocolos requeridos en OBS (dando no viabilidad al usuario de saltarse el protocolo), se definen estas funcionalidades de la aplicación móvil:

- Poder ubicar un equipo por medio del dispositivo móvil.

10.3. Diseño del modelo de datos

El diseño del modelo entidad – relación es bastante sencillo, porque en el presente proyecto contiene sólo una tabla, dicha tabla está compuesta por valores sencillos (conocidos como valores atómicos), dichos valores no tienen relación con otros atributos. En la Figura 16 se puede observar la única tabla, que en su modelo de entidad-relación cuenta con atributos atómicos. (Véase debajo de la flecha de la figura 16).

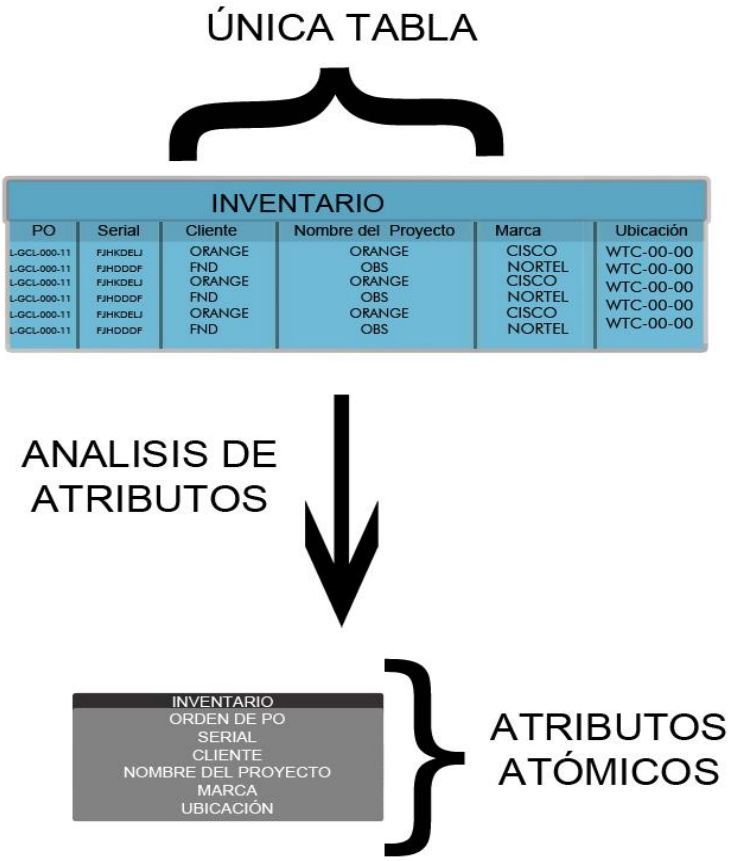


Figura 16. Análisis del modelo de la base de datos.

10.4 . Diseño de las interfaces de usuario

Se diseñó una interfaz, que tenga solo una ventana para la aplicación de escritorio, en la misma ventana el usuario puede ingresar o buscar datos, en la Figura 17 se puede observar la interfaz gráfica diseñada, en la misma figura se puede ver que existe una subsección de la venta, cuyo nombre es “Resultados de búsqueda”, en ella se pueden ver unas listas, estas listas deben mostrar los resultados de la respuesta lógica de la búsqueda de la base de datos. Estas listas se muestran de manera eficiente en comparación a las listas convencionales que ofrecen los JFrame de java, porque, en el presente trabajo se colocó con tablas que ofrecen la capacidad de redimensionar el espacio entre columnas y mostrar de manera ordenada las filas. La tabla de la que se habla se puede ver en la parte izquierda de la Figura 17. En la misma se puede observar que en la parte derecha, hay un espacio para el ingreso de equipos o en su defecto para la búsqueda del equipo.

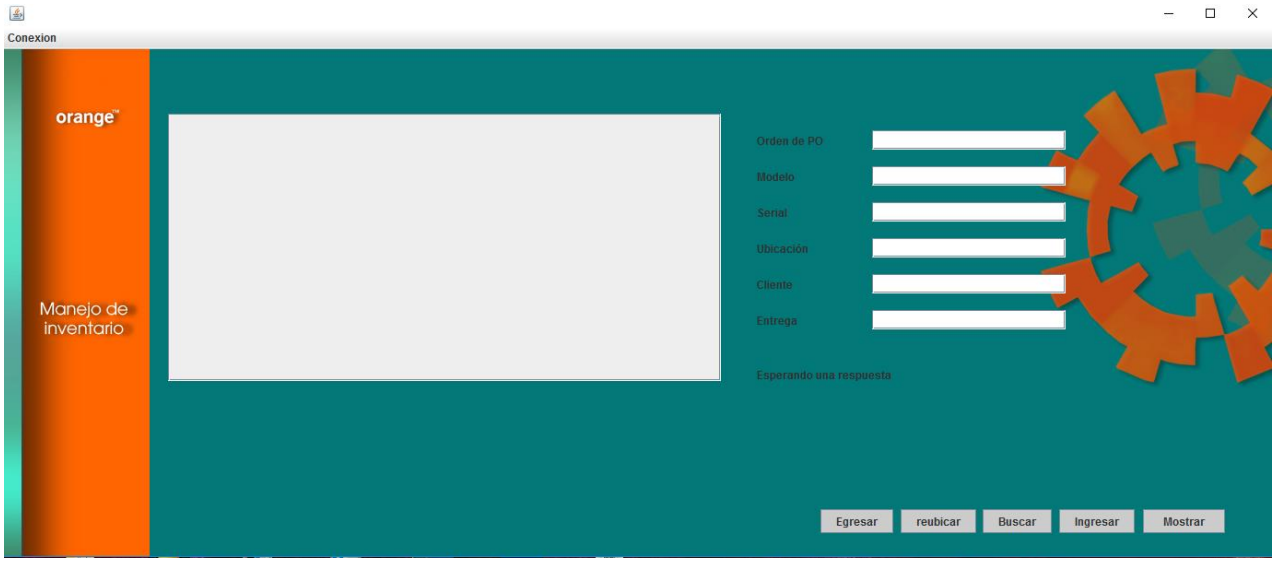


Figura 17. Diseño de la interfaz gráfica a desarrollar.

Para la aplicación móvil, se planeó desarrollar una interfaz gráfica similar, sin embargo se divide en dos ventanas (o Layaouts), una ventana muestra los resultados de la búsqueda y la otra podrá solicitar ingresar a los atributos que se desean buscar. La ventana o Layaout se puede observar en la Figura 18. En la parte izquierda de la figura 18, está la sección principal. A la derecha de la figura 18, está la sección de presentación de datos.

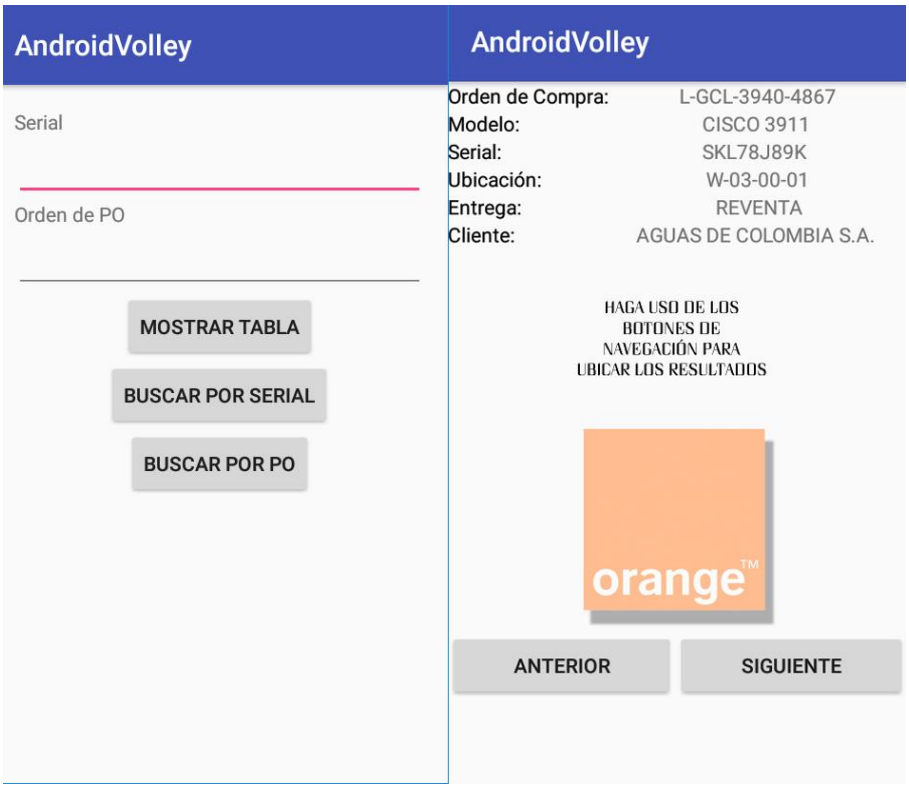


Figura 18. Interfaz para la aplicación móvil.

10.5. Diseño de la infraestructura de comunicación necesaria (Redes)

10.5.1 Diseño lógico de Red LAN.

Las redes LAN posibilitan que viaje todo flujo de información y con ello la integridad de datos. El diseño lógico se realizó bajo las necesidades que tenía el sistema, una necesidad fundamental es compartir los datos por medio de la red, lo cual permite y facilita la comunicación entre usuarios, sin descuidar la seguridad de la misma.

10.5.1.1 Diseño de la Topología

La topología de red es la estructura lógica donde están distribuidos y conectados los equipos que pertenecen a la red. El objetivo de la topología es explorar la manera más sencilla y eficaz de conectar todos los equipos, brindando facilidad de control y acceso a la red.
El diseño de la topología LAN se divide en tres capas del modelo OSI, como se observa en la Figura 19. En la topología de la red (Figura 19), se pueden detallar los distintos equipos electrónicos (Router, Switch, Hosts), que son usados para realizar la conexión de la Red LAN que se desarrolló en el proyecto, añadiendo el AccessPoint para que

haya conectividad con el dispositivo móvil.

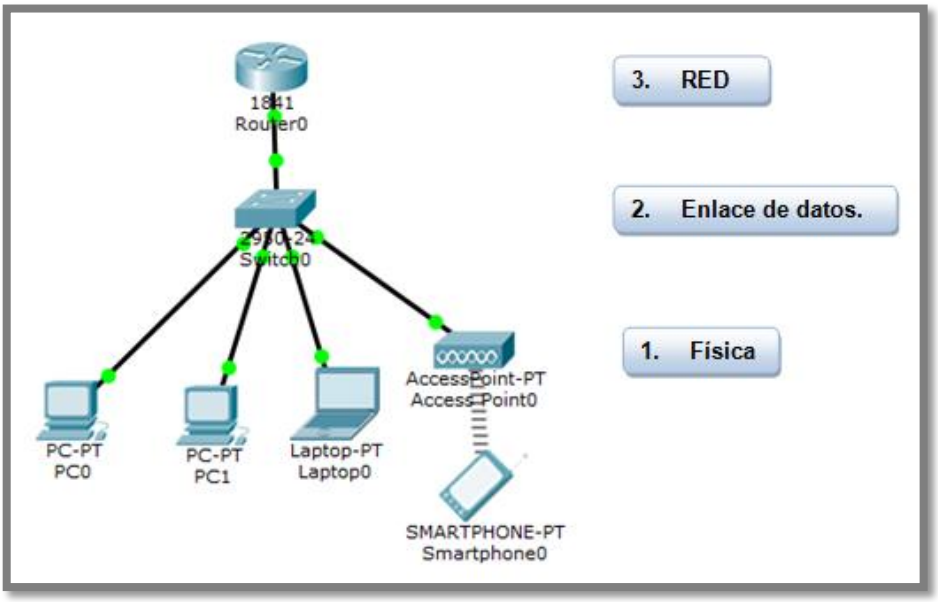


Figura 19. Topología de la Red LAN (Capas OSI).

Para realizar la conexión entre estos hosts, es necesaria la configuración de un Router, el cual está conectado a un Switch y simultáneamente este se conecta a los hosts que sean necesarios, y al AccesPoint para la conexión del dispositivo móvil.

10.5.1.2 Desarrollo

El desarrollo de la red LAN que se implementó, se creó con distintos equipos, se muestran en la siguiente tabla, cada uno de los equipos fue conectado por medio de cables Ethernet. Para la realización de la red LAN, es necesario la configuración de un Router, haciendo uso del protocolo DHCP (Dynamic Host Control Protocol).

EQUIPOS	CANTIDAD
ROUTER CIS-CISCO1841	1
SWITCH CISCO-WS-C2960-24TT-L	1
Host(ASUS)	2
AccesPoint (Point TP-LINK TL-WA850RE)	1

Fue necesario la configuración del Router, la configuración se realizó en el entorno CLI por medio del puerto de consola y haciendo uso del programa de emulación PuTTY (Figura 20). Al momento de encender el Router, se puede detallar el arranque del mismo, para realizar cualquier modificación es necesario pasar al modo privilegiado con

el comando “enable”, en modo privilegiado, se ingresa a “configure terminal”, para después ingresar la interface que se desea activar para nuestro caso “interface fastethernet 0/0”, se le ingresa su correspondiente dirección IP(192.168.10.1) y mascara de red (255.255.255.0), y para realizar la activación de la interfaz 0028f0/0) se ingresa el comando “no shutdown”.

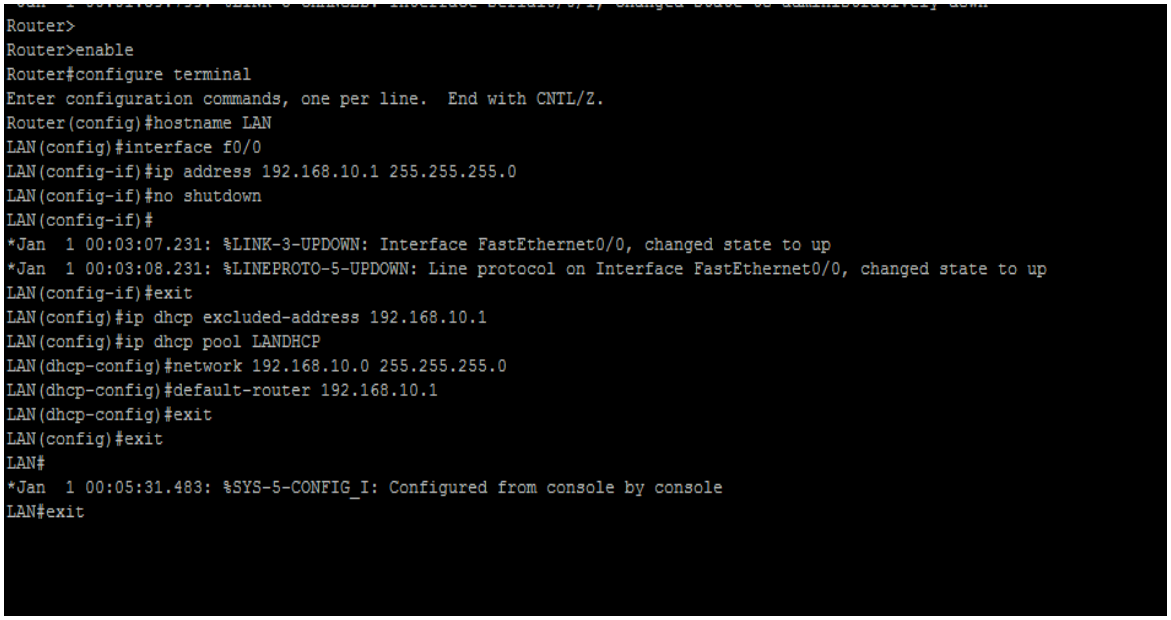


Figura 20. Configuración del Router.

10.5.1.3 Direccionamiento IP

Para el direccionamiento de IP, se definió el medio más adecuado, eficaz y cómodo es el direccionamiento dinámico, este protocolo DHCP (Dynamic Host Control Protocol) permite configurar la IP de los dispositivos finales de una manera automática de direcciones IPv4 (cuatro números decimales entre 0 y 255, asignados a los dispositivos conectados a una red).

Para la configuración y activación del DHCP en el Router es necesario excluir la dirección IP que se le asignó al Router, esto se hace solo como prevención debido que en el momento de asignación de IP, no se encuentre en conflicto o cruce con la IP del Router. Se procede a la creación del pool de direcciones de DHCP donde se le puede asignar un nombre. Luego de haber configurado el pool de direcciones de DHCP se debe colocar el Router en modo configuración pool DHCP, se puede comprobar que se está en la configuración observando el principio del nombre del Router (Router (config-dhcp)#), este parámetro permitirá ingresar la dirección IP de la red, con el comando “network 192.168.10.0 255.255.255.0” y para culminar con el proceso se configura la puerta de enlace o Gateway, con el comando “default-router 192.168.10.1”.

10.5.1.4 Demostración de conectividad.

Después de haber realizado toda la configuración del Router, se conectan todos los equipos, y se procede a realizar una prueba de conectividad. Para realizar la prueba, se hace uso de la herramienta ping desde la consola cmd.exe. En la Figura 21 se observa la primera verificación de conectividad, realizando ping desde el primer host con IP 192.168.10.3 asignada automáticamente por el protocolo DHCP, al segundo host conectado a la red con una IP 192.168.10.4, igualmente asignada automáticamente, y en la Figura 22 se puede detallar el mismo proceso pero inverso, haciendo ping del segundo host al primero.

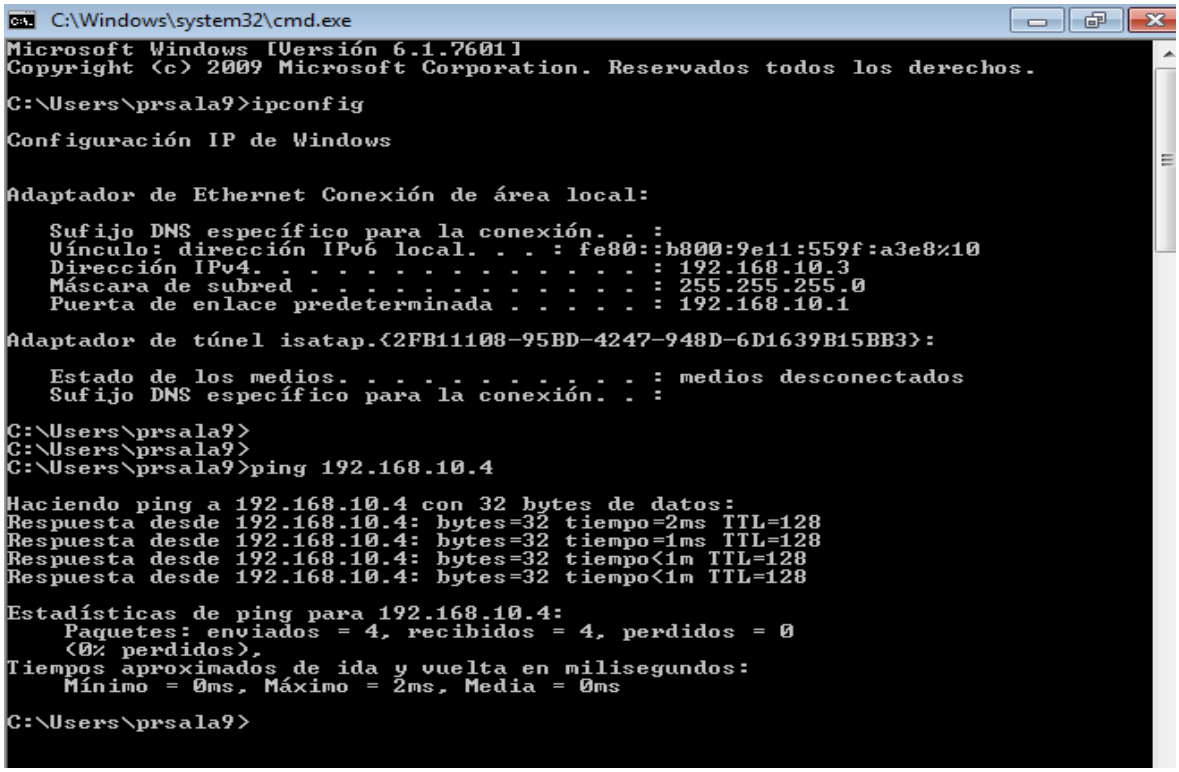


Figura 21. Verificación de conectividad (Ping 192.168.10.4).

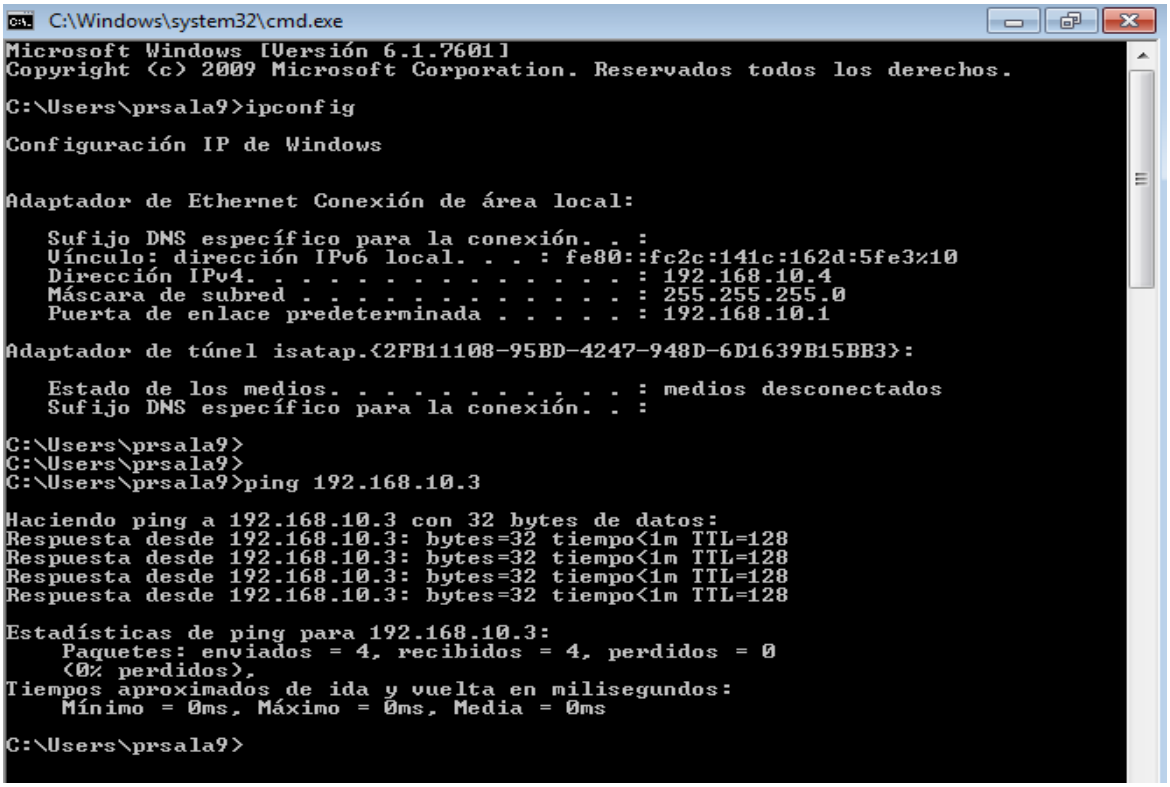


Figura 22. Verificación de conectividad (Ping 192.168.10.3).

El comando ping envía cuatro paquetes de prueba de 32 bytes desde el host emisor, y estos cuatro paquetes tardan un tiempo de 1 ms en devolverse. Esta verificación por el comando ping lo que hace es comprobar que está en funcionamiento correcto la tarjeta de interfaz de red y la implementación del suite de protocolos TCP/IP.

10.5.2 Diseño lógico de Red WAN

Las redes WAN, son muy usadas por las grandes empresas o compañías, este tipo de red permite establecer comunicación entre diferentes lugares de grandes distancias geográficas, como lo puede ser en el caso de varias sedes de la empresa, en una misma ciudad o en diferentes ciudades. Por tal motivo, de ser una red a grande escala, solo se desarrolló la correspondiente simulación de la red WAN, compuesta por las dos sedes en Bogotá.

Es necesario recordar que esta red WAN se simula únicamente en el programa del Cisco Packet Tracer, porque la compañía Orange Business Services no permite ninguna modificación de su red por causas de seguridad, entonces, para cumplir con el requisito de diseño de redes, se simula la red WAN de Bogotá.

10.5.2.1 Diseño de la Topología y simulación.

Para realizar el diseño, se procedió a desarrollar la simulación correspondiente a dos de las sedes que se encuentran en Bogotá. Se implementa la topología punto a punto, esta consta de un enlace permanente entre dos terminales (Routers), los Routers son los puntos de conexión dentro de una red WAN, determinando la ruta más adecuada para el flujo de datos que se requiera, en la Figura 23 se puede detallar la topología de la red WAN.

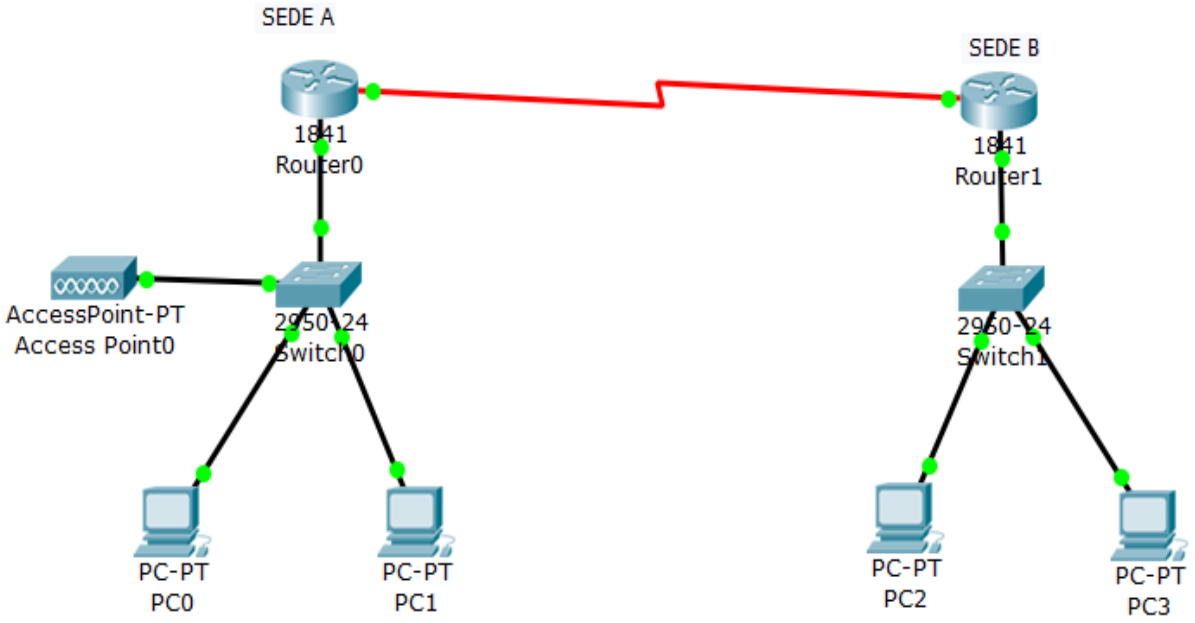


Figura 23.Simulacion red WAN.

En la Figura 23 se observa la simulación por medio de la herramienta Cisco Packet Tracer. En cuanto la configuración del Router, se usan los mismos comandos que se usaron para realizar la configuración de red de área local (LAN), agregando unos comandos adicionales, para la activación y uso del puerto serial (se0/1/0), para la correspondiente conexión de Router a Router, de esta manera se satisface la comunicación correcta de la red WAN, en la Figura 24 se puede observar la correspondiente configuración de uno de los Routers.

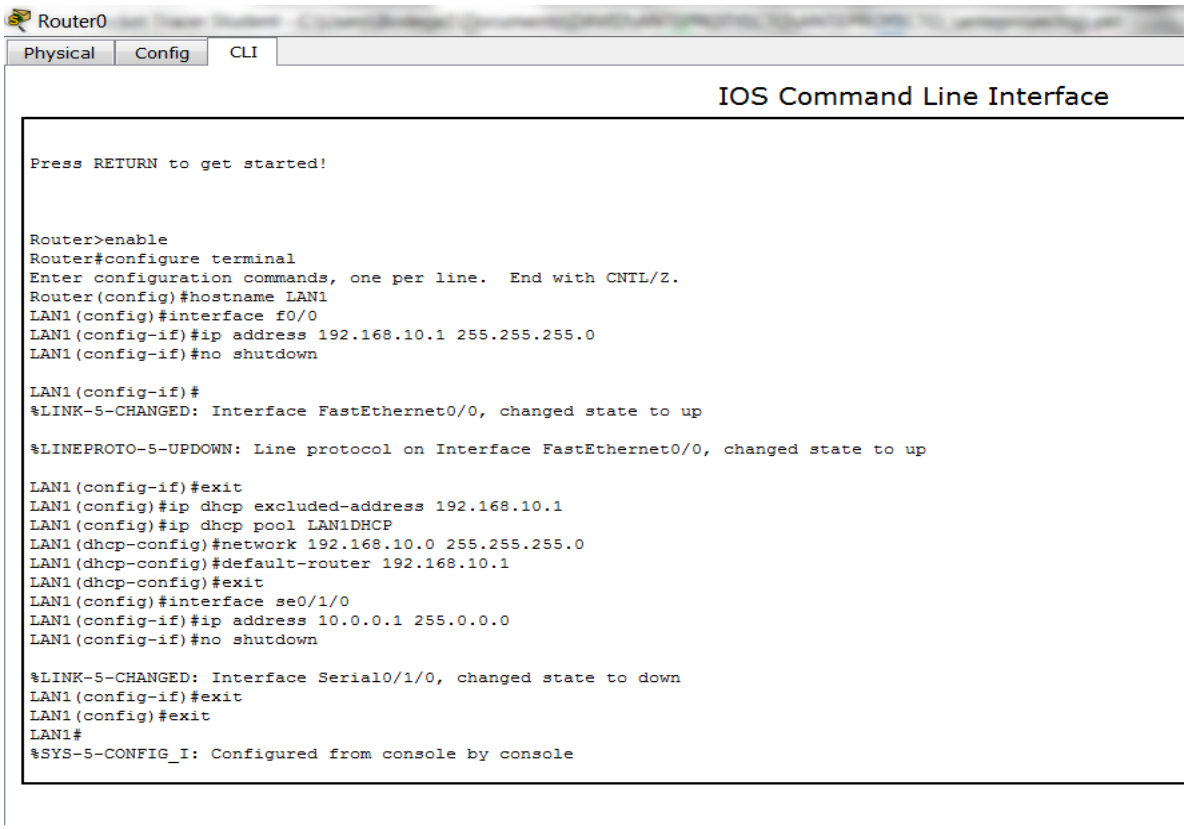


Figura 24. Configuración del Router1.

Para comprobar que el puerto serial que se utilizara se encuentra activo es necesario realizar la misma configuración en el segundo Router, para activar el mismo puerto (se0/1/0) y de este modo se pueda establecer la conexión. En la Figura 25 se puede observar la configuración del segundo Router y se puede detallar que después de ingresar el comando “Interface se0/1/0”, luego se activa el puerto con el comando “no shutdown”, en la Figura 25 se observa que el estado del puerto cambia a modo “up. De este modo, la conexión punto a punto estará establecida y se puede contar con la configuración y conexión exitosa de una red WAN.

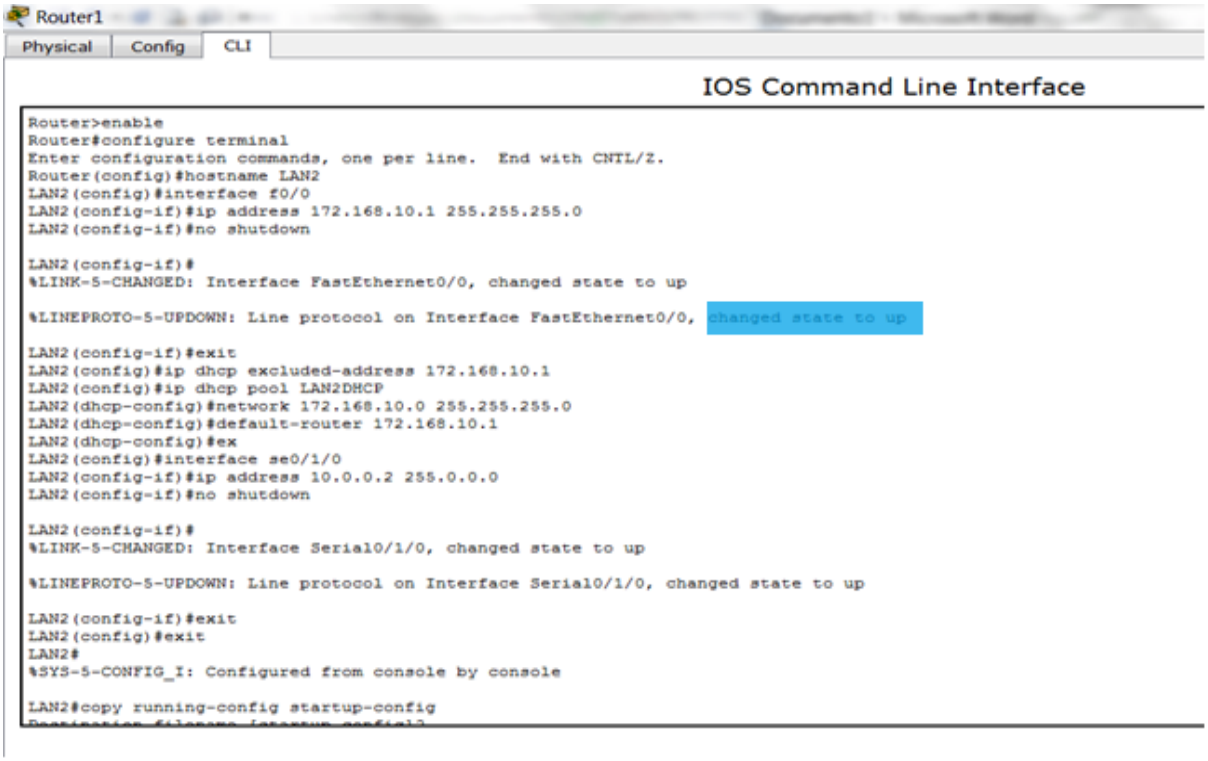


Figura 25. Configuración del Router2.

Para terminar con la correcta conexión de la red WAN, es necesario hacer uso del protocolo Routing Information Protocol (RIP), este es un protocolo de enrutamiento usado por distintos Routers para intercambiar información y de este modo conocer por cual camino se debería enrutar el paquete para que llegue a su destino. Esta configuración del protocolo RIP se puede observar en la Figura 26, es necesario introducir el comando “router rip”, luego utilizar el comando “network”, seguido de la red que se desea declarar en el protocolo, este mismo proceso se realiza en el otro Router. De este modo, habrá conexión entre ambos Routers y hosts de cada red LAN.

Figura 26.Configuracion RIP(Routing Information Protocol).

Para finalizar, se realiza las pruebas funcionales que nos facilita la herramienta de Cisco Packet Tracer, con su modo de simulación, se puede enviar un mensaje de un host de la red LAN1 (PC0) a otro host de la red LAN2 (PC3), verificando y validando su correcto funcionamiento.

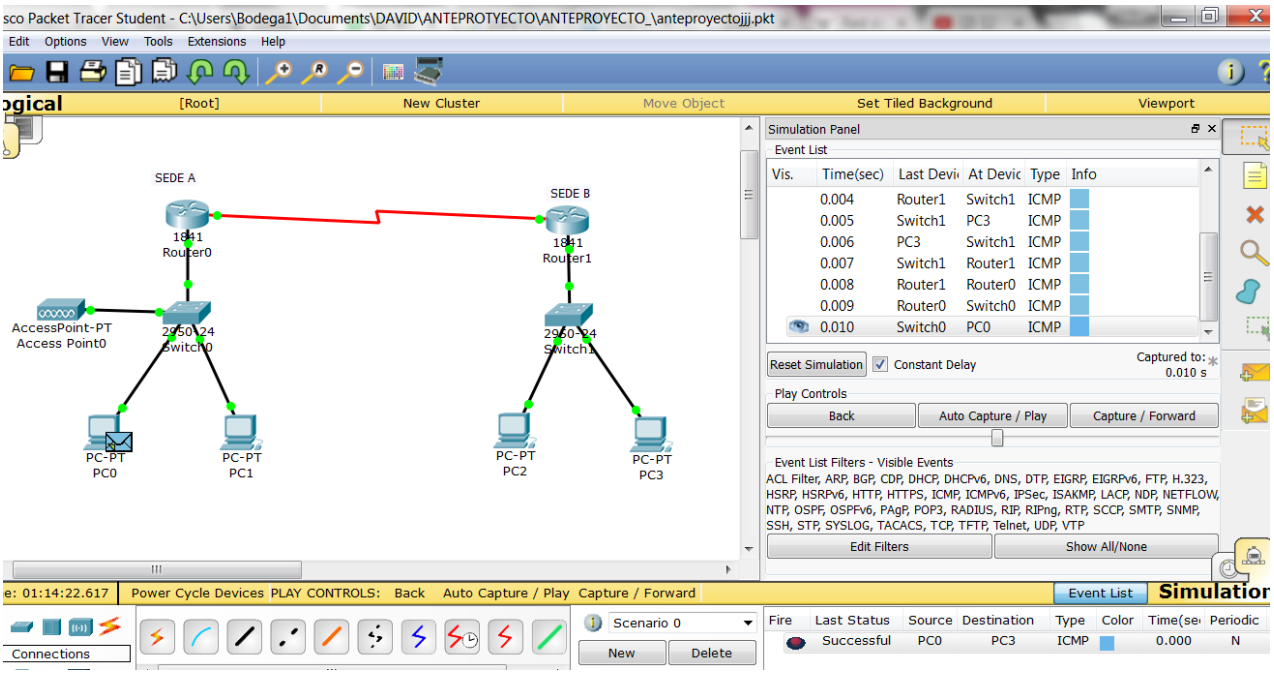


Figura 27.Prueba funcional de simulación.

En la Figura 27 se puede observar la transición del paquete, este se envía hasta llegar a su destino, y al finalizar se muestra el símbolo Check con color verde, esto indica que el paquete fue enviado correctamente.

10.6 Desarrollo del Sistema

El desarrollo del sistema se realizó de forma convencional, haciendo alusión a 3 capas de programación, la primera capa es la interfaz gráfica, la segunda capa es la parte lógica del sistema y por último el acceso a datos, que, para el caso del sistema, son los servicios web para acceder a la base de datos.

10.6.1. Desarrollo de la aplicación de escritorio

En los siguientes capítulos (8.6.1.1. hasta 8.6.1.4.) se explica el desarrollo del programa en JAVA para escritorio. En estos capítulos se muestra lo más fundamental del desarrollo.

10.6.1.1. Clases Java del proyecto

A continuación, se muestra la lista de clases que pertenecen al proyecto de la aplicación de escritorio en JAVA:

- BaseOrange: Es la clase principal, que simplemente ejecuta la clase llamada Ventana1.
- ConexionBD: Es la clase que permite establecer un enlace entre el programa en JAVA y la base de datos en MySQL. En esta clase se maneja la capa de negocios, dado que se reciben las claves de acceso a la base de datos en MySQL.
- Ventana1: Es una clase que hereda de la clase JFrame, que está definida por la librería javax.swing.JFrame. En esta clase se encarga de manipular la interfaz gráfica del programa en JAVA. Esta clase se muestra con mayor profundidad en el capítulo 8.6.1.5.
- ventanaIp: Es una clase similar a Ventana1, su única funcionalidad es poder cambiar la dirección ip a la que se conecta el programa.
- ventanaUsu: Es una clase similar a Ventana1, su única funcionalidad es poder cambiar el usuario de acceso de la base de datos en MySQL.

Se puede observar en la Figura 28 el mapa de clases, en la misma figura se expresa que ninguna clase hereda de otra, simplemente se instancian.

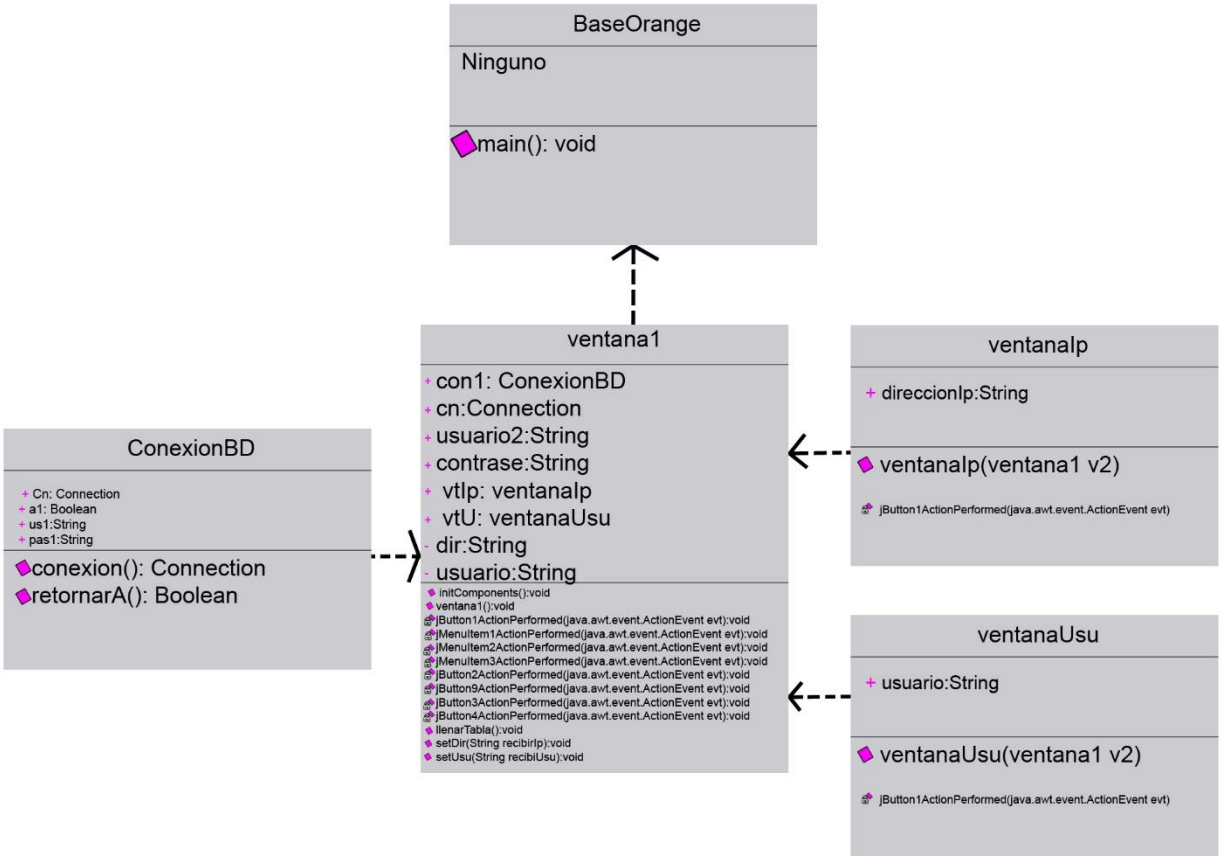


Figura 28. Mapa de clases del programa JAVA de escritorio.

10.6.1.2. Conexión a la base de datos

La clase que maneja el acceso a la base de datos es la clase ConexionBD. (Como se nombró en el capítulo 8.6.1.1.) Esta clase maneja la dirección IP a la cual se va a acceder a la base de datos de MySQL. Asimismo, maneja el nombre de usuario y contraseña con los cuales se acceden a la base de datos. Tanto la dirección IP, nombre de usuario y contraseña se guardan en una cadena de caracteres, como se muestra a continuación:

```
public String us1="root";
public String pas1="";
public String dir="localhost";
```

Se colocan de índole público, para que las clases Ventana1, ventanaIp y ventanaUsu puedan modificar cada cadena de caracteres.

A continuación se muestra el código del método que permite establecer la conexión entre la base de datos y el programa en JAVA. El código es idéntico al que se mostró en

el capítulo 7.1.5., su función es idéntica, se recibe un nombre de usuario (por medio del String dir), usuario de acceso (String us1) y una contraseña (String pas1). Para su mayor entendimiento se puede referenciar el capítulo 7.1.5., dado que esta clase se basó al ejemplo desarrollado anteriormente en el documento.

```
public Connection conexion()
   {

      try {
      Class.forName("com.mysql.jdbc.Driver");
      cn = DriverManager.getConnection("jdbc:mysql://"+dir+"/sistema",us1,pas1);
      System.out.println("conecto");
      a1=true;

      } catch(Exception e){
          System.out.println(e.getMessage());
      } return cn;

   }
```

10.6.1.3. Eventos de la aplicación de escritorio

Como se nombró en el capítulo 10.6.1.1. Clases Java del proyecto, la clase que se encarga del manejo de la interfaz gráfica del programa es la clase ventana1.Asimismo, se había nombrado que la clase ventana1 hereda de la clase JFrame. Todo el código de la clase ventana1 se puede visualizar en la sección de **ANEXOS** (Sección 11.2.4).

La clase ventana1 por ser una clase que hereda las propiedades de JFrame, será una clase que permita el diseño de una ventana, colocar unos botones (llamados por defecto como objetos jButton), colocar espacios para poder capturar información en cadena de caracteres (llamados por defecto como objetos jTextfield), una barra horizontal para colocar opciones tipo menú, una tabla de información (llamado por defecto como un objeto jTable), etiquetas de texto (llamados como JLabel), un menú despegable (llamado jMenu) e ítems del mismo menú (jMenuItem). En el siguiente cuadro de texto se muestran los botones, espacios para texto (Textfield) y tabla que se instancian en la clase ventana1.

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton9;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JMenu jMenu1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JMenuItem jMenuItem1;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JScrollPane jScrollPane7;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
```

En los eventos de los botones y de los ítems del menú, se encuentra la lógica con la que funciona la interfaz. Uno de los primeros eventos que se encontrará el usuario es hacer clic sobre el ítem de “conectar”, como expresa su nombre, este evento conectará el programa con la base de datos de MySQL. El evento asociado al ítem “conectar” ejecuta el siguiente código:

```
ConexionBD con1 = new ConexionBD();
Connection cn = null;
…
private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
       cn=con1.conexion();
       boolean A1=con1.geta1();
  }
```

En el código anterior se puede ver que, la clase ConexionBD se instancia como un objeto llamado con1 (esta sección se encuentra al iniciar la clase ventana1), el método "jMenuItem3ActionPerformed" es el mismo evento que se hablaba en el párrafo anterior. A su vez, en el método se puede ver que el objeto cn de la clase Connection, se inicializa por medio de la sentencia "cn=con1.conexion()". La clase connection es la que permite se devuelvan los resultados de cada sentencia SQL (Oracle, 2016).

A continuación se muestra el código que se ejecuta cuando se hace clic en el botón de mostrar.

```
void llenarTabla()
 {
    DefaultTableModel tabla1 = new DefaultTableModel();
    tabla1.addColumn("Orden PO");
    tabla1.addColumn("Modelo");
    tabla1.addColumn("Serial");
    tabla1.addColumn("Ubicación");
    tabla1.addColumn("Cliente");
    tabla1.addColumn("Entrega");
    jTable1.setModel(tabla1);


    String Sql1= "SELECT * FROM `equipos` ";

    String datos[] = new String [6];
    Statement st;
  try {
      st = cn.createStatement();
       ResultSet rs = st.executeQuery(Sql1);
       while(rs.next())
       {
          datos[0]=rs.getString(1);
          datos[1]=rs.getString(2);
          datos[2]=rs.getString(3);
          datos[3]=rs.getString(4);
          datos[4]=rs.getString(5);
          datos[5]=rs.getString(6);
          tabla1.addRow(datos);
       }
         jTable1.setModel(tabla1);
  } catch (SQLException ex) {
     Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
  }


  }
```

El código que se mostró anteriormente, tiene dos secciones, para entender su funcionamiento, la primera es con el objeto tabla1 de la clase “DefaultTableModel”, cuya función principal es crear el modelo de la tabla que se va a mostrar en pantalla. Cuando se implementa el método “addColumn”, se está creando una nueva columna para la tabla.

Luego, se crea una cadena de caracteres con el nombre de “sql1”, la sentencia “SELECT * FROM `equipos`” que permite mostrar toda la tabla de equipos. Después, se crea un objeto de la clase “statement”, que es la clase de objeto que permite ejecutar una sentencia estática de SQL y devolver un valor según sea su resultado. Por este motivo, cuando se instancia un objeto de la clase “statement” y “ResultSet” (objeto que aloja la respuesta de SQL), se debe realizar dentro de una estructura Try y Catch, por si

el programa encuentra un error. Y por último, el arreglo de datos (que corresponde al identificador datos[]) recibe las cadenas de caracteres por cada fila de la tabla equipos (Esto se realiza dentro de la estructura while que tiene como condición la sentencia rs.next()).

El evento del botón de buscar funciona exactamente igual que el evento anteriormente nombrado (evento del botón “mostrar”). Sin embargo, existe un cambio en la sentencia SQL, que permite buscar según sea el valor de cada columna.

A continuación, se muestra el código que se ejecuta cuando el usuario hace clic sobre el botón de ingresar.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        // TODO add your handling code here:
        PreparedStatement pps = cn.prepareStatement(" INSERT INTO
equipos(Ordendepo,Modelo,Serial,Ubicacion,Entrega,Cliente) VALUES(?,?,?,?,?,?)");
        pps.setString(1, jTextField1.getText());
         pps.setString(2, jTextField2.getText());
         pps.setString(3, jTextField3.getText());
         pps.setString(4, jTextField4.getText());
         pps.setString(5, jTextField5.getText());
         pps.setString(6, jTextField6.getText());
         pps.executeUpdate();
         jLabel13.setText("Los cambios se han guardado satisfactoriamente ");

    } catch (SQLException ex) {
        Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
    }

    }
```

El código de ingresar se ejecuta sobre la instancia del objeto de la clase “PreparedStatement” con el nombre de “pps”. El objeto “pps” se inicializa por medio de la sentencia “cn.preparedStatement()”. Se puede observar que dentro de la inicialización del objeto pps hay una sentencia SQL. Esta sentencia lo que permite es insertar información dentro de la tabla equipos. Luego, el objeto pps permite colocar las cadenas de caracteres que corresponden al valor que se guardara en la tabla. El método “.executeUpdate()” permite ejecutar la sentencia SQL.

El código del evento del botón “egresar”, es similar al código de ingresar información a la tabla. Sólo se cambia la sentencia en SQL. Que permite eliminar (por medio de la palabra DELETE) una fila dentro de la tabla según sea el serial que ingrese el usuario, esta sentencia se instancia por medio de un objeto “PrepraredStatement” como el código de “ingresar”. Asimismo, el funcionamiento del evento del botón “reubicar”, se ejecuta por medio de la sentencia SQL que comienza por la palabra UPDATE, que

permite modificar información en específico de la tabla de equipos.

10.6.1.4. Código para cambiar la dirección IP

Este código se ejecuta por medio del objeto “jMenuItem1”, de la clase ventana1. El código se puede ver en detalle en seguida:

```
    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {

       vtIp = new ventanaIp(this);
       vtIp.setVisible(true);
    }
```

El código anterior muestra cómo se instancia un objeto de la clase ventanaIP que en su método de contructor recibe el contexto de la clase ventana1, esto permite que desde el objeto de la clase ventanaIP se puedan manipular las variables de ventana1. Y en específico, la variable que aloja la cadena de caracteres que corresponden a la dirección IP que necesita la clase conexionBD, para iniciar la conexión con la base de datos en MySQL.

A continuación se muestra el código que corresponde a la clase ventanaIp:

```
public String direccionIp="";
 private ventana1 v1=null;

  public ventanaIp(ventana1 v2) {
       v1=v2;
       initComponents();
    }
…
```

En el código anterior se puede observar que, se crea la cadena de caracteres (direccionIp) que alojara la nueva dirección ip, también, se crea un objeto de clase ventana1, sin embargo, el objeto v1 se instancia en el constructor que recibe en el caso puntual el contexto de la ventana1 que ejecutó la clase ventanaIp.

En seguida se muestra el código del evento del botón “Aceptar” de la ventanaIp:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
       direccionIp=jTextField1.getText();
       v1.setDir(direccionIp);
    }
```

El código anteriormente mostrado, tiene la función de tomar el texto que se encuentra en pantalla y lo envía como una cadena de caracteres en un método propio de ventana1, el nombre del método es “setDir”. A continuación se muestra el código que ejecuta el método “setDir”.

```
public String dir="";
…
public void setDir(String recibirIp)
   {
      dir=recibirIp;
      vtIp.setVisible(false);
      con1.setDireccionIp(dir);
   }
```

El código del método “setDir” tiene dos funciones, la primera función es guardar como una cadena de caracteres la dirección IP en una variable global llamada “dir”. La segunda función es ocultar el objeto de la clase ventanaIp.

El código para el cambio de usuario es exactamente igual al proceso de cambiar la dirección IP. Para mayor profundidad del código, véase la sección de **ANEXOS** (desde la sección 11.2.9. hasta la sección 11.2.12):

10.6.2. Desarrollo de la aplicación de Android

En los siguientes capítulos (8.6.2.1. hasta 8.6.2.3.) se explica el desarrollo de la aplicación Android. En estos capítulos se muestra lo más fundamental del desarrollo.

10.6.2.1. Actividades y diseño Android del proyecto

Los proyectos para aplicaciones Android, se componen de dos factores fundamentales, el primero son las actividades, que son las clases JAVA que manipulan la lógica de la aplicación móvil. El segundo factor es el diseño de cada pantalla (conocidos comúnmente como Layouts). El diseño de pantalla se programa por medio del lenguaje XML para Android.

Las dos principales actividades son “MainActivity” y “PresentacionLayout”.

10.6.2.2. Código de la actividad principal

Es necesario recordar que los métodos OnCreate son los que se ejecutan cuando la actividad de la aplicación acaba de iniciar. A continuación se muestra el método OnCreate de la actividad principal (MainActivity):

```
private EditText eSerial;
private EditText ePOrder;

public String dirIP ="192.168.0.14";
private String urlLogin = "http://"+dirIP+"/login/procesar_login.php";
private String urlSerial = "http://"+dirIP+"/login/procesar_serial.php";
private String urlpOrder = "http://"+dirIP+"/login/procesar_po.php";

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    eSerial = (EditText) findViewById(R.id.eSerial);
    ePOrder = (EditText) findViewById(R.id.ePOrder);

}
```

Antes de analizar el código del método OnCreate, primero se instancian unos objetos de la clase EditText, cuya función es capturar el texto que ingresara el usuario, los dos espacios para editar texto guardarán la cadena de caracteres del serial u orden de compra por la cual se está buscando. Después, se crea una cadena de caracteres que contenga la dirección IP dónde está alojada la base de datos. Luego, se crean tres cadenas de caracteres que contienen las rutas lógicas de los servicios web correspondientes al valor que vaya a solicitar el usuario.

En el método OnCreate, se instancia el constructor de la clase (es decir el constructor que se va a encargar que la clase funcione como una actividad normal para Android), se relaciona la actividad con la plantilla (tipo Layout) “layout.activity_main”, por medio de la sentencia “setContentView”. Luego, se instancian los objetos “eSerial” y “ePOrder” por medio de la sentencia “findViewById”. Los métodos findViewById están definidos por la documentación oficial para Android, como “… una vista que fue identificada por el atributo “android: id XML” que se procesó en el método onCreate...” (Activity, 2017). Es decir que por medio del método “findViewById” se logra hacer un enlace entre la interfaz gráfica que maneja el objeto “EditText” (pero en general para cualquier objeto funciona igual) y sus funciones lógicas en JAVA.

En el código XML llamado “layout_activity_main”, se puede encontrar la sentencia “android:onClick="btnBuscar"”, esta sentencia en especifico relaciona cada botón con el método nombrado entre comillas (en este caso el método que ejecutará por medio del clic será el método “btnBuscar”). A continuación se muestra el código del método

“btnBuscar”:

```
public void btnBuscar(View view){
        final String serial = eSerial.getText().toString();
        // INICIAR LA CONEXION CON VOLLEY
        StringRequest request = new StringRequest(Request.Method.POST, urlSerial,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        // SE EJECUTA CUANDO LA CONSULTA SALIO BIEN, SIN ERRORES
                        if(response.equals("0")) {
                            Toast.makeText(MainActivity.this, "Ingreso incorrecto...",
Toast.LENGTH_SHORT).show();
                        } else {
                            try {
                                System.out.println("RESPUESTA DE SERVIDOR : "+response);
                                Toast.makeText(MainActivity.this, response,
Toast.LENGTH_LONG).show();
                                JSONArray jsonArray = new JSONArray(response);
                                String respSql = response;
                                Intent intent = new Intent(MainActivity.this, PresentacionLayout.class);
                                intent.putExtra("respSql",respSql);
                                startActivity(intent);
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                        }
                    }
                },

                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                        Toast.makeText(MainActivity.this, "ERROR DE CONEXION...",
Toast.LENGTH_SHORT).show();
                    }
                }){
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> parametros = new HashMap<>();
                parametros.put("serial", serial);
                return parametros;
            }
        };
        AppController.getInstance().addToRequestQueue(request);
    }
```

El método “btnBuscar” inicia instanciando una cadena de caracteres que toma el valor

correspondiente al EditText, dónde se colocaría el serial que se desea buscar. Luego, se crea un objeto de clase “StringRequest”. Según la librería oficial de Volley, se dice que un “StringRequest” es “…Una solicitud comprimida para recuperar el cuerpo de respuesta en una URL determinada como una cadena…” (Volley, 2013). Dado que la plataforma XAMPP no está habilitada para soportar Web Services convencionales, se realiza una clase base que contiene información relacionada con la red (en este caso la cadena de caracteres que contiene la URL), como métodos HTTP , para este caso, contiene un método POST, que simplemente servirá para recibir el valor correspondiente al serial. Sin embargo, como se trata de desarrollar un servicio web, se recibe un objeto de notación JSON. A este tipo de web services se les conocen como “RESTful” (Netbeans, 2017), dado que se cuenta con una cabecera HTTP (en este caso la solicitud POST que se realiza mediante la librería Volley) y se recibe una respuesta ya sea en formato XML, JSON (que es como se recibe acá) o en formato HTML.

El código de “StringRequest” lanza por defecto dos métodos. El primero es el método “onResponse”, que solicita el tipo de solicitud que se va a realizar (POST), a su vez, se solicita la Url donde se aloja el servicio web (String urlSerial). Luego, si la consulta es errónea, lanzará un mensaje de error (el mensaje es “Ingreso incorrecto…”). En caso de ser una consulta exitosa, lanzará el mensaje de “RESPUESTA DE SERVIDOR:” junto a la respuesta de la consulta en SQL. Hay una explicación con mayor profundidad en el capítulo de desarrollo de los servicios web 8.7. El segundo método del código “StringRequest” es el método “onErrorResponse”, el método simplemente se ejecutará en caso de tener un error en cuanto a la conexión con la base de datos.

Por último, el evento (de índole “listener”) del botón de búsqueda por serial, ejecuta un método llamado “getParams”, que permite usar un objeto de clase Map por medio de la interfaz llamada HashMap, el objeto de clase Map se llama parámetros, parámetros alojará la llave (llamada “serial”) y su valor correspondiente (que es el valor de la cadena de caracteres llamada serial), todo esto se desarrolla gracias a la interfaz de HashMap por medio del método “.put("serial", serial)”. El objeto de clase Map llamado “parametros”, es el objeto que dará la respuesta (tipo POST) en el script de PHP.

Los eventos (tipo “listener”) de la actividad principal, funcionan igual al método “btnBuscar”, sólo cambian las direcciones URL.; en el caso del evento del botón “MOSTRAR TABLA” no se utiliza el objeto de clase Map, porque para el caso en específico de mostrar toda la tabla no se necesita enviar ningún elemento o respuesta en cadena de caracteres.

10.6.2.3. “Intent” para la segunda actividad del aplicativo

Ya teniendo la respuesta de la consulta de SQL en un objeto tipo JSON. Se procede a crear un objeto de clase “Intent”, el método que permitirá enviar la respuesta de la consulta hacía la segunda actividad, es el método “putExtra” (la sentencia de código exacta que se encarga de esto es “intent.putExtra("respSql",respSql);”). Una vez instanciado el objeto de clase “Intent”, se ejecuta la segunda actividad de la aplicación Android. (PresentacionLayout)

```
…
private String glResp;
…
Bundle bundle = getIntent().getExtras();
…
  glResp=bundle.getString("respSql");
```

El código anterior muestra cómo se toma el valor de la respuesta a la consulta SQL, por medio del objeto de clase “intent”. Todo consiste en coger la información extra de un objeto tipo Bundle y del mismo objeto Bundle tomar la cadena de caracteres correspondiente a la respuesta de la consulta SQL.

10.7. Desarrollo de los servicios web

Con el fin de proteger los datos, se debe alojar la base de datos en un único computador (o equipo que permita compartir la base de datos por medio de la red) y el resto de equipos que deseen usar la base de datos, comunicarse por medio de los servicios web, para poder manipular la base de datos. En la Figura 29. Topología del acceso a la base de datos. Se puede observar una topología que muestra un computador que aloja una base de datos y el resto de equipos, sólo acceden al equipo por medio de servicios web, con el fin de acceder a la base de datos.

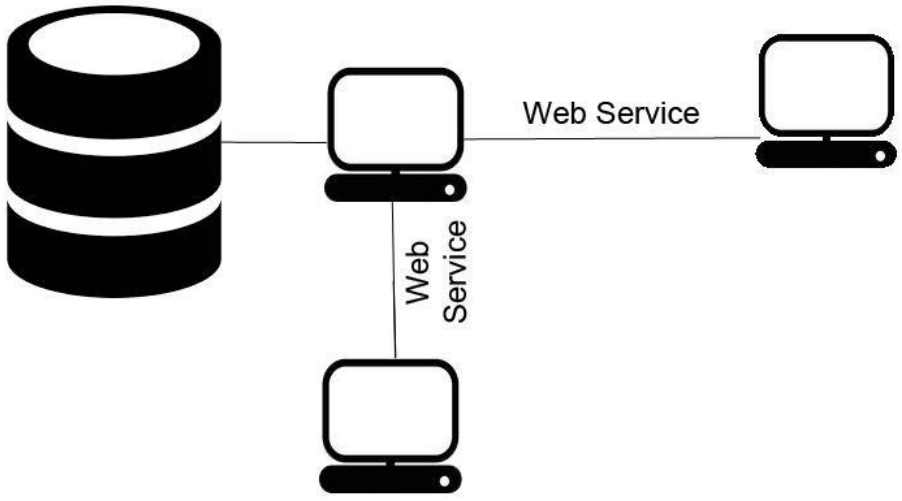


Figura 29. Topología del acceso a la base de datos.

Los servicios web se desarrollaron por medio de la plataforma XAMPP, que como se

explicó en el marco teórico, XAMPP posee consigo el Apache que permitirá simular (se coloca la palabra simular porque realmente un computador no es por defecto un servidor) un servidor en la dirección local. Entonces, como cualquier servidor, es necesario administrar las carpetas que alojan la configuración HTTP del sistema. En estas carpetas se coloca el código en PHP que va a permitir crear el servicio web. Por consiguiente, se crea una nueva carpeta llamada “login” con la siguiente dirección por defecto:

```
C:\xampp\htdocs\login
```

En la carpeta “login”, se procede a crear los ficheros PHP, llamados:

- conexion.php: Script que permite establecer una conexión con la base de datos de MySQL.
- procesar_log.php: Script que permite realizar una consulta sobre la tabla completa de equipos.
- procesar_po.php: Script que permite realizar una consulta según sea la orden de compra en la tabla equipos.
- procesar_serial.php: Script que permite realizar una consulta según sea el serial en la tabla equipos.

A continuación, se muestra el código fuente del fichero “conexion.php”:

```
<?php
$mysqli = new mysqli("localhost", "root", "", "sistema");

	if($mysqli->connect_errno) {
		die("Fallo la conexion");
	} else {
		//echo "Conexion exitosa !";
	}
?>
```

El fichero “conexion.php” funciona de manera similar a la clase ConexionBD, simplemente requiere de una dirección IP donde está alojada la base de datos, un nombre de usuario y el nombre de la base de datos.

A continuación, se muestra el código fuente del fichero “procesar_serial.php”:

```
<?php
        require 'conexion.php';

        $serial = $_POST['serial'];

        $query = "SELECT * FROM `equipos` WHERE `Serial` LIKE '$serial' ";

        $result = mysqli_query($mysqli, $query);

        if (mysqli_num_rows($result) > 0) {

                $datos = array();

            while($row = mysqli_fetch_assoc($result))
            {
              $datos[] = array_map("utf8_encode", $row);
            }
                $json_array = json_encode($datos);
                echo $json_array;
        } else {

                echo "0";
        }
?>
```

El fichero “procesar_serial.php” empieza por requerir la conexión a la base de datos. Luego, toma el valor de la respuesta tipo POST en la variable “$serial”. Después, el código realiza la consulta SQL deseada (en este caso realiza la búsqueda de filas que contengan el valor de “$serial”). Entonces, va guardando los resultados de la consulta en un arreglo de datos, todo se desarrolla por medio de la estructura “while”, que realiza la comparación en la espera de un nuevo resultado (que es lo mismo que recibir una fila de la tabla). Por último, codifica la respuesta obtenida en un objeto tipo JSON, esto se logra por medio de la sentencia “json_encode”, que según la librería oficial de PHP, devolverá la representación JSON del valor que se le ingrese.

A continuación, se muestra el código Android que permite recibir el objeto JSON (gracias a la librería Volley):

```
JSONArray jsonArray = new JSONArray(respuesta);
String ordenPo = jsonArray.getJSONObject(apuntRes).getString("Ordendepo");
String modelo =  jsonArray.getJSONObject(apuntRes).getString("Modelo");
String serial =  jsonArray.getJSONObject(apuntRes).getString("Serial");
String ubicacion =  jsonArray.getJSONObject(apuntRes).getString("Ubicacion");
String entrega =  jsonArray.getJSONObject(apuntRes).getString("Entrega");
String cliente =  jsonArray.getJSONObject(apuntRes).getString("Cliente");
```

Como se puede observar en el código anterior, el objeto JSON se instancia por medio del constructor “JSONArray(respuesta)” (recordando que la variable respuesta contiene el valor de la consulta realizada al web service). El método “getJSONObject()” permite apuntar al objeto JSON que se ha recibido. El método “.getString("…")” permite buscar el valor según sea la columna (ya sea Ordendepo, Modelo, Serial, etc.) de la fila correspondiente. Después, se muestran al usuario las cadenas de caracteres capturadas por medio de los objetos JSON.

10.8 Pruebas del sistema

A raíz de que se maneja una base de datos, se puede implementar una interfaz en JAVA (como el ejemplo del marco teórico) que realice la comunicación, entre el usuario y la base de datos. Entonces se debe crear la base de datos en algún gestor de base de datos. Para ello se propone que el gestor sea la plataforma XAMPP, que cuenta con la herramienta PhpMyAdmin. Luego para verificar que se puede acceder a la base de datos, se realiza la conexión por medio de la plataforma XAMPP, la prueba se realiza direccionando a la IP del computador que aloja la base de datos.

Con el fin de verificar la estructura del sistema, se deben desarrollar unas pruebas de la Red. Para las pruebas del desarrollo de conectividad de Red se usará un programa de simulación de red llamado Cisco Packet Tracer, es una herramienta didáctica que brinda simulación visual, creación de topologías de red, insertar diversos equipos o paquetes de datos y permite configurar los equipos desde sus consolas.

De no haber conexión, se hacen distintas pruebas desde la consola del host de origen (por la ventana donde se puede enviar cualquier tipo de comandos al sistema operativo), para este caso se usaría el comando Ping, es un comando que frecuentemente se usa para verificar si hay errores de Red o si la conexión es exitosa. Desde el host de origen se usa el comando Ping seguido de la dirección Ip del host de destino para verificar la conexión, el comando Ping envía cuatro paquetes de un tamaño de 32 bytes al host de destino, si los cuatro paquetes son enviados con éxito, hay una conexión de acceso de un host al otro, de no haber conexión entre los hosts no se enviarán los paquetes y puede ser por distintos inconvenientes, pueden haber problemas de conexión físicos como cables con rupturas o error al configurar los puertos Ethernet en el Router o las direcciones Ip que se asignan a los diferentes hosts en la Red LAN.

10.9 Recolección de defectos y ajustes necesarios

Para la solución que se plantea al problema existente, no se necesita una metodología

extensiva para la recolección de defectos y ajustes necesarios. Sin embargo, para el producto final (que sería la interfaz de escritorio en JAVA y aplicación móvil), se planea evaluar la calidad del mismo, mediante una matriz QFD, matriz usada frecuentemente en la ingeniería de productos y calidad. En la Figura 30, se muestran las necesidades básicas de la matriz QFD, asimismo, en la Figura 30 se puede observar de forma simbólica, la arquitectura de la matriz QFD. Para poder medir la matriz QFD, se debe realizar una prueba piloto del sistema. De los resultados de la prueba piloto, se recolectan los datos para corregir los defectos y ajustes necesarios.

Al aplicar la matriz QFD, se puede comparar la optimización del nuevo sistema con respecto al anterior. Donde los usuarios (normalmente referidos como clientes en la aplicación de la QFD) podrán dar una valoración al sistema, lo que permite tener una retroalimentación clara de quienes usarían el sistema.

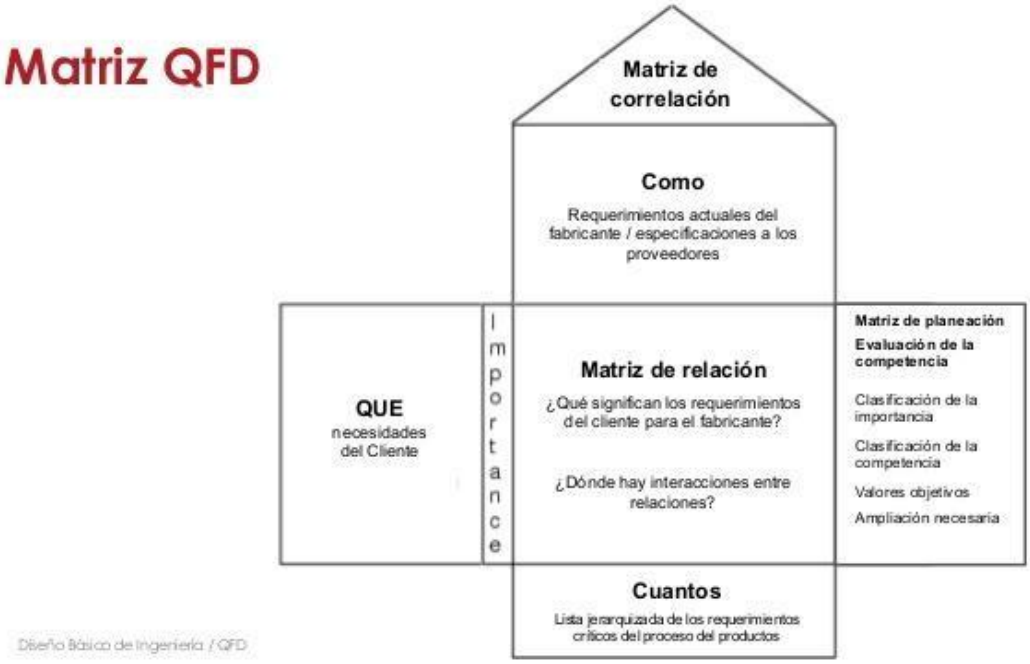


Figura 30. Especificaciones y forma de la matriz QFD.

La matriz QFD soporta la capacidad de evaluar el sistema, alejándose del ámbito meramente técnico. Por ello, su información es valiosa para la retroalimentación por parte de algunos usuarios.

Después de recopilar los datos obtenidos de la prueba piloto, se plantea optimizar el sistema diseñado, haciendo uso de la matriz QFD. A raíz de las pruebas técnicas del sistema, se realizarán las debidas correcciones.

Al aplicar la matriz QFD, se puede comparar la optimización y viabilidad del nuevo

sistema con respecto al anterior, de igual manera, nos permite identificar necesidades del cliente, priorizando la necesidad en función de su importancia. Donde los usuarios (normalmente referidos como clientes en la aplicación de la QFD) podrán dar una valoración al sistema, lo que permite tener una retroalimentación clara de quienes usarían el sistema.

La matriz QFD soporta la capacidad de evaluar el sistema, alejándose del ámbito meramente técnico. Por ello, su información es valiosa para la retroalimentación por parte de algunos usuarios. Se decidió realizar la matriz QFD a usuarios que conocieran el sistema de Orange, para que se pudiera realizar la comparación adecuada.

Los resultados de las dos matrices QFD realizadas dieron resultados muy similares, estas dos matrices se encuentran en ANEXOS **(Sección 11.3.1)**. En la parte de comparación con el sistema anterior, se pudo evidenciar de manera directa que se sienten más cómodos con el sistema nuevo que con el anterior, debido a que sus repuestas reflejan ventajas sobre el sistema actual. En cuanto a identificar las necesidades más importantes a mejorar, se observa la prioridad de mejorar el aspecto de Conexión a la base de datos, porque es el aspecto donde se tiene el número mayor en el ponderado de relatividad, indicando que es el aspecto más importante a trabajar.

Al tener las prioridades de las necesidades que desea el cliente ayuda en mejorar los procesos del sistema o servicios que se presten, paralelamente muestra los aspectos fuertes del sistema. Igualmente al compararlo con el sistema anterior da una visión a futuro de que tan viable y estructurado está el sistema nuevo.

11 ADMINISTRACIÓN DEL PROYECTO

Por lo anteriormente nombrado, el cronograma de actividades del proyecto, se muestra a continuación:

ACTIVIDADES	Mes I				Mes II				Mes III			
	1	2	3	4	5	6	7	8	9	10	11	12
Definir los componentes del sistema con sus funcionalidades	48 h											
Realizar interfaz JAVA	48 h											
Desarrollar la Base de datos MySQL		48 h	48 h									
Conexión entre la base de datos y interfaz en JAVA			48 h									
Conexión entre dos computadores de la base de datos				48 h								
Simular la red					48 h							
Probar funcionamiento de la red					48 h							
Desarollar la aplicación móvil (sin web services)						48 h						
Implementación de los web services						48 h						
Conexión entre la base de datos y aplicación móvil							48 h					
Prueba piloto del sistema								48 h				
Ajustes finales									48 h	48 h		
Presentación del proyecto											48 h	

NOTA: Cada semana del cronograma hace referencia a 48 horas de trabajo individual. Al ser un trabajo colectivo el periodo de tiempo a desarrollar se acorta con respecto al trabajo de una sola persona, se estipula para una persona en un periodo de seis meses, pero en este trabajo son dos (aprobado por el consejo de facultad), dado a la dificultad y extensión del desarrollo. Se estipula culminar en un periodo de tres meses.

12. CONCLUSIONES

En el presente proyecto se lograron determinar dos conclusiones principales. La primera conclusión es en sí mismo el desarrollo del proyecto, los conocimientos aplicados en cada paso del diseño del sistema y ejecución del desarrollo del mismo. La segunda conclusión consiste en una posible implementación a futuro del sistema para suplir una necesidad en la administración de negocios (tanto locales como nivel multinacional), esta conclusión se fundamenta en una similitud hallada a los sistemas ERP. A continuación se muestra cada conclusión en detalle.

12.1. Primera conclusión: Resumen de logros alcanzados y sistema desarrollado

Dado que este trabajo no es de índole investigativo, más bien de la ejecución de un proyecto basado en el levantamiento de requerimientos, una conclusión es el mismo desarrollo del proyecto y su funcionalidad. En la Tabla 1. Resumen de objetivos desarrollados en la ejecución del proyecto. se muestran los pasos que se ejecutaron para cumplir cada objetivo. (Especificando en que subcapítulo del capítulo Ejecución se encuentran desarrollados)

En la siguiente tabla se muestra un contraste entre los capítulos del presente documento donde se muestra la ejecución y análisis del objetivo, objetivos planteados y resultados que se produjeron al ejecutar las directrices, que cumplen el objetivo.

Objetivo	Ejecución y análisis	Resultados
Describir el proceso del manejo de inventario que se desea sistematizar para poder extraer los requerimientos funcionales del prototipo a implementar.	1. Problema y 10.1. Levantamiento de requerimientos	Entender el proceso que se desea optimizar. (Esto se ve reflejado en la tabla 2)
Definir los componentes del sistema con sus funcionalidades para planear el desarrollo del prototipo y medir el avance del desarrollo.	1. Problema y 10.2 Definición de casos de uso	Comprender las necesidades de los sujetos que interactúan con el sistema y proceso actual. (Véase la figura)
Diseñar el modelo de datos del sistema basado en los requerimientos para garantizar la disponibilidad de la información en el	10.3. Diseño del modelo de datos	Definir los parámetros que deben contener la tabla de la base de datos. (Véase la figura)

sistema a implementar.		
Desarrollar una interfaz en java para dar viabilidad de manejar la base de datos a múltiples usuarios que desconocen el manejo de bases de datos en MySQL.	7.1 Bases de Datos, 8. Detalles Técnicos para La Conexión de la Base de Datos en MySQL con Java y 10.4. Diseño de las interfaces de usuario.	Concretar un diseño gráfico para la aplicación de escritorio y aplicación móvil.
Diseñar la infraestructura de redes necesaria para acceder la información del sistema desde un dispositivo móvil.	7.4 Redes y 10.5. Diseño de la infraestructura de comunicación necesaria (Redes)	Detallar el proceso de implementación de una red, que será un medio de comunicación para el sistema.
Simular la infraestructura diseñada para emplearla como canal de comunicación entre un dispositivo móvil y la información de inventario.	7.4 Redes y 10.5.2.1 Diseño de la Topología y simulación.	Definir los detalles técnicos de la configuración del router, para una implementación real de la red.
Desarrollar los servicios web necesarios para soportar los requerimientos extraídos y sirva como interface entre los dispositivos móviles y la información en la base de datos.	7.2 Servicios Web y 10.7. Desarrollo de los servicios web	Poder contar con la base de datos en común para aplicaciones móviles y aplicaciones de escritorio.
Desarrollar una aplicación en un dispositivo móvil que consuma los servicios web desarrollados para que el sistema conformado cumpla con los requerimientos extraídos.	7.3 Aplicaciones móviles y 10.8. Desarrollo de la aplicación de Android	Alcanzar un valor agregado para el mismo sistema. Lo que permite dar más opciones de uso a los usuarios finales del sistema.

Tabla 3. Relación entre objetivos alcanzados, ejecución del mismo y resultados producidos.

Basados en la información que se muestra en la tabla 3, se evidencia que el sistema desarrollado cumple con los objetivos propuestos, permitiendo que este proyecto sea base de muchos procesos administrativos a mejorar. Siendo un proyecto que depende de un código de programación, facilita cualquier cambio o mejoras continuas, que se requieran.

El desarrollo de esta aplicación hace uso de una base de datos permitiendo el fácil manejo de información, para cualquier usuario, así mismo, se desarrolla una red local para poder visualizar la base de datos que es suministrada desde un servidor.

12.2. Segunda conclusión: Trabajo a futuro y comercialización del sistema

La aplicación de escritorio y móvil, se enfocaron en facilitar un proceso de administración de recursos, que en el caso específico de este trabajo es la administración del inventario. Sin embargo, el desarrollo de este trabajo propició un escenario ideal para poder facilitar

cualquier proceso administrativo o de índole gerencial. Porque este sistema cuenta con dos ventajas técnicas. La primera ventaja es que el desarrollo de los programas (aplicación de escritorio y aplicación móvil), los programas permiten que se le puedan añadir nuevos valores agregados, ejemplo, un nuevo valor podría ser documentar informes de manera automática con respecto a los cambios del inventario, otro podría ser poder registrar el flujo de caja de la empresa, entre otros tantos, que implican la administración de una empresa. La segunda ventaja es la funcionalidad de la red, que permite abarcar distintas sedes de una empresa (especialmente una empresa como OBS) y poder compartir información entre distintas oficinas dentro de una misma sede, tanto con equipos de escritorio como con equipos móviles.

Sin embargo, en el mercado ya existen sistemas similares, se conocen como sistemas ERP. Este nombre se debe a sus siglas en inglés, teniendo en cuenta el significado de las siglas ERP (Enterprise Resource Planning), se podría definir los sistemas ERP como los sistemas encargados de la planificación de los recursos empresariales (InformaticaHoy, 2017). Según la revista TIC Portal (tic.Portal, 2017), las principales funciones de un sistema ERP son:

- Contabilidad
- RR.HH.
- Proyectos
- CRM
- Inventario
- Cadena de producción
- Cadena de suministro
- Ventas
- Gestión documental

Todas las funciones anteriormente descritas se podrían agregar al sistema desarrollado, porque los sistemas ERP, en cuanto a lo técnico, cuentan con las mismas características del sistema de inventariado desarrollado. Estas características son la base de datos, la aplicación de escritorio o móvil y una red que permita la comunicación entre oficinas o sedes.

13 BIBLIOGRAFÍA

Activity. (2017). Obtenido de Android Developers: https://developer.android.com/reference/android/app/Activity.html

InformaticaHoy. (2017). *InformaticaHoy*. Obtenido de ¿Qué son los sistemas ERP?: https://www.informatica-hoy.com.ar/software-erp/Que-son-los-sistemas-ERP.php

Netbeans. (2017). *Introduction to Web Services.* Obtenido de Netbeans ORG.

Oracle. (2016). *Interface Connection.* Obtenido de https://docs.oracle.com/javase/7/docs/api/java/sql/Connection.html

tic.Portal. (2017). *Funciones de un ERP*. Obtenido de https://www.ticportal.es/temas/enterprise-resource-planning/funciones-erp

Volley. (2013). *JavaDocs*. Obtenido de http://javadocs.okadatech.com/javadoc/volley/reference/com/android/volley/toolbox/StringRequest.html

Aguilar, I. C. (s.f.). *Modelado de datos*. Obtenido de Interactive and Cooperative Technologies Lab de la Universidad de las Ámericas Puebla.

Alarcón, V. F. (2006). 8. Modelado de datos. En V. F. Alarcón, *Desarrollo de sistemas de información: una metodología basada en el modelado* (págs. 151 - 155). Barcelona: Ediciones UPC.

Alicante, D. d. (26 de Junio de 2014). *Introducción a los Servicios Web. Invocación de servicios web SOAP*. Obtenido de Dpto de Ciencia de la Computación e inteligencia artificial de la Universidad de Alicante: http://www.jtech.ua.es/j2ee/publico/servc-web-2012- 13/sesion01-apuntes.html#%C2%BFQu%C3%A9+es+un+Servicio+Web%3F

Android. (2016). *developer android.* Obtenido de Transmitting Network Data Using Volley. Android, D. (2017). *Android Studio*. Obtenido de El IDE oficial para Android: https://developer.android.com/studio/index.html?hl=es-419

android, s. f. (31 de Mayo de 2016). *La historia de Android Studio*. Obtenido de android studio faqs: http://androidstudiofaqs.com/conceptos/android-studio-historia

Androide, L. (08 de Mayo de 2014). *Xamarin, la API que nos permitirá olvidarnos de Java*. Obtenido de El Androide Libre:

https://www.elandroidelibre.com/2014/05/xamarin-la-api-para- crear-aplicaciones-multiplataforma-en-c-net.html

Artificial, Á. d. (Octubre de 2008). *Tema 4. Servicios WEB.* Obtenido de ccia.ei.uvigo.es.

Center, I. K. (2017). *IBM Knowledge Center*. Obtenido de https://www.ibm.com/support/knowledgecenter/es/SS7K4U_9.0.0/com.ibm.websphere.z series.doc/ae/cwbs_soap.html

CISCO, N. A. (2017). *CISCO, Networl Academy*. Obtenido de CISCO, Networl Academy: http://ecovi.uagro.mx/ccna1/course/module1/index.html#1.2.2.2

CORDOVA. (2017). *Resumen*. Obtenido de Resumen: https://cordova.apache.org/docs/es/latest/guide/overview/

Culturacion. (2012). *¿Qué es y para qué sirve un web service?* Obtenido de Culturacion: http://culturacion.com/que-es-y-para-que-sirve-un-web-service/

El Androide Libre. (29 de 11 de 2015). Obtenido de Despierta tu ordenador de forma remota con estas aplicaciones: http://www.elandroidelibre.com/2015/11/despierta-tu-ordenador-de- forma-remota-con-estas-aplicaciones.html

Elmasri, R., & Navathe, S. B. (2003). *Fundamentals of Database Systems.* Boston, EEUU: Pearson Addison Wesley.

Godoy, M. Á. (2015). *Diseño de una aplicación Android para la recogida de datos de tráfico y su comunicación a un servidor.* Valencia: Escola Tècnica Superior d'Enginyeria Informàtica; Universitat Politècnica de València.

Hoy, I. (2015). *Informatica Hoy.net*. Obtenido de LAN, WAN, MAN, WLAN, WMAN, WWMAN, SAN y PAN: ¿Qué significa cada término?: http://ecomputernotes.com/computernetworkingnotes/computer-network/metropolitan-area-network.

IBM. (s.f.). *Introducción a SOA y servicios web*. Recuperado el 2016, de IBM

Developer Works. IBM. (s.f.). *SOAP Y HTTPS*. Obtenido de IBM Knowledge Center : http://www.ibm.com/support/knowledgecenter/es/temp_sterlingb2bcloud/com.ibm.help.process_enrich.doc/SCN_WSG_Protocols.html

IEEE, C. S. (1983). *IEEE STANDARDS ASSOCIATION.* Obtenido de https://standards.ieee.org/findstds/standard/729-1983.html

Lozano Pérez, M. (2012). *Desarrollo de una aplicación móvil Android para*

control remoto de un servicio web. Madrid, España: Universidad Carlos III de Madrid.

Matesanz, A. G. (2014). *Aplicación Android para la empresa Travelling.* Madrid, España: Universidad Autónoma de Madrid. Obtenido de https://repositorio.uam.es/bitstream/handle/10486/662281/gomez_matesanz_alfonso_tf g.pdf?sequence=1

Merino, M. (12 de Julio de 2014). *¿Qué es una API y para qué sirve?* Obtenido de TICbeat.

Microsoft. (s.f.). *Microsoft TECHNET*. Recuperado el 2016, de Conceptos sobre el motor de base de datos: https://technet.microsoft.com/es-es/library/ms187079(v=sql.105).aspx

MySQL. (2016). *MySQL TM*.

NBAndroid, O. (s.f.). *Develop Android applications using NetBeans IDE.* Obtenido de nbandroid.org: http://nbandroid.org/neoteo. (9 de Diciembre de 2014). *Android Studio 1.0: El primer IDE estable para crear aplicaciones Android*. Obtenido de neoteo: http://www.neoteo.com/android-studio-1-0-el-primer-ide- estable-para-crear-aplicaciones-android/

Netbeans. (2017). *Welcome to the NetBeans Community*. Obtenido de Netbeans.org: https://netbeans.org/about/

ORACLE. (2015). *How to Make Frames (Main Windows)*. Obtenido de How to Make Frames (Main Windows): http://docs.oracle.com/javase/tutorial/uiswing/components/frame.html

Pavón, B. C. (2012). *Diseño de una red WAN para una compañía nacional.* Barcelona: Universidad Oberta de Catalunya.

phpMyAdmin. (2016). *Bringing MySQL to the web*. Obtenido de phpmyadmin.net: https://www.phpmyadmin.net/

PostgreSQL. (2016). *Postgre SQL*. Recuperado el 2016, de The PostgreSQL Global Development Group.

Project, T. L. (09 de Agosto de 2006). *The Linux Information Project*. Obtenido de LINFO: http://www.linfo.org/wan.html

René C Martínez Baños, P. (2005). Diseño de BD Relacionales. Otro enfoque. *Empresa Telefónica de Cuba SA*.

Research, C. B. (2015). *Computer Business Research*. Obtenido de Metropolitan Area

Network. sgoliver.net. (2012). *Acceso a Servicios Web REST en Android* . Obtenido

de sgoliver:

http://www.sgoliver.net/blog/acceso-a-servicios-web-rest-en-android-12/

Villegas, I. G. (2013). *¿Qué es una Aplicación Móvil?* Obtenido de Universidad Abierta y a Distancia. W3C. (11 de 02 de 2004). *Web Services Glossary*. Obtenido de Web Services Glossary:
https://www.w3.org/TR/ws-gloss/#defs

WEBSERVICEX.NET. (2017). *Airport Information Webservice*. Obtenido de GetAirportInformationByCountry:
http://www.webservicex.net/New/Home/ServiceDetail/20

14 ANEXOS

14.1 Código del ejemplo de base de datos.

14.1.1. Código de la clase ComBD.

```
package combd;

/**
 *
 * @author GOGA
 */
public class ComBD {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        JFrame1 JFrame11 = new JFrame1();
        JFrame11.setVisible(true);
    }
```

14.2.1 Código de la clase ConexionBD.

```
package combd;
import java.sql.*;

/**
 *
 * @author GOG
 */
public class ConexionBD {
    Connection cn = null;
    boolean a1= false;
    public Connection conexion()
    {

        try {
        Class.forName("com.mysql.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://localhost/usuarios","root","");
        System.out.println("conecto");
        a1=true;

        } catch(Exception e){
            System.out.println(e.getMessage());
        } return cn;

    }
```

```
    Statement createStatement (){
        throw new UnsupportedOperationException("no soportado");
    }

    boolean geta1 ()
    {
        return a1;
    }

}

}
```

14.3.2 Código de la clase JFrame1

```
package combd;

import java.awt.Color;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author GOGA
 */
public class JFrame1 extends javax.swing.JFrame {
 ConexionBD con1 = new ConexionBD();
 Connection cn = null;
    /**
     * Creates new form JFrame1
     */
    public JFrame1() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jButton3 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
```

```
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jButton3.setFont(new java.awt.Font("Arial", 0, 11)); // NOI18N
        jButton3.setText("Ingresar");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jLabel1.setText("Nombre");

        jLabel2.setText("Edad");

        jLabel3.setText("Correo");

        jButton1.setText("conectar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 28, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 21, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(58, 58, 58)
                        .addComponent(jButton1)
                        .addGap(18, 18, 18)
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addGap(76, 76, 76)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jTextField1)
                        .addComponent(jTextField2)
                        .addComponent(jTextField3, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 72, Short.MAX_VALUE))))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(jLabel1)
                        .addComponent(jLabel2)
                        .addComponent(jLabel3))
                    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addGroup(layout.createSequentialGroup()
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 118,
Short.MAX_VALUE)
                    .addComponent(jButton3)
                    .addGap(30, 30, 30))))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(91, 91, 91)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel2))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 83,
Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jButton3)
                        .addComponent(jButton1))
                    .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
            cn=con1.conexion();
            boolean A1=con1.geta1();


          if(A1)
          {
             jPanel1.setBackground(Color.green);

          }
          else
          {
             jPanel1.setBackground(Color.red);

          }
        }

        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
          int as= Integer.parseInt(jTextField2.getText());
            try {
            PreparedStatement pps = cn.prepareStatement(" INSERT INTO
    datosb(Nombre,Edad,Correo) VALUES(?,?,?)");
            pps.setString(1, jTextField1.getText());
            pps.setInt(2, as);
            pps.setString(3, jTextField3.getText());
            pps.executeUpdate();

            jPanel1.setBackground(Color.green);

         } catch (SQLException ex) {
            Logger.getLogger(JFrame1.class.getName()).log(Level.SEVERE, null, ex);
         }
        }

        /**
         * @param args the command line arguments
         */
        public static void main(String args[]) {
            /* Set the Nimbus look and feel */
            //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
            /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
             * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
             */
            try {
                for (javax.swing.UIManager.LookAndFeelInfo info :
    javax.swing.UIManager.getInstalledLookAndFeels()) {
                    if ("Nimbus".equals(info.getName())) {
                        javax.swing.UIManager.setLookAndFeel(info.getClassName());
                        break;
                    }
                }
            } catch (ClassNotFoundException ex) {


java.util.logging.Logger.getLogger(JFrame1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(JFrame1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(JFrame1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(JFrame1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JFrame1().setVisible(true);
            }
        });
    }


    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JPanel jPanel1;
    private  javax.swing.JTextField  jTextField1;
    private  javax.swing.JTextField  jTextField2;
    private javax.swing.JTextField jTextField3;}
```

14.3.3 Código de la clase Ventana1.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package baseorange;

/**
 *
 * @author goga_
 */
import baseorange.ConexionBD;
import baseorange.ventanaIp;
import java.awt.Color;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.table.DefaultTableModel;

public class ventana1 extends javax.swing.JFrame {
ConexionBD con1 = new ConexionBD();
Connection cn = null;
String usuario2=null;
String contrase=null;
 ventanaIp vtIp = null;
 ventanaUsu vtU = null;
 public String dir="";
 private String usuario="";
    /**
     * Creates new form ventana1
     */
    public ventana1() {

        initComponents();

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel7 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jLabel9 = new javax.swing.JLabel();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jTextField6 = new javax.swing.JTextField();
        jLabel10 = new javax.swing.JLabel();
        jLabel11 = new javax.swing.JLabel();
```

```
jLabel12 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jLabel13 = new javax.swing.JLabel();
jScrollPane7 = new javax.swing.JScrollPane();
jTable1 = new javax.swing.JTable();
jButton9 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
jMenu1 = new javax.swing.JMenu();
jMenuItem1 = new javax.swing.JMenuItem();
jMenuItem2 = new javax.swing.JMenuItem();
jMenuItem3 = new javax.swing.JMenuItem();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setBackground(new java.awt.Color(204, 204, 204));

jLabel7.setText("Orden de PO");

jLabel8.setText("Modelo");

jLabel9.setText("Serial");

jTextField3.setVerifyInputWhenFocusTarget(false);

jLabel10.setText("Ubicación");

jLabel11.setText("Cliente");

jLabel12.setText("Entrega");

jButton1.setBackground(new java.awt.Color(204, 204, 204));
jButton1.setText("Ingresar");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jButton2.setBackground(new java.awt.Color(204, 204, 204));
jButton2.setText("Mostrar");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jLabel13.setText("Esperando una respuesta");

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {

    },
    new String [] {

    }
```

```
));
jScrollPane7.setViewportView(jTable1);

jButton9.setBackground(new java.awt.Color(204, 204, 204));
jButton9.setText("Buscar");
jButton9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton9ActionPerformed(evt);
    }
});

jButton3.setBackground(new java.awt.Color(204, 204, 204));
jButton3.setText("reubicar");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jButton4.setBackground(new java.awt.Color(204, 204, 204));
jButton4.setText("Egresar");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jMenu1.setText("Conexion");

jMenuItem1.setText("Conectar a dirección IP");
jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem1ActionPerformed(evt);
    }
});
jMenu1.add(jMenuItem1);

jMenuItem2.setText("Cambiar Usuario");
jMenuItem2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem2ActionPerformed(evt);
    }
});
jMenu1.add(jMenuItem2);

jMenuItem3.setText("Conectar");
jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem3ActionPerformed(evt);
    }
});
jMenu1.add(jMenuItem3);

jMenuBar1.add(jMenu1);

setJMenuBar(jMenuBar1);
```

```
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane7, javax.swing.GroupLayout.PREFERRED_SIZE, 638,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(171, 171, 171)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE, 121,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED_SIZE, 94,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel11, javax.swing.GroupLayout.PREFERRED_SIZE, 84,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 210,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(jLabel13, javax.swing.GroupLayout.PREFERRED_SIZE, 343,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(69, 69, 69)
                        .addComponent(jButton4)
                        .addGap(18, 18, 18)
                        .addComponent(jButton3)
                        .addGap(18, 18, 18)
                        .addComponent(jButton9)
                        .addGap(18, 18, 18)
                        .addComponent(jButton1)
                        .addGap(18, 18, 18)
                        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(0, 14, Short.MAX_VALUE))
```

```
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(88, 88, 88)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel7)
                            .addComponent(jTextField1,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel8)
                            .addComponent(jTextField2,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel9)
                            .addComponent(jTextField3,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jTextField4,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel10))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jTextField5,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel11))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jTextField6,        javax.swing.GroupLayout.PREFERRED_SIZE,        21,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel12))
                        .addGap(41, 41, 41)
                        .addComponent(jLabel13))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(58, 58, 58)
                        .addComponent(jScrollPane7,        javax.swing.GroupLayout.PREFERRED_SIZE,        322,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,                    119,
Short.MAX_VALUE)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2)
                    .addComponent(jButton9)
                    .addComponent(jButton3)
                    .addComponent(jButton4))
                .addGap(25, 25, 25))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    try {
        // TODO add your handling code here:
        PreparedStatement pps = cn.prepareStatement(" INSERT INTO
equipos(Ordendepo,Modelo,Serial,Ubicacion,Entrega,Cliente) VALUES(?,?,?,?,?,?)");
        pps.setString(1, jTextField1.getText());
         pps.setString(2, jTextField2.getText());
         pps.setString(3, jTextField3.getText());
         pps.setString(4, jTextField4.getText());
         pps.setString(5, jTextField5.getText());
         pps.setString(6, jTextField6.getText());
         pps.executeUpdate();
         jLabel13.setText("Los cambios se han guardado satisfactoriamente ");

    } catch (SQLException ex) {
        Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
    }

    }

    private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
       //con1.setDireccionIp("localhost");
       vtIp = new ventanaIp(this);
       vtIp.setVisible(true);


    }

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        vtU=new ventanaUsu(this);
        vtU.setVisible(true);

    }

    private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
         cn=con1.conexion();
         boolean A1=con1.geta1();
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
            llenarTabla();

    }

    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        DefaultTableModel tabla1 = new DefaultTableModel();
       tabla1.addColumn("Orden PO");
       tabla1.addColumn("Modelo");
       tabla1.addColumn("Serial");
       tabla1.addColumn("Ubicación");
       tabla1.addColumn("Cliente");
       tabla1.addColumn("Entrega");
       jTable1.setModel(tabla1);
```

```
      String a1= jTextField1.getText();
     String a2= jTextField2.getText();
      String a3= jTextField3.getText();
      String a4= jTextField4.getText();
      String a5= jTextField5.getText();
      String a6= jTextField6.getText();

      if(a1.equals(""))
      {
         a1="%";
      }

      if(a2.equals(""))
      {
         a2="%";
      }

      if(a3.equals(""))
      {
         a3="%";
      }

      if(a4.equals(""))
      {
         a4="%";
      }
      if(a5.equals(""))
      {
         a5="%";
      }
      if(a6.equals(""))
      {
         a6="%";
      }

     String sql2 = "SELECT * FROM `equipos` WHERE `Ordendepo` LIKE '"+a1+"' AND `Modelo` LIKE
'"+a2+"' AND `Serial` LIKE '"+a3+"' AND `Ubicacion` LIKE '"+a4+"' AND `Entrega` LIKE '"+a5+"' AND
`Cliente` LIKE '"+a6+"' ";




     String datos[] = new String [6];
      Statement st;
   try {
      st = cn.createStatement();
       ResultSet rs = st.executeQuery(sql2);
       while(rs.next())
       {
          datos[0]=rs.getString(1);
          datos[1]=rs.getString(1);
          datos[2]=rs.getString(2);
          datos[3]=rs.getString(3);
          datos[4]=rs.getString(4);
          datos[5]=rs.getString(5);
```

```
        tabla1.addRow(datos);
     }
      jTable1.setModel(tabla1);
} catch (SQLException ex) {
    Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
}


}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    PreparedStatement pps;
    String ser=jTextField3.getText();
    String ub1=jTextField4.getText();
    if(!ser.equals("")||!ub1.equals(""))
    {
    String sql2 = "UPDATE equipos SET Ubicacion='"+ub1+"'  WHERE Serial='"+ser+"'";
try {
    pps = cn.prepareStatement(sql2);
    pps.executeUpdate();
    jLabel13.setText("Se reubico el equipo con serial "+ser);
} catch (SQLException ex) {
    Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
}
    }
    else
    {
        jLabel13.setText("No se puede realizar reubicación");
    }
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String ser=jTextField3.getText();
    if(!ser.equals(""))
    {
    String sql3 = "DELETE FROM equipos WHERE Serial='"+ser+"'";
    PreparedStatement pps;
try {
    pps = cn.prepareStatement(sql3);
     pps.executeUpdate();
} catch (SQLException ex) {
    Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
}
    }
    else
    {
        jLabel13.setText("Primero se debe ingresar el serial del equipo");
    }

}
void llenarTabla()
{
   DefaultTableModel tabla1 = new DefaultTableModel();
   tabla1.addColumn("Orden PO");
   tabla1.addColumn("Modelo");
```

```
        tabla1.addColumn("Serial");
        tabla1.addColumn("Ubicación");
        tabla1.addColumn("Cliente");
        tabla1.addColumn("Entrega");
        jTable1.setModel(tabla1);


        String Sql1= "SELECT * FROM `equipos` ";

        String datos[] = new String [6];
        Statement st;
    try {
        st = cn.createStatement();
         ResultSet rs = st.executeQuery(Sql1);
         while(rs.next())
         {
            datos[0]=rs.getString(1);
            datos[1]=rs.getString(2);
            datos[2]=rs.getString(3);
            datos[3]=rs.getString(4);
            datos[4]=rs.getString(5);
            datos[5]=rs.getString(6);
            tabla1.addRow(datos);
         }
           jTable1.setModel(tabla1);
    } catch (SQLException ex) {
        Logger.getLogger(ventana1.class.getName()).log(Level.SEVERE, null, ex);
    }


    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(ventana1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(ventana1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(ventana1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(ventana1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ventana1().setVisible(true);
            }
        });
    }

    public void setDir(String recibirIp)
    {
        dir=recibirIp;
        vtIp.setVisible(false);
        con1.setDireccionIp(dir);
    }
    public void setUsu(String recibiUsu)
    {
        usuario2=recibiUsu;
        vtU.setVisible(false);
        con1.setUsuario(usuario2);
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton9;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JMenuItem jMenuItem3;
    private javax.swing.JScrollPane jScrollPane7;
    private javax.swing.JTable jTable1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
    // End of variables declaration
}
```

14.3.4 Clase ConexionBD de la aplicación de escritorio

```
package baseorange;
import java.sql.*;

/**
 *
 * @author GOGA
 */
public class ConexionBD {
    Connection cn = null;
    boolean a1= false;
    public String us1="root";
    public String pas1="";
    public String dir="localhost";
    public Connection conexion()
    {

        try {
        Class.forName("com.mysql.jdbc.Driver");
        cn = DriverManager.getConnection("jdbc:mysql://"+dir+"/sistema",us1,pas1);
        System.out.println("conecto");
        a1=true;

        } catch(Exception e){
            System.out.println(e.getMessage());
        } return cn;

    }
    void setUsuario(String a1)
    {
        us1=a1;

    }
    void setDireccionIp(String dirIp)
    {
        dir=dirIp;
    }

    Statement createStatement (){
        throw new UnsupportedOperationException("no soportado");
    }

    boolean geta1 ()
    {
        return a1;
    }

}
```

14.3.5 Clase ventanaIp de la aplicación de escritorio

```
package baseorange;
import baseorange.ventana1;

/**
 *
 * @author goga_
 */
public class ventanaIp extends javax.swing.JFrame {
    public String direccionIp="";

    private ventana1 v1=null;
    /**
     * Creates new form ventanaIp
     */
    public ventanaIp(ventana1 v2) {
        v1=v2;
        initComponents();
    }

    private ventanaIp() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jTextField1.setText("Coloque aquí la dirección IP");

        jButton1.setBackground(new java.awt.Color(204, 204, 204));
        jButton1.setText("Aceptar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
```

```
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(77, 77, 77)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 184,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(139, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(47, 47, 47))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(33, 33, 33)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        direccionIp=jTextField1.getText();
        v1.setDir(direccionIp);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
```

```
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ventanaIp.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ventanaIp.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ventanaIp.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ventanaIp.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ventanaIp().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}
```

14.3.6 Clase ventanaUsu de la aplicación de escritorio

```
package baseorange;

/**
 *
 * @author goga_
 */
```

```
public class ventanaUsu extends javax.swing.JFrame {
  public String usuario="";

    private ventana1 v1=null;
    /**
     * Creates new form ventanaUsu
     */
    public ventanaUsu(ventana1 v2) {
         v1=v2;
        initComponents();
    }

    private ventanaUsu() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }



    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jTextField1.setText("Coloque aquí el nombre de Usuario");

        jButton1.setBackground(new java.awt.Color(204, 204, 204));
        jButton1.setText("Aceptar");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(53, 53, 53)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButton1)
                    .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 286,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(61, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(24, 24, 24)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jButton1)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        usuario=jTextField1.getText();
        v1.setUsu(usuario);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ventanaUsu.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ventanaUsu.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ventanaUsu.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ventanaUsu.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ventanaUsu().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JTextField jTextField1;
    // End of variables declaration
}
```

14.3.7 Clase BaseOrange de la aplicación de escritorio

```
package baseorange;

/**
 *
 * @author goga_
 */
public class BaseOrange {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        ventana1 vt1 = new ventana1();
        vt1.setVisible(true);

    }

}
```

14.3.8 Clase MainActivity de la app Android

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.tutos.elvis.androidvolley.VolleyController.AppController;

import org.json.JSONArray;
import org.json.JSONException;

import java.util.HashMap;
import java.util.Map;

public class MainActivity extends AppCompatActivity {

    private EditText eSerial;
    private EditText ePOrder;
    public String dirIP ="192.168.0.14";
    private String urlLog = "http://"+dirIP+"/login/procesar_log.php";
    private String urlSerial = "http://"+dirIP+"/login/procesar_serial.php";
    private String urlpOrder = "http://"+dirIP+"/login/procesar_po.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        eSerial = (EditText) findViewById(R.id.eSerial);
        ePOrder = (EditText) findViewById(R.id.ePOrder);

    }
    private int contarResultados(String sqlResul){
        char[] cadenaCh = sqlResul.toCharArray();
        int tam =0;
        for (int i=0; i < cadenaCh.length; i++)
        {
            if(cadenaCh[i]=='{')
            {
                tam++;
            }
        }
        return tam;
    }

    public void btnAceptar(View view){
```

```
        // INICIAR LA CONEXION CON VOLLEY
        StringRequest request = new StringRequest(Request.Method.POST, urlLogin,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {

                        if(response.equals("0")) {
                            Toast.makeText(MainActivity.this, "Ingreso incorrecto...",
Toast.LENGTH_SHORT).show();
                        } else {
                            try {
                                System.out.println("RESPUESTA DE SERVIDOR : "+response);

                                Toast.makeText(MainActivity.this, response, Toast.LENGTH_LONG).show();
                                JSONArray jsonArray = new JSONArray(response);


                                String respSql = response;

                                Intent intent = new Intent(MainActivity.this, PresentacionLayout.class);

                                intent.putExtra("respSql",respSql);

                                startActivity(intent);
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                        Toast.makeText(MainActivity.this, "ERROR DE CONEXION...",
Toast.LENGTH_SHORT).show();
                    }
                }){
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {

                Map<String, String> parametros = new HashMap<>();

                return parametros;
            }
        };
        AppController.getInstance().addToRequestQueue(request);
    }

    public void btnBuscar(View view){
        final String serial = eSerial.getText().toString();
```

```
    // INICIAR LA CONEXION CON VOLLEY
    StringRequest request = new StringRequest(Request.Method.POST, urlSerial,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    // SE EJECUTA CUANDO LA CONSULTA SALIO BIEN, SIN ERRORES
                    if(response.equals("0")) {
                        Toast.makeText(MainActivity.this, "Ingreso incorrecto...",
Toast.LENGTH_SHORT).show();
                    } else {
                        try {
                            System.out.println("RESPUESTA DE SERVIDOR : "+response);

                            Toast.makeText(MainActivity.this, response, Toast.LENGTH_LONG).show();
                            JSONArray jsonArray = new JSONArray(response);



                            String respSql = response;

                            Intent intent = new Intent(MainActivity.this, PresentacionLayout.class);

                            intent.putExtra("respSql",respSql);

                            startActivity(intent);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

                    Toast.makeText(MainActivity.this, "ERROR DE CONEXION...",
Toast.LENGTH_SHORT).show();
                }
            }){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {

            Map<String, String> parametros = new HashMap<>();
            parametros.put("serial", serial);

            return parametros;
        }
    };
    AppController.getInstance().addToRequestQueue(request);
  }
```

```
    public void btnBuscarpo(View view){
        final String pOrder = ePOrder.getText().toString();


        // INICIAR LA CONEXION CON VOLLEY
        StringRequest request = new StringRequest(Request.Method.POST, urlpOrder,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        // SE EJECUTA CUANDO LA CONSULTA SALIO BIEN, SIN ERRORES
                        if(response.equals("0")) {
                            Toast.makeText(MainActivity.this, "Ingreso incorrecto...",
Toast.LENGTH_SHORT).show();
                        } else {
                            try {
                                System.out.println("RESPUESTA DE SERVIDOR : "+response);

                                Toast.makeText(MainActivity.this, response, Toast.LENGTH_LONG).show();
                                JSONArray jsonArray = new JSONArray(response);



                                String respSql = response;

                                Intent intent = new Intent(MainActivity.this, PresentacionLayout.class);

                                intent.putExtra("respSql",respSql);

                                startActivity(intent);
                            } catch (JSONException e) {
                                e.printStackTrace();
                            }
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {

                        Toast.makeText(MainActivity.this, "ERROR DE CONEXION...",
Toast.LENGTH_SHORT).show();
                    }
                }){
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {

                Map<String, String> parametros = new HashMap<>();
                parametros.put("pOrder", pOrder);

                return parametros;
            }
```

```
        };
        AppController.getInstance().addToRequestQueue(request);
    }
}
```

14.3.9 Clase PresentacionLayout de la app Android

```
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import org.json.JSONArray;
import org.json.JSONException;

/**
 * Created by elvis on 01/10/2016.
 */
public class PresentacionLayout extends AppCompatActivity implements View.OnClickListener{

    private TextView tvSaludo;
    //private  Bundle bundle = getIntent().getExtras();
    private int apuntRes;
    private int cantRel;
    private String glResp;
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.presentacion_layout);
        Bundle bundle = getIntent().getExtras();
        cantRel = contarResultados(bundle.getString("respSql"));
        //tvSaludo = (TextView) fin
dViewById(R.id.tvSaludo);
        Button sig = (Button)findViewById(R.id.btnSig);
        Button ant = (Button)findViewById(R.id.btnAnt);
        sig.setOnClickListener(this);
        ant.setOnClickListener(this);
        //llenarSaludo();
        apuntRes=0;
        glResp=bundle.getString("respSql");
        imprimirResultado(bundle.getString("respSql"));
    }

    private void imprimirResultado(String respuesta) {
        TextView tv1 = (TextView)findViewById(R.id.textView6);
        TextView tv2 = (TextView)findViewById(R.id.textView10);
        TextView tv3 = (TextView)findViewById(R.id.textView12);
        TextView tv4 = (TextView)findViewById(R.id.textView22);
```

```
    TextView tv5 = (TextView)findViewById(R.id.textView32);
    TextView tv6 = (TextView)findViewById(R.id.textView42);

    try {
        JSONArray jsonArray = new JSONArray(respuesta);
        String ordenPo = jsonArray.getJSONObject(apuntRes).getString("Ordendepo");
        String modelo =  jsonArray.getJSONObject(apuntRes).getString("Modelo");
        String serial =  jsonArray.getJSONObject(apuntRes).getString("Serial");
        String ubicacion =  jsonArray.getJSONObject(apuntRes).getString("Ubicacion");
        String entrega =  jsonArray.getJSONObject(apuntRes).getString("Entrega");
        String cliente =  jsonArray.getJSONObject(apuntRes).getString("Cliente");
        tv1.setText(ordenPo);
        tv2.setText(modelo);
        tv3.setText(serial);
        tv4.setText(ubicacion);
        tv5.setText(entrega);
        tv6.setText(cliente);

    } catch (JSONException e) {
        e.printStackTrace();
    }

}

private int contarResultados(String sqlResul){
    char[] cadenaCh = sqlResul.toCharArray();
    int tam =0;
    for (int i=0; i < cadenaCh.length; i++)
    {
        if(cadenaCh[i]=='{')
        {
            tam++;
        }
    }
    return tam;
}

@Override
public void onClick(View view) {
    Bundle bundle = getIntent().getExtras();
    //imprimirResultado(bundle.getString("respSql"));
    switch (view.getId())
    {
       case R.id.btnSig:
        apuntRes++;
           if(apuntRes>=cantRel)
               apuntRes=0;
           imprimirResultado(bundle.getString("respSql"));
        break;

        case R.id.btnAnt:
            apuntRes--;
```

```
            if(apuntRes<=0)
                apuntRes=cantRel-1;
            imprimirResultado(bundle.getString("respSql"));
            break;
        }

    }
}
```

14.3.10 Layouts XML del MainActivity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context="com.tutos.elvis.androidvolley.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Serial" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/eSerial" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Orden de PO"/>
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/ePOrder"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/btnAceptar"
        android:text="Mostrar Tabla"
        android:layout_gravity="center"
        android:onClick="btnAceptar"/>
```

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="btnBuscar"
        android:text="Buscar por serial" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="btnBuscarpo"
        android:text="Buscar por PO" />

</LinearLayout>
```

14.3.11 Layouts XML del PresentacionLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textColor="@color/colorOscuro"
            android:text="Orden de Compra:" />

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Sin resultados" />
    </LinearLayout>
```

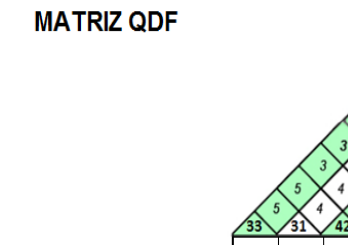
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorOscuro"
        android:text="Modelo:                    " />

    <TextView
        android:id="@+id/textView10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sin resultados" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/textView13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorOscuro"
        android:text="Serial:                    " />

    <TextView
        android:id="@+id/textView12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sin resultados" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/textView24"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorOscuro"
        android:text="Ubicación:              " />

    <TextView
        android:id="@+id/textView22"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sin resultados" />
</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/textView34"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorOscuro"
        android:text="Entrega:                  " />

    <TextView
        android:id="@+id/textView32"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Sin resultados" />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/textView44"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textColor="@color/colorOscuro"
        android:text="Cliente:                  " />

    <TextView
        android:id="@+id/textView42"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:layout_weight="1"
            android:text="Sin resultados" />
    </LinearLayout>

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:srcCompat="@drawable/adv1" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <Button
            android:id="@+id/btnAnt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Anterior" />

        <Button
            android:id="@+id/btnSig"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Siguiente" />
    </LinearLayout>


</LinearLayout>
```

11.3.1. Matrices QFD.

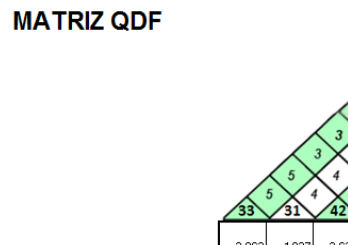
MATRIZ QDF



Comos / QUÉ	Importance	Funcionamiento de la red LAN	Enlace de datos con el computador que aloja la BD	Conexión a la base de datos	Recibirs entencias SQL en el motor de MySQL	Envío de sentencias de SQL en el motor de MySQL	Funcionamiento del SCRIPT en PHP	Funcionamiento de los eventos de la interfaz	Acceso a la base de datos por usuario y contraseña	Funcionamiento del JDBC driver o librería de acceso	Funcionamiento por medio de un computador o app movil	Mejoramiento	
		2,062	1,937	2,625	2,5	1,812	1,75	1	1,625	1,562	1,75	Evaluación de competencias (Sist.Nvo VS Sist.Ant)	
												1 Peor / 2 / 3 / 4 / 5 Mejor	
Ingreso de equipos.	5	5	5	4	4	5	4	5	5	5	5	x	4
Egreso de equipos	5	5	5	4	4	5	4	5	5	5	5	x	4
Reubicación de equipos	5	5	5	4	4	5	4	5	5	5	5	x	4
Busqueda de equipos	5	5	5	4	4	5	4	5	5	5	5	x	5
Mostrar inventario	5	4	4	4	5	4	4	4	5	5	5	x	4
Protección de información	4	5	5	5	4	4	4	5	5	4	4	x	5
Compatibilidad con otros equipos	4	5	5	5	4	3	4	4	4	5	5	x	4
Compatibilidad con dispositivos móviles	4	5	4	4	3	4	5	4	4	3	5	x	5
Interfaz de fácil uso	3	4	4	3	4	4	2	5	5	4	4	x	5
Interfaz llamativa	3	4	4	3	4	4	2	5	5	4	4	x	5
Fácil manipulación del sistema	4	5	4	4	4	5	3	4	5	4	5	x	4
Importancia del producto		224	216	190	189	208	176	218	227	213	225		
Ponderación Absoluta		461,89	418,39	498,75	472,5	376,9	308	218	368,88	332,71	393,8	207	
Ponderación Relativa		2,1188	1,9192	2,2878	2,1674	1,7289	1,4128	1	1,6921	1,5262	1,806		
Importancia		3	4	1	2	6	9	10	7	8	5		

Ponderaciones
1
2
3
4

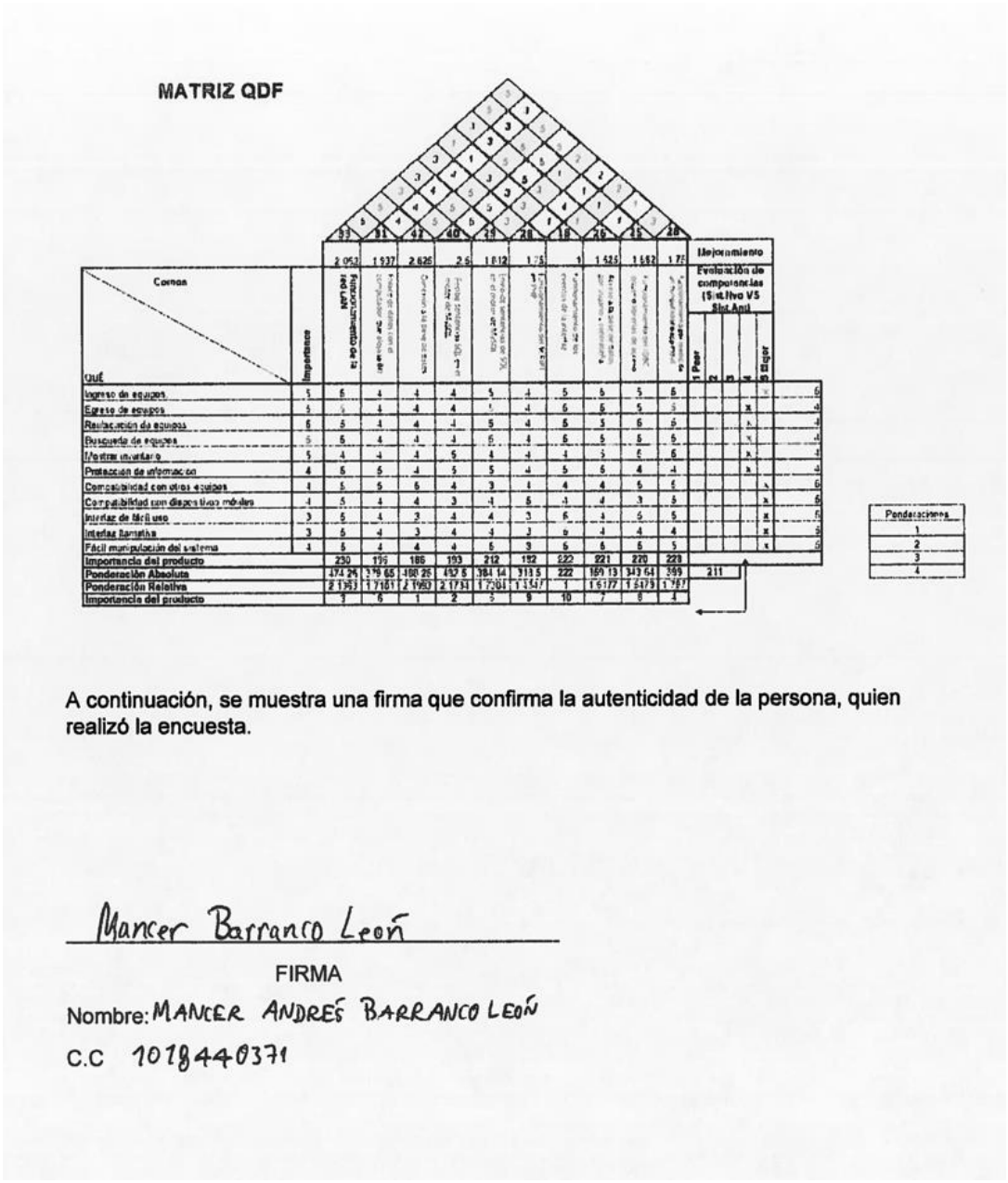
MATRIZ QDF



Comos / QUÉ	Importance	Funcionamiento de la red LAN	Enlace de datos con el computador que aloja la BD	Conexión a la base de datos	Recibirs entencias SQL en el motor de MySQL	Envío de sentencias de SQL en el motor de MySQL	Funcionamiento del SCRIPT en PHP	Funcionamiento de los eventos de la interfaz	Acceso a la base de datos por usuario y contraseña	Funcionamiento del JDBC driver o librería de acceso	Funcionamiento por medio de un computador o app movil	Mejoramiento	
		2,062	1,937	2,625	2,5	1,812	1,75	1	1,625	1,562	1,75	Evaluación de competencias (Sist.Nvo VS Sist.Ant)	
												1 Peor / 2 / 3 / 4 / 5 Mejor	
Ingreso de equipos.	5	5	4	4	4	5	4	5	5	5	5	x	5
Egreso de equipos	5	5	4	4	4	5	4	5	5	5	5	x	4
Reubicación de equipos	5	5	4	4	4	5	4	5	5	5	5	x	4
Busqueda de equipos	5	5	4	4	4	5	4	5	5	5	5	x	4
Mostrar inventario	5	4	4	4	5	4	4	4	5	5	5	x	4
Protección de información	4	5	5	4	5	5	4	5	5	4	4	x	4
Compatibilidad con otros equipos	4	5	5	5	4	3	4	4	4	5	5	x	5
Compatibilidad con dispositivos móviles	4	5	4	4	3	4	5	4	4	3	5	x	5
Interfaz de fácil uso	3	5	4	3	4	4	3	5	4	5	5	x	5
Interfaz llamativa	3	5	4	3	4	4	3	5	4	4	4	x	5
Fácil manipulación del sistema	4	5	4	4	4	5	3	5	5	5	5	x	5
Importancia del producto		230	196	186	193	212	182	222	221	220	228		
Ponderación Absoluta		474,26	379,65	488,25	482,5	384,14	318,5	222	359,13	343,64	399	211	
Ponderación Relativa		2,1363	1,7101	2,1993	2,1734	1,7304	1,4347	1	1,6177	1,5479	1,797		
Importancia del producto		3	6	1	2	5	9	10	7	8	4		

Ponderaciones
1
2
3
4

11.3.1. Evidencia de matrices QFD



A continuación, se muestra una firma que confirma la autenticidad de la persona, quien realizó la encuesta.

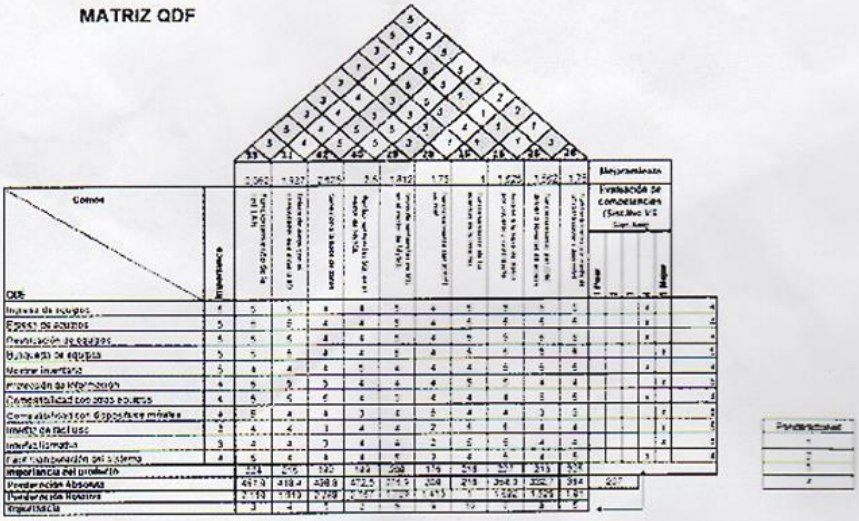
Mancer Barranco León

FIRMA

Nombre: MANCER ANDRÉS BARRANCO LEÓN

C.C 1018440371

MATRIZ QDF



A continuación, se muestra una firma que confirma la autenticidad de la persona, quien realizo la encuesta.

FIRMA

Nombre: JUAN MIGUEL RODRIGUEZ MORADA

C.C 1030576638