



Tecnológico Nacional de México

Instituto Tecnológico de Veracruz

Desarrollo de un Sistema para el Rastreo  
del Transporte Urbano basado en  
sensores GPS: Módulo de Visualización  
de Información

T E S I S

QUE PARA OBTENER EL TÍTULO DE:  
Ingeniero en Sistemas Computacionales

PRESENTA:  
Emilio Ernesto Hernandez Huerfano

Asesor:  
Dr. Rogelio Hasimoto Beltrán

Coasesor:  
Dr. Rafael Rivera López

2018

# Resumen

A medida que la tecnología avanza, la calidad de vida de las personas y el ambiente en que se desenvuelven se ven mejorados, desde la manera en que se comunican e interactúan hasta el proceso mediante el cual aprenden. Uno de los cambios más notables en los últimos años ha sido el desmesurado aumento en el uso de smartphones<sup>1</sup> como una herramienta indispensable para el desarrollo de cualquier actividad del día a día, dígase comunicarse con otros a través de llamadas, videoconferencias o mensajes de texto, ejercitarse, tomar fotografías y videos, medir el ritmo cardíaco, ubicar a otros en un mapa, etc.

El presente trabajo de tesis describe parte de un proyecto de software que se está desarrollando en el Centro de Investigación en Matemáticas, A. C. (CIMAT), cuyo objetivo en conjunto es aportar una actualización o mejora al sistema de transporte urbano basada en el uso de dispositivos móviles y tecnologías o estándares web. En la actualidad un alto porcentaje de personas cuentan con un teléfono inteligente o una computadora personal, siendo usuarios del internet y de los sistemas de comunicación digital [INEGI, 2017]. Dicho objetivo pretende ser alcanzado a través de la implementación de un sistema de software para monitorear el servicio de transporte urbano de cualquier ciudad o población, en especial aquellas en las que el servicio presenta deficiencias asociadas a la mala administración de las unidades que circulan las rutas de autobuses, o de la incapacidad para cubrir los requerimientos de unidades.

En particular, esta tesis trata acerca de dos componentes que se encargan de la interacción con el usuario final, una aplicación para dispositivos móviles y una aplicación para la web que comparten funciones muy similares. Ambas aplicaciones se encargan de desplegar información de interés para las personas que utilizan el servicio de transporte, como las rutas de la ciudad de manera gráfica, la distribución de los parabuses oficiales para cierta ruta, la ubicación en tiempo real del grupo de

---

<sup>1</sup>Traducción al inglés de *Teléfono Inteligente*.

autobuses que la circulan en una ruta y de estimaciones de distancias y tiempos de llegada.

Esta tesis se divide en 4 capítulos, que son descritos a continuación:

En el **capítulo 1** se introduce al lector de manera breve al problema que se está abordando, describiéndolo de manera concisa, mencionando algunos de sus antecedentes, exponiendo los objetivos que se pretenden alcanzar y planteando las hipótesis ideadas a partir del análisis del problema en cuestión.

El **capítulo 2** presenta las bases teóricas que fundamentan este trabajo y que son necesarias para la comprensión de éste por parte del lector. Entre los conceptos tratados se tienen en su mayoría términos técnicos relacionados a los lenguajes de programación, tecnologías y formatos utilizados en el desarrollo del software y algunos otros que tratan acerca de las bases geográficas del Sistema de Posicionamiento Global (GPS) utilizado para el rastreo de las unidades de transporte público.

En el **capítulo 3** se expone una descripción a grandes rasgos del sistema, su fundamentación en los requisitos y necesidades expuestos por parte de los usuarios, los módulos arquitectónicos, el diseño que se propuso con base en los requerimientos y el modelo de datos básico sobre el que fue implementado.

El **capítulo 4** describe a detalle el desarrollo de las aplicaciones móvil y web que fungen como capa de interacción con el usuario final. Expone información como los requerimientos que satisfacen de primera mano, la arquitectura y modelo empleados para su desarrollo, los patrones de diseño utilizados y el esquema lógico de flujo de datos sobre el que se basan. Finalmente se muestran algunas pruebas de implementación de ambas versiones en diferentes escenarios de interacción.

# Índice general

<b>1. Marco Metodológico</b>	<b>1</b>
1.1. Antecedentes . . . . .	1
1.2. Planteamiento del problema . . . . .	3
1.3. Objetivos . . . . .	4
1.3.1. Objetivo general . . . . .	4
1.3.2. Objetivos específicos . . . . .	4
1.4. Justificación . . . . .	5
1.5. Hipótesis . . . . .	5
1.6. Alcances y limitaciones . . . . .	6
<b>2. Marco Teórico</b>	<b>7</b>
2.1. Conceptos y herramientas geográficas . . . . .	7
2.1.1. GPS . . . . .	8
2.1.2. Geoposicionamiento y técnicas para dispositivos móviles . . . . .	9
2.1.3. Sistemas de información geográfica . . . . .	10
2.2. Sistemas distribuidos . . . . .	12
2.2.1. Definición y elementos . . . . .	12
2.2.2. Modelos arquitectónicos para sistemas distribuidos . . . . .	13
2.3. Protocolos de comunicación . . . . .	16
2.3.1. IP como protocolo de la capa de red . . . . .	17
2.3.2. Protocolos de la capa de transporte . . . . .	18
2.3.3. Protocolos de la capa de aplicación . . . . .	20
2.4. Paradigmas, arquitecturas y patrones de diseño de software. . . . .	22

2.4.1.	Paradigma orientado a servicios	22
2.4.2.	Patrones arquitectónicos	25
2.4.3.	Patrones de diseño	28
2.5.	Base de datos	31
2.5.1.	Definición y objetivo	31
2.5.2.	Tipos de bases de datos	32
2.6.	Lenguajes y tecnologías	35
2.6.1.	Sistema Android	35
2.6.2.	Java	36
2.6.3.	XML	37
2.6.4.	Formato JSON	40
2.6.5.	HTML	40
2.7.	Comentarios finales	42
<b>3.</b>	<b>Análisis y Diseño del Sistema</b>	<b>43</b>
3.1.	Requerimientos de desarrollo	43
3.2.	Abstracción del modelo de datos	45
3.3.	Módulos principales del sistema	47
3.3.1.	Recolección de datos geográficos	48
3.3.2.	Lógica del servidor	50
3.3.3.	Visualización de la información	53
3.4.	Comentarios finales	54
<b>4.</b>	<b>Implementación del Módulo de Visualización (Móvil y Web)</b>	<b>56</b>
4.1.	Interacción con el usuario final	56
4.2.	Obtención de los datos	58
4.3.	Arquitectura de la aplicación móvil	60
4.3.1.	Flujo de datos basado en eventos	65
4.4.	Implementación en el ambiente web	65
4.5.	Implementación del módulo de visualización (móvil y web)	67
4.6.	Comentarios finales	72

---

**Conclusiones** 74

**Referencias Bibliográficas** 76

# Índice de figuras

1.1. Sistema de navegación GPS de un vehículo. . . . .	2
1.2. Comparativa de la deforestación del medio ambiente con el paso del tiempo. . . . .	2
1.3. Detección de derrames de petroleo sobre superficies marinas utilizando información GPS. . . . .	3
2.1. Red de satélites GPS que vuelan alrededor de la Tierra. . . . .	8
2.2. Representación de la latitud y la longitud en el planeta tierra. . . . .	9
2.3. Triangulación para la obtener de la posición de un objetivo con base en tres referencias.	11
2.4. Separación de capas de la información que un GIS provee. . . . .	11
2.5. Esquema de un sistema distribuido. . . . .	12
2.6. Esquema de un sistema que se comunica empleando una arquitectura P2P. . . . .	15
2.7. Pila de capas definidas por el modelo TCP/IP. . . . .	16
2.8. Los Sockets son el mecanismo que se utiliza como intermediario entre la capa de transporte y la de aplicación. . . . .	20
2.9. Diagrama que muestra el proceso que sigue un explorador web (cliente) para desplegar una página, comunicándose con el servidos vía HTTP. . . . .	21
2.10. Esquema de un sistema SOA básico que indica el desacoplamiento entre los servicios ofrecidos. . . . .	23
2.11. Representación abstracta de la arquitectura MVC. . . . .	27
2.12. Patrón de diseño Modelo-Vista-Presentador. . . . .	28
2.13. Descriptor UML básico de la implementación del patrón Singleton. . . . .	29
2.14. Diagrama de Clases para la situación que ejemplifica el uso del patrón Factory Method. . . . .	30
2.15. Aplicación del patrón Observer a un sistema de gráficas en tiempo real. . . . .	31

2.16. Ejemplo gráfico del almacenamiento de información en una estructura tabular. . . . .	32
2.17. Ejemplo de un diagrama Entidad-Relación. . . . .	33
2.18. Distribución y conexión entre documentos organizados en una base de datos orientada a documentos. . . . .	34
2.19. Grafo utilizado para modelar una situación real en una base de datos. . . . .	35
2.20. Diagrama que muestra la ubicación de la JVM en el proceso de ejecución de aplicaciones, permitiendo ejecutar la misma aplicación sobre cualquier plataforma.	36
2.21. Diagrama acerca del funcionamiento de un Servlet. . . . .	37
2.22. Muestra de los componentes sintácticos del lenguaje XML. . . . .	38
2.23. Ejemplo de la conversión de un documento XML a un árbol por medio del modelo DOM. . . . .	40
2.24. Gramática descriptora del formato de texto JSON. . . . .	41
3.1. Mapa de una ciudad que remarcá la ubicación de los parabuses oficiales. . . . .	45
3.2. Abstracción gráfica de la información sobre el transporte público expresada en la figura 3.1, según el esquema propuesto. . . . .	46
3.3. Simulación de la formación de rutas sobre un grafo. . . . .	46
3.4. Módulos principales del sistema de monitoreo. . . . .	47
3.5. Diagrama de flujo general de los pasos que sigue la aplicación rastreadora. . . . .	48
3.6. Ubicación de una unidad de transporte sobre un segmento de línea de la ruta. . . .	49
3.7. Interfaz REST entre el modelo de datos y los clientes que entrega información en formatos KML y JSON, según sea necesario. . . . .	50
3.8. Esquema de la entrega de información bajo demanda por rutas de transporte. . . . .	52
3.9. Despliegue de información geográfica en tiempo real sobre la web y móviles. . . .	53
3.10. Diagrama del funcionamiento general de las aplicaciones que interactúan con el usuario final del sistema. . . . .	54
4.1. Esquema de interacción entre la aplicación móvil y el usuario final. . . . .	57
4.2. Esquemas <i>Wireframe</i> móviles que plasman el contenido y estructura de las primeras escenas para el usuario. . . . .	57

4.3. Esquemas <i>Wireframe</i> móviles que representan el contenido de la última escena de interacción con el usuario. . . . .	59
4.4. Arquitectura de la aplicación móvil. . . . .	61
4.5. Esquema que plasma el proceso de activación de procesos de acuerdo a los eventos disparados por el ciclo de vida definido por Android. . . . .	62
4.6. Casos a tomar en cuenta al calcular la distancia desde la posición de un autobús a un parabús con alerta. . . . .	63
4.7. Comunicación entre procesos basada en el patrón de diseño Observer-Observable. .	65
4.8. Modelo wireframe básico de la interfaz web equivalente a la aplicación móvil. . . .	66
4.9. Árbol generado por el modelo DOM al procesar el archivo descriptor KML. . . . .	67
4.10. Vistas de la selección de la ciudad de interés. . . . .	68
4.11. Vistas de la selección de la ruta a visualizar. . . . .	69
4.12. Vistas que muestra la construcción de una ruta sobre un mapa dinámico basado en GoogleMaps. . . . .	70
4.13. Pantallas en las que se plasma la manera en que se muestra la información y opciones de un parabús seleccionado. . . . .	71
4.14. Vistas que muestran la manera en que se despliega una alerta en ambas plataformas.	72

# **Capítulo 1**

## **Marco Metodológico**

### **1.1. Antecedentes**

La tecnología GPS fue en un inicio diseñada con fines exclusivamente militares durante la época de la Guerra Fría para proveer a los sistemas de navegación de las flotas estadounidenses con estimaciones de la posición y la velocidad de los objetos que eran de interés. En el año de 1980 fue liberada de dicha exclusividad permitiendo a grupos de otras áreas experimentar y crear aplicaciones basadas en ella [MiTAC Intl, 2011].

En la actualidad el uso más común que ésta tiene se encuentra en los sistemas de navegación que se emplean en todo tipo de medios de transporte (terrestres, marinos y aéreos), permitiendo a cualquiera que cuente con un receptor GPS calcular la velocidad a la que se desplaza y la posición en la que se encuentra con gran precisión. También permite a los conductores que cuentan en su vehículo con un dispositivo de navegación, como el de la figura 1.1, calcular y seguir una ruta entre dos puntos y encontrar rutas alternas [Papinski and Scott, 2011], entre otras funcionalidades. Un claro ejemplo de su uso se encuentra en la plataforma Google Maps, la cual permite realizar muchas de las tareas antes mencionadas [Li and Zhijian, 2010].

De las diversas aplicaciones de esta tecnología destacan aquellas orientadas a la preservación del medio ambiente, al transporte y al entretenimiento. En cuanto a la preservación del medio ambiente se ha implementado para poder llevar a cabo estudios aéreos de zonas de difícil acceso, logrando así evaluar su flora y fauna, topografía e infraestructura humana [U.S. Air Force, 2016] y poder atender casos como la deforestación y la sobre población de zonas específicas (figura 1.2). También



Figura 1.1.- Sistema de navegación GPS de un vehículo.

existen receptores instalados en boyas capaces seguir el movimiento y expansión de los derrames de petróleo en el mar (figura 1.3) [Yu et al., 2018], y hay helicópteros que la aplican para determinar el perímetro de los incendios forestales de modo que el uso de los recursos contra incendios sea eficiente [Jo et al., 2003].



Figura 1.2.- Comparativa de la deforestación del medio ambiente con el paso del tiempo.

El uso del GPS ha sido adoptada por una gran variedad de áreas de estudio y solución de problemas como se mencionó en las líneas anteriores, sin embargo no solo existen implementaciones para lograr la solución de problemáticas, si no que también las hay con fines recreativos como es

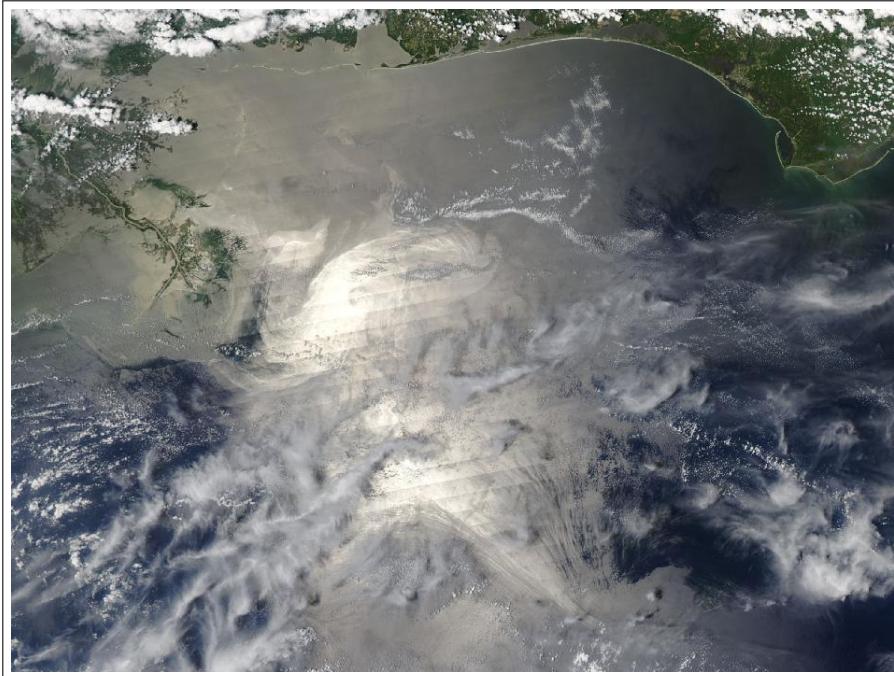


Figura 1.3.- Detección de derrames de petroleo sobre superficies marinas utilizando información GPS.

el caso del popular juego *Geocaching* [Neustaedter et al., 2013], en el que los jugadores utilizan el GPS de los dispositivos móviles para encontrar contenedores que son escondidos por otros jugadores en ciertos lugares de interés alrededor del mundo. Además, las personas cuyo pasatiempo es el excursionismo hacen uso de esta tecnología de manera similar a como lo hacen los vehículos, ya que les ayuda a asegurarse de que están siguiendo la ruta correcta y puedan marcar puntos de reunión o descanso durante el trayecto [Taczanowska et al., 2014].

Finalmente, en cuanto al transporte, el GPS es utilizado para el monitoreo de unidades que transportan diversos intereses, asegurando que la unidad sigue la dirección correcta y la mejor ruta posible, enfatizando temas de logística y seguridad.

## 1.2. Planteamiento del problema

En la rutina diaria de la mayoría de las personas se encuentra el desplazarse de un lugar a otro, ya sea desde su hogar hasta el lugar en el que trabajan o en el que estudian, para lo cual un gran porcentaje de ellos utilizan un transporte urbano, éste puede ser un autobús, un microbús e incluso un auto modelo Combi.

El uso de estos medios de transporte público presenta muchas ventajas que van desde la reducción del tráfico al no haber tantos autos particulares circulando en las calles, hasta ventajas ecológicas ya que un solo autobús puede transportar de 40 a 70 pasajeros que están evitando utilizar un auto particular y así reduciendo en gran medida las emisiones de gases contaminantes al medio ambiente. Sin embargo, este servicio de transporte presenta también ciertos inconvenientes, donde la insuficiencia en el número de unidades por ruta es uno de los más comunes, razón por la cual para el usuario es sumamente difícil predecir el momento en el que el autobús pasará por el lugar en el que él se encuentra, concluyendo en retrasos y pérdida de tiempo, ya que solo cuenta con una hora aproximada que no involucra variables como el tráfico que haya en el momento, los accidentes automovilísticos en la ruta que el autobús sigue o un cambio repentino en la ruta a causa de algún evento.

## **1.3. Objetivos**

A continuación se plantean los objetivos, tanto general como específicos, que se siguen en el desarrollo del proyecto.

### **1.3.1. Objetivo general**

Desarrollar el software necesario para lograr rastrear las unidades de transporte urbano de una localidad y proporcionar información de interés sobre éstas y las rutas que siguen a lo usuarios que cuenten con un dispositivo móvil Android o acceso a un portal de internet, para que puedan tomar mejores decisiones respecto a la planificación de su tiempo y traslado.

### **1.3.2. Objetivos específicos**

- Mostrar de manera gráfica la ruta de interés indicando la ubicación de las paradas oficiales registradas y el camino que el autobús sigue en la ruta completa.
- Permitir al usuario la visualización de los autobuses activos en tiempo real de la ruta que seleccione.

- Dar a los usuarios la capacidad de activar alarmas sobre las paradas oficiales para que se les notifique cuando un autobús de la ruta seleccionada está cerca de ella.
- Mostrar al usuario información aproximada de los tres autobuses más cercanos a la parada oficial de su elección, tal como la distancia a las que se encuentran de dicha parada, la velocidad a la que se están desplazando y el tiempo en el que llegarán.
- Implementar un buen diseño de los interfaces para el usuario utilizando las tendencias actuales para el diseño y desarrollo de software.

## 1.4. Justificación

El servicio de transporte urbano es uno de los más utilizados en todas las poblaciones del país, sin embargo, a pesar de que este servicio comenzó a ser implementado hace ya muchos años, sigue presentando algunas deficiencias para sus usuarios, la principal de ellas es el tiempo que los usuarios invierten a la espera de la unidad que los llevará a su destino. Es muy usual que no haya suficientes autobuses por cada ruta en la ciudad o población, por lo que resultan muy frecuentes los retrasos por factores como el tráfico, derivando así en pérdida de tiempo por parte de los usuarios.

El problema mencionado ya ha sido el foco de atención en algunos países de primer mundo, como Japón, Suiza, Estados Unidos y Alemania, en donde el sistema de transporte se encuentra lo suficientemente automatizado como para ofrecer a los usuarios una precisión de segundos. Sin embargo, ese no es el caso de muchas poblaciones mexicanas pequeñas.

## 1.5. Hipótesis

La hipótesis que se establece como base del desarrollo planteado en este trabajo de tesis es la siguiente:

- “*El desarrollo del sistema de rastreo propuesto reducirá el tiempo de espera de todas aquellas personas que utilizan el servicio de transporte urbano para trasladarse diariamente.*”

## **1.6. Alcances y limitaciones**

El alcance de este proyecto es el desarrollo de un sistema que facilite la rutina de las personas que utilizan el servicio de transporte urbano de manera constante, dándoles información confiable acerca del estado del autobús que esperan con respecto a la parada oficial o parabús de su preferencia.

Dentro de las limitaciones que se tomaron en cuenta durante el desarrollo del proyecto se encuentra el hecho de que en ciertas ciudades (como en la ciudad de Guanajuato) no se tiene acceso libre a algún mapa que ya tenga trazados los datos necesarios acerca de las rutas de autobuses, como lo son las paradas oficiales con nombres y los lugares por los que los autobuses pasan. También es importante mencionar que otra limitante puede ser la estructura de algunas ciudades y pueblos, puesto que los receptores GPS pierden mucha precisión al pasar por zonas con muchos árboles o bajo túneles.

# **Capítulo 2**

## **Marco Teórico**

En este capítulo se presenta una descripción de los conceptos teóricos y herramientas o tecnologías que fueron utilizadas a lo largo del desarrollo de este proyecto de tesis. Se da inicio por aquellos relacionados con la tecnología GPS y su uso en la actualidad, tales como el geoposicionamiento y los Sistemas de Información Geográfica.

Posteriormente se introducen algunos términos orientados al área de desarrollo de software, como arquitecturas, patrones de diseño, y frameworks. También se incluyen conceptos relacionados al área de diseño gráfico como modelado, maquetación y patrones visuales.

### **2.1. Conceptos y herramientas geográficas**

El área de la computación, y la tecnología en general, se ha ido mezclando con otras a lo largo del tiempo, tal es el caso de la sinergia con la geografía y la cartografía que surgió con la idea del rastreo y localización de objetos sobre la superficie terrestre, dando así origen a sistemas tecnológicos robustos que son utilizados para una gran cantidad de propósitos en todo el mundo, como la geolocalización de objetos, personas o lugares, el cálculo de rutas entre dos puntos localizados con coordenadas espaciales y la visualización, análisis e interpretación de datos geográficos [[Esri team, 2016](#)].

### 2.1.1. GPS<sup>1</sup>

El GPS, o Sistema de Posicionamiento Global en español, es un sistema satelital utilizado para conseguir posicionar un determinado objeto o región sobre la superficie terrestre [Quispe and René, 2011]. El Sistema GPS provee posicionamientos tridimensionales en tiempo real las 24 horas del día, coordenadas de navegación y tiempos mundiales. Cualquier persona que posea un dispositivo que tenga receptor GPS, ya sea dedicado como en el caso de los sistemas de navegación vehiculares, o embebido como los smartphones, puede acceder a estos servicios.

El GPS está compuesto por una constelación de poco más de 24 satélites cuya labor es la transmisión de señales de radio a los usuarios. En esta malla, 24 satélites deben encontrarse siempre activos y el resto actúan como un repuesto en caso de falla o una ventana de mantenimiento. [U.S. Air Force, 2016].

Los satélites activos antes mencionados se encuentran repartidos equitativamente entre 6 órbitas inclinadas 55° tomando como base relativa al ecuador y a una altitud aproximada de 20,200km, como se muestra en la figura 2.1, de manera que cada satélite pueda completar dos vueltas alrededor de la Tierra diariamente y así mantener disponible el servicio en casi cualquier lugar del mundo.

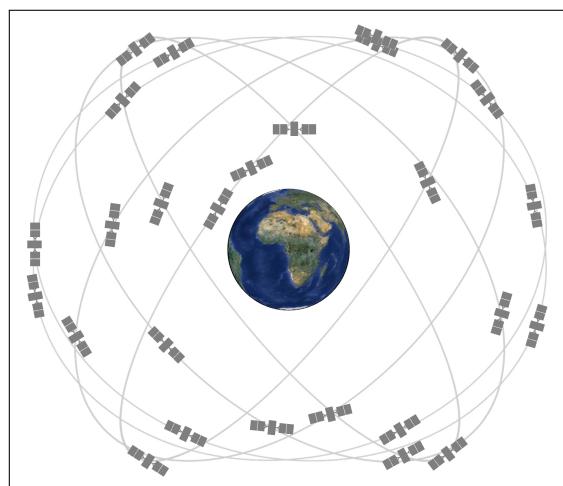


Figura 2.1.- Red de satélites GPS que vuelan alrededor de la Tierra.

---

<sup>1</sup>Por sus siglas en inglés *Global Positioning System*.

### 2.1.2. Geoposicionamiento y técnicas para dispositivos móviles

La geoposición de un objeto es la ubicación aproximada de éste sobre la superficie terrestre, se encuentra dada por un vector de tres dimensiones ( $\phi, \lambda, h$ ) [Google, 2014], a continuación una breve explicación:

- **Latitud ( $\phi$ ):** Es la distancia angular que existe desde cualquier punto de la Tierra con respecto al ecuador medida en grados, minutos y segundos (figura 2.2). Puede encontrarse en un rango de  $-90^\circ$  a  $90^\circ$  representando los valores positivos posiciones del ecuador hacia el polo norte, y los negativos posiciones del ecuador al polo sur.
- **Longitud ( $\lambda$ ):** Es la distancia angular que existe desde cualquier punto de la Tierra con respecto al meridiano de Greenwich (figura 2.2), exceptuando a los polos norte y sur, los cuales no tiene longitud. Este valor representa la posición de un punto en la orientación este-oeste sobre la superficie terrestre en un rango de  $-180^\circ$  a  $180^\circ$ .
- **Altitud ( $h$ ):** Es la distancia angular vertical de un origen dado que es considerado como el nivel 0, el cual usualmente es el nivel del mar, a un objeto en cuestión.

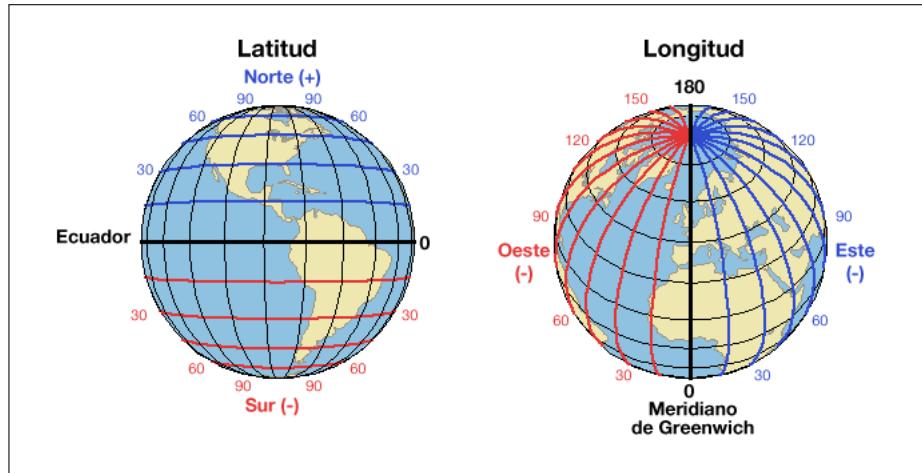


Figura 2.2.- Representación de la latitud y la longitud en el planeta tierra.

Existen varias formas para poder obtener la geoposición de un objetivo con un receptor GPS, las cuales son explicadas en las siguientes líneas:

## Consultas al sistema GPS

Este método resulta ser el más utilizado, consiste en que el dispositivo o receptor GPS ubique al menos tres satélites en órbita y envíe una serie de datos que ayuden a estos a calcular sus distancias y posiciones con respecto al mismo. Cuando éstos responden, el dispositivo es capaz de calcular una triangulación (ilustrada en la figura 2.3) y obtener su ubicación con alto nivel de precisión [Zeimpekis et al., 2002].

## Geolocalización por Wi-Fi

El sistema de posicionamiento por Wi-Fi se utiliza en casos en los que la geolocalización por GPS no es accesible, por ejemplo cuando el dispositivo se encuentra dentro de un edificio y no es capaz de detectar los satélites necesarios. Esta técnica se basa principalmente en medir el grado de intensidad con el que la señal Wi-Fi llega del Access Point (Modem) al dispositivo móvil [Zeimpekis et al., 2002], sin embargo la precisión de ésta es muy baja.

## Uso de antenas de telefonía móvil

Para localizar un dispositivo utilizando esta técnica se realiza un procedimiento muy parecido al del caso del sistema GPS, pero en lugar de emplear una red satelital se utilizan las antenas de la red de celular. Mientras un dispositivo está en movimiento puede ir conectándose a diferentes antenas de telefonía de acuerdo a la intensidad de señal que presenten, guardando así registros de las antenas a las que se ha conectado recientemente, registro que en conjunto a la posición de la antena a la que se encuentra conectado al momento puede dar como resultado una aproximación a la geoposición del dispositivo [Zeimpekis et al., 2002]. Es importante resaltar que este método resulta tener un nivel de precisión menor que los dos mencionados anteriormente.

### 2.1.3. Sistemas de información geográfica

También conocidos como *GIS*<sup>2</sup>, son sistemas de software diseñados para almacenar, recuperar, administrar y mostrar todo tipo de información geoespacial, permitiendo a los usuarios organizarla en capas (figura 2.4), realizar consultas interactivas, editar datos en un mapa y analizar la información

---

<sup>2</sup>Siglas del inglés *Geographic Information System*.

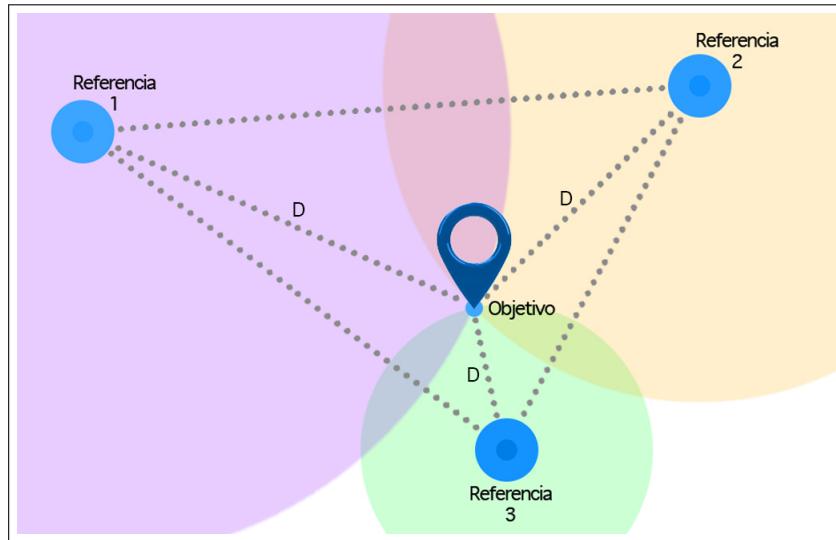


Figura 2.3.- Triangulación para la obtener de la posición de un objetivo con base en tres referencias.

disponible con el fin de gestionar situaciones que van desde encontrar rutas para conducir en la ciudad hasta el manejo de fenómenos meteorológicos [Neteler and Mitasova, 2008].

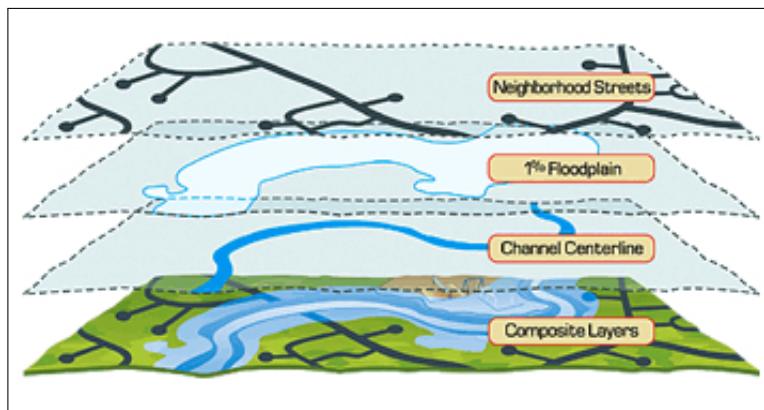


Figura 2.4.- Separación de capas de la información que un GIS provee.

Las aplicaciones de los GIS no se enfocan solamente en cuestiones de cálculo de rutas o medio ambiente y meteorología, ya que pueden ser utilizados en cualquier situación que requiera el análisis de información geoespacial. En el sector salud son utilizados para encontrar relaciones o patrones entre algunos padecimientos y enfermedades en determinada población, como los casos de cáncer pulmonar en lugares en los que se practica la metalurgia, o el gradiente socioeconómico en la mortalidad infantil [Kirch, 2008]

Las simulaciones de la superficie terrestre que un GIS es capaz de hacer, también conocidas con

el nombre de “Mapas Web dinámicos”, son construidas a partir de la técnica *Tiling*. Esta técnica consiste, a grandes rasgos, en mapear posiciones terrestres en una superficie en dos dimensiones utilizando una cuadrícula regular para posteriormente asociar una imagen (usualmente satelital) que rellene el área que cada espacio de la cuadrícula involucra [Sample and Ioup, 2014].

## 2.2. Sistemas distribuidos

### 2.2.1. Definición y elementos

Se denomina Sistema Distribuido a aquél compuesto por un conjunto de dispositivos que trabajan bajo la misma red y que ejecutan software diseñado para colaborar en diferentes tareas de manera colaborativa, compartiendo recursos e información (figura 2.5), y aportando ventajas de concurrencia y alta escalabilidad en los procesos ejecutados [Jia and Zhou, 2004].

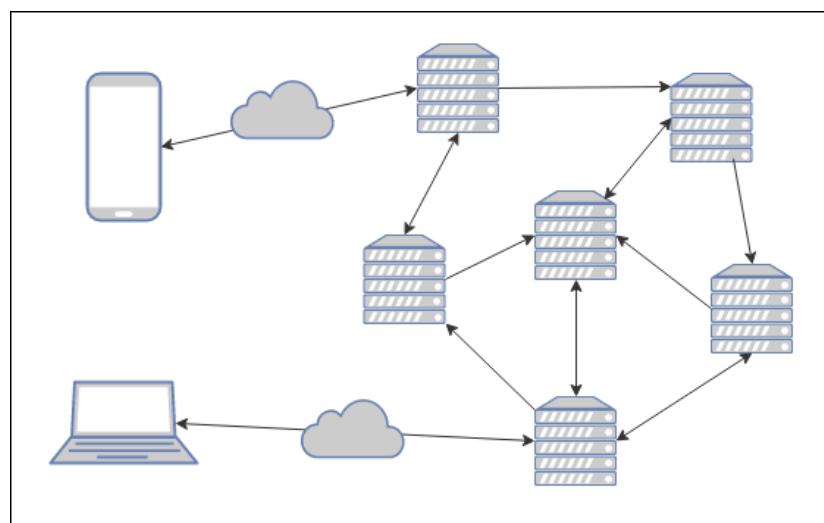


Figura 2.5.- Esquema de un sistema distribuido.

Los elementos básicos que componen cualquier sistema distribuido son los siguientes:

#### Servidor

Un servidor es un programa o proceso que ofrece algún tipo de servicio a otros procesos [Kerrisk, 2010]. Los servicios que puede ofrecer son muy diversos, que van desde el acceso a bases de datos, páginas web y correo electrónico, hasta uso de contenido especializado como información multimedia.

## Cliente

Un proceso cliente es aquél que solicita algún servicio a un proceso servidor [Shklar and Rosen, 2003]. Por ejemplo, un explorador web como Firefox o Google Chrome es un cliente que se comunica con servidores web para obtener los documentos necesarios para mostrar una página web y Thunderbird es un cliente de correos electrónicos que se conecta a un servidor de correo para solicitar correos de un usuario.

Existen dos tipos de cliente:

- **Cliente pesado:** Es aquel proceso diseñado para solicitar información específica al servidor y realizar todos los procesos de análisis y presentación por si solo [Sommerville and Galipienso, 2005]. Los juegos por computadora son un ejemplo de este tipo de cliente pues el renderizado de las escenas y movimientos son realizados por él mismo, mientras que en el servidor solo se guardan pequeños registros.
- **Cliente ligero:** Es el proceso cuyo objetivo es mandar ciertos datos al servidor para realizar todo el procesamiento necesario, retornando una respuesta concreta [Sommerville and Galipienso, 2005]. Una aplicación móvil de un banco solo envía pequeñas instrucciones y claves al servidor para que éste realice transacciones bancarias.

La comunicación entre ambos tipos de procesos se puede dar de las siguientes maneras:

- **Local:** Cuando tanto el proceso servidor como el proceso cliente están siendo ejecutados en la misma computadora.
- **Remota:** Cuando ambos procesos están siendo ejecutados en diferentes computadoras que se encuentran conectadas a la misma red, en este tipo de comunicación no importa la ubicación geográfica de las máquinas.

### 2.2.2. Modelos arquitectónicos para sistemas distribuidos

Existen varias opciones cuando de la construcción de un sistema distribuido se trata, el análisis del problema es el que definirá cual de los modelos arquitectónicos se apega más a la solución. A continuación se describen algunos de estos modelos:

## Cliente-Servidor

Este modelo arquitectónico se centra en la separación de un sistema en dos segmentos, por un lado aquellos procesos que requieren del uso de funciones o información específicas (clientes), y por el otro lado el o los procesos que proveen estas funciones e información (servidores) [Pressman et al., 2003] [Stephens, 2015]. La separación en estos dos segmentos cumple la función de disminuir el tiempo de desarrollo puesto que se puede trabajar en éstas por separado.

En un principio, era utilizado solamente para la construcción de sistemas de software centralizado, en los que una sola máquina guardaba la lógica y el modelo de datos, sin embargo después se convirtió en el *estándar de facto* para el desarrollo de plataformas distribuidas [Chung, 2000].

## Arquitectura de tres capas

La arquitectura de software basada en tres capas tuvo su auge entre los años 80's y los 90's al permitir a los desarrolladores de software separar cualquier sistema en tres segmentos principales [Liduo and Yan, 2010] [Gomaa, 2011]:

- Capa de presentación: Programa de interfaz gráfica que es ejecutado en un cliente, tomando en cuenta una distribución tipo Cliente-Servidor, y cuyo objetivo es la interacción con el usuario final.
- Capa de lógica del negocio: También llamada “Capa intermedia”, es la encargada de que el flujo de datos sea acorde a la lógica establecida, de la validación de datos de entrada y salida al sistema, y del manejo de transacciones y dependencias.
- Capa de persistencia de datos: Responsable de la interacción entre la lógica del negocio y la base de datos, contiene las reglas impuestas por el modelo de datos a fin de conservar la integridad de la información que se almacena y consulta.

## Arquitectura multicapas

Una arquitectura multicapas, también llamada de  $N$ -capas, resulta ser la generalización a la arquitectura de tres capas antes mencionada. El objetivo de esta idea es el de organizar el software de manera granular al fraccionarlo en capas y subcapas extra. La tendencia actual se evoca a esta

arquitectura con base en la facilidad para implementar sistemas distribuidos codificando capas que funcionan como interfaces de comunicación entre los componentes [Buschmann et al., 1996].

Muchos casos actualmente involucran segmentos de software destinados a la interpretación de datos, a diversas formas de comunicación a través de la red, conexión a diferentes tipos de bases de datos de manera remota y demás código fuente que fácilmente puede ser conceptualizado en capas intermedias a las clásicas.

### Arquitectura Peer-to-Peer

Mayormente conocida por el acrónimo “P2P”, es una arquitectura de software distribuida cuyo objetivo es permitir a los individuos comunicarse y compartir información entre ellos (como en la figura 2.6), sin la necesidad de un proceso que sirva de servidor central [PANDA SECURITY, 2010].

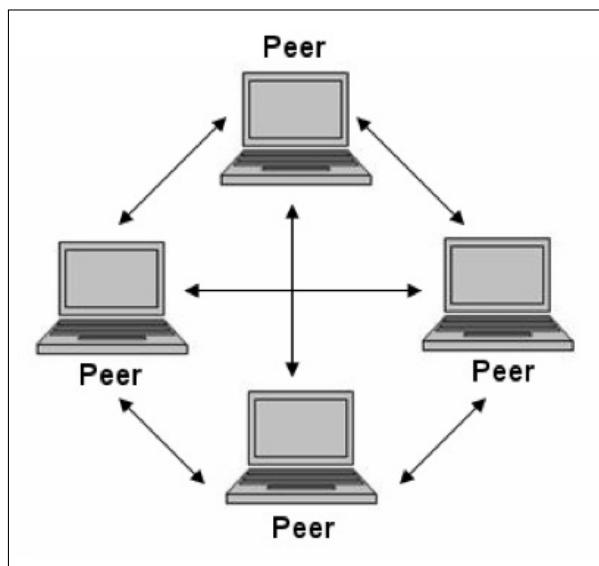


Figura 2.6.- Esquema de un sistema que se comunica empleando una arquitectura P2P.

El uso principal de esta tecnología es el intercambio de archivos a través de la red y los sistemas tipo *Torrent* son ejemplos de este caso. El sistema de videollamadas Skype emplea esta arquitectura para la transmisión de voz [Baset and Schulzrinne, 2005].

## 2.3. Protocolos de comunicación

Como se mencionó en líneas anteriores, la tendencia en el desarrollo de sistemas de software apunta a la implementación de arquitecturas y modelos distribuidos, principalmente por cuestiones de escalabilidad y disponibilidad. Para que la distribución de procesos en varios nodos de un sistema sea posible es necesaria la especificación de los protocolos de comunicación entre procesos o IPC<sup>3</sup>, los cuales definen estándares para el transporte de información entre los componentes de sistema, ya sea de manera local o remota.

Una de las propuestas más sobresalientes es el modelo TCP/IP, cuyas primeras definiciones se dieron en [Cerf and Kahn, 1974], donde se estipulan una serie de capas que indican la manera correcta de enviar información a través de la red [Fall and Stevens, 2011]. A continuación se describen las capas del modelo, cuyo orden se encuentra plasmado en la figura 2.7:

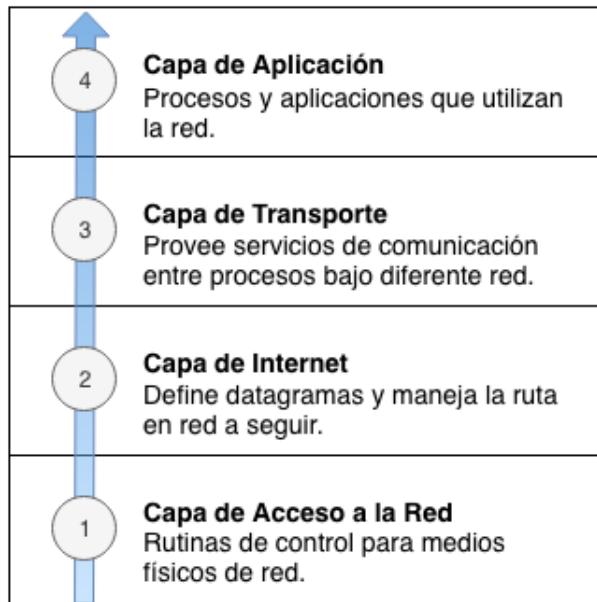


Figura 2.7.- Pila de capas definidas por el modelo TCP/IP.

**Capa de enlace y acceso a la red:** Es la capa de más abajo en la pila, detalla la manera en que la información es enviada físicamente a través de la red hacia el destino. Estos detalles involucran lo

<sup>3</sup>Siglas del inglés *Inter-Process Communication*.

relacionado al envío de cada bit de manera eléctrica u óptica y el enlace físico a utilizar como cable coaxial, fibra óptica o cable de par trenzado [Dye et al., 2007].

**Capa de internet:** Se encarga de definir un formato y codificación estándar para los datos que van a viajar a través de la red resultando en una serie de paquetes de datos conocidos como datagramas IP que contienen direcciones de origen y destino, poniendo especial atención en los métodos de fragmentación de los datos y el ruteo de estos hacia el dispositivo destino.

**Capa de transporte:** El objetivo principal de esta capa es definir un acuerdo entre el dispositivo que envía los datos y el que los recibe respecto a cómo establecer la conversación e iniciar la transmisión de datos. Usualmente esta decisión se reduce a dos opciones, tener un canal de conexión fijo (TCP) o no tener una conexión estable y enviar ágilmente la información (UDP).

**Capa de aplicación:** Ésta es la capa superior en la pila, se encarga del manejo de la representación y codificación de los datos que serán enviados, a demás de controlar el diálogo establecido entre los dispositivos. Su tarea en general es recabar información para pasarla a la capa de transporte, y leer información de esta última para presentarla a los procesos o aplicaciones pertinentes [Alani, 2014].

A continuación algunos de los protocolos más utilizados según la capa en la que se encuentran.

### 2.3.1. IP como protocolo de la capa de red

El protocolo IP<sup>4</sup> es el más importante de la capa de red, la versión que es utilizada en la actualidad es la 4, la versión 5 es utilizada solo para experimentar con algoritmos de transmisión en tiempo real y la versión 6 fue liberada recientemente pero al no ser interoperable con la versión 4 no ha sido implementada [Hunt, 1992] [Alani, 2014]. Las tareas que éste desempeña son principalmente las siguientes:

- Establecer el esquema de direccionamiento origen-destino a través de la red al adicionar una cabecera que contiene la información de direccionamiento o dirección IP, la cual es una sucesión de 32 bits que sigue un formato conocido como *dotted-decimal* [Cowley, 2013].

---

<sup>4</sup>Del inglés *Internet Protocol*.

- Administrar datagramas IP, refiriéndose a su generación, su fragmentación en partes pequeñas y fáciles de administrar y el reensamble para obtener la información original, acción que es útil cuando la información pasa por redes que especifican diferente tamaño máximo de paquete.

Un **Datagrama** es un paquete de información que consta de dos partes llamadas cabecera y cuerpo o datos. La cabecera es la que contiene la información de direccionamiento necesaria: las direcciones IP del origen y del destino, el tamaño del paquete, la versión y el protocolo utilizado. Por otra parte, la sección de los datos contiene la información a entregar al receptor.

### 2.3.2. Protocolos de la capa de transporte

La capa de transporte se enfoca prácticamente en dos protocolos: TCP<sup>5</sup> y UDP<sup>6</sup>. Éstos se caracterizan por diferir en dos variables principalmente, la velocidad de transmisión de la información y la confiabilidad con la que ésta se transfiere.

#### TCP

El Protocolo de Control de la Transferencia es considerado un mecanismo altamente confiable para la transmisión de información entre dos procesos conectados a la red, y uno de los más importantes en el mundo de las redes de computadoras. Pertenece a la rama de los protocolos conocidos como *Orientados a Conexión* puesto que para iniciar el envío de datos es necesario previamente establecer un canal de comunicación o conexión fija entre el proceso origen y el destino mediante un proceso llamado *3-way Handshake*<sup>7</sup>. El objetivo de este proceso es establecer parámetros que indican como se llevará a cabo la transmisión y la recepción de la información [Kurose and Ross, 2012].

TCP es implementado principalmente en aquellas aplicaciones en las que la pérdida de paquetes no es una opción, como en el envío de correo electrónico o de mensajes de texto. Éste ofrece un mecanismo de control de transmisión/retransmisión automática de datos en caso de fallo. Los paquetes son enviados de manera ordenada al proceso destino, y cada que un paquete es enviado se espera un paquete que confirma su recepción antes de enviar el siguiente.

---

<sup>5</sup>Siglas del inglés *Transmission Control Protocol*.

<sup>6</sup>Siglas del inglés *User Datagram Protocol*.

<sup>7</sup>Traducido al español como *Negociación en tres pasos*.

De manera predeterminada, TCP provee un canal de comunicación *full-duplex* o en dos sentidos, por lo que una vez establecida la conexión, ambos procesos quedan habilitados para enviar y recibir información cuando sea necesario, pero es posible desactivar esta función estableciendo una conexión en un solo sentido.

## UDP

El Protocolo de Datagramas de Usuarios es un estándar de facto para la construcción de redes de transmisión de datos que pertenece, de manera contraria al protocolo TCP, al conjunto conocido como *Connectionless*. Es un protocolo que no requiere de una conexión previamente establecida para transferir información de un proceso a otro. El tipo de paquete utilizado por este protocolo es conocido como Datagrama y se caracteriza por contener toda la información necesaria para identificarlo y direccionarlo a través de la red [Stevens, 1993] [Tanenbaum and Wetherall, 2010].

Se caracteriza por ser más rápido que TCP, a costa de ser mucho menos seguro. UDP no implementa ningún control de seguridad para el envío de paquetes, ni tampoco espera una señal de confirmación de recepción de estos por parte del receptor, aumentando la probabilidad de pérdida de paquetes [Stevens, 1993].

La transmisión de audio y video en tiempo real, conocida actualmente como *Streaming*, son algunas de las aplicaciones en que UDP tiene lugar, ya que toleran cierto grado de pérdida de paquetes y requieren alta velocidad de transferencia.

## Sockets

Los sockets son puertos lógicos que permiten que la información sea intercambiada entre aplicaciones o programas que se ejecutan en un sistema distribuido [Kerrisk, 2010] de manera bidireccional. Son la forma utilizada para implementar el intercambio de información basada en los protocolos TCP y UDP, situándolos como la interfaz de comunicación entre la capa de aplicación y la de transporte (figura 2.8).

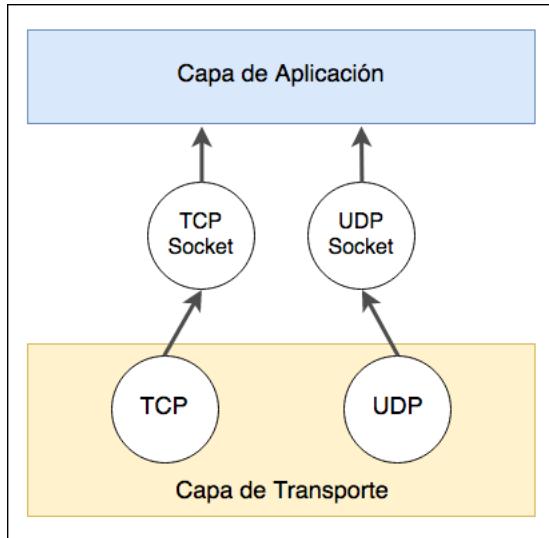


Figura 2.8.- Los Sockets son el mecanismo que se utiliza como intermediario entre la capa de transporte y la de aplicación.

### 2.3.3. Protocolos de la capa de aplicación

La capa de aplicación se encuentra relacionada a la manera en que la información transmitida interactuará con la implementación de procesos de software y los usuarios. Los protocolos de esta capa se implementan principalmente sobre TCP y UDP.

En el caso de UDP, algunos de los diseñados para utilizarlo como base son el Protocolo de Tiempo Real (RTP<sup>8</sup>) y el protocolo VoIP<sup>9</sup> para el flujo de audio. Por otro lado, entre los que se implementan sobre TCP encontramos a los protocolos HTTP<sup>10</sup>, su derivado HTTPS<sup>11</sup> y el nuevo estándar WebSocket, los cuales son explicados en las siguientes líneas.

#### HTTP

El Protocolo de Transferencia de Hipertexto es el más conocido de entre los protocolos que se pueden implementar sobre la capa de aplicación según el modelo TCP/IP. Éste fue desarrollado para ser implementado en sistemas distribuidos que trabajen de manera colaborativa puesto que, gracias a su apertura para la asignación de tipos de datos y a la negociación de la manera en que estos serán representados, permite que dichos sistemas sean desarrollados de manera independiente

<sup>8</sup>Siglas del inglés *Real-Time Protocol*.

<sup>9</sup>Del inglés *Voice over IP*.

<sup>10</sup>Siglas del inglés *Secure HTTP*.

<sup>11</sup>Contracción de *Hypertext Transfer Protocol*.

a la información que se transferirá [Fielding et al., 1999].

Un explorador Web, como Mozilla Firefox, Google Chrome o Safari por mencionar algunos, es el mejor ejemplo de implementación del protocolo HTTP, su trabajo es desplegar una página web al usuario al interpretar información que solicita al servidor empleando una o varias peticiones HTTP (figura 2.9) [Kurose and Ross, 2012].

Es importante mencionar que las peticiones HTTP que una aplicación realiza a uno o varios servidores no guardan relación alguna entre sí, razón por la cual HTTP es denominado *Stateless*, haciendo referencia a que el servidor no tiene la capacidad de guardar información de cada cliente basándose exclusivamente en las peticiones que realiza y cada petición es tomada como algo completamente nuevo.

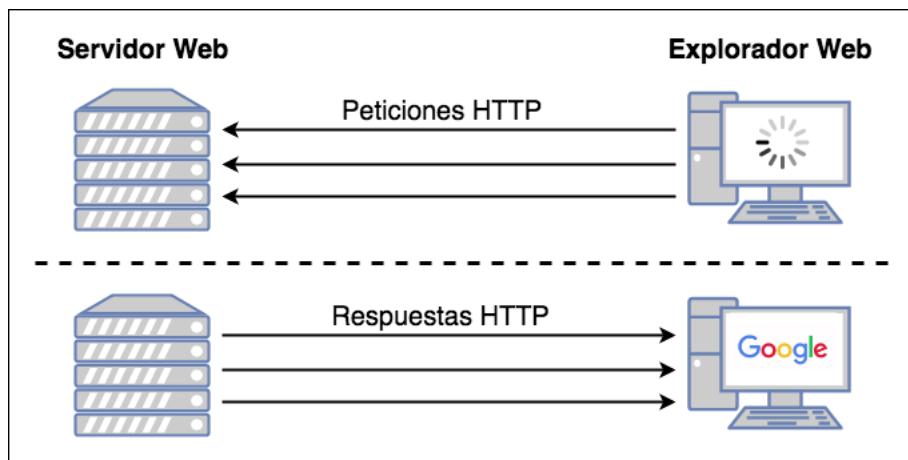


Figura 2.9.- Diagrama que muestra el proceso que sigue un explorador web (cliente) para desplegar una página, comunicándose con el servidos vía HTTP.

## HTTPS

Es la versión del protocolo HTTP que implementa ciertas medidas de seguridad al intercambiar información a través de la red. Un problema que presenta HTTP es que la información que se envía es fácilmente interpretada [Alani, 2014], aumentando la vulnerabilidad a ataques informáticos destinados a robarla. Los sistemas que manejan información sensible o transacciones interbancarias fueron los primeros en adoptar esta versión del estándar.

Durante el handshake que se dan al enviar una petición HTTPS al servidor, se automatiza la

verificación de identidad de ambos procesos y se acuerdan ciertas claves de encriptación, logrando que la información no corra el peligro de ser leída por un tercero.

## WebSocket

El protocolo WebSocket surgió a partir de la necesidad de poder establecer una conexión full-duplex para una transmisión de datos en tiempo real sin tener que acudir al mecanismo tradicional en que el cliente envía peticiones HTTP [Fette and Melnikov, 2011] [Wang et al., 2013].

Éste fue diseñado para implementarse sobre el protocolo HTTP, y por consiguiente sobre TCP, tal que pueda aportar el canal bidireccional de TCP aunado a ventajas como el filtrado, la capacidad de autenticación y de estándares de seguridad que aporta HTTP.

Al ser implementado sobre HTTP es posible, y preferible, evolucionar a la versión segura del protocolo WebSocket, conocida como WSS<sup>12</sup>, enviando información encriptada en tiempo real y evitando así que sea extraída por atacantes informáticos.

Los WebSockets son implementados usualmente en aquellas aplicaciones que requieren transferencia de información a alta velocidad, como el caso de juegos en línea y herramientas de edición de documentos colaborativos, también para interfaces gráficas que despliegan información al usuario que solicitan al servidor en tiempo real.

## 2.4. Paradigmas, arquitecturas y patrones de diseño de software.

### 2.4.1. Paradigma orientado a servicios

Cuando se plantea el desarrollo de software para una empresa de tamaño indistinto, e inclusive al tratarse de un proyecto personal, muchas veces se presenta la situación en que hay componentes que realizan tareas lógicas muy similares o inclusive iguales, sobre todo cuando se agregan módulos nuevos a una plataforma o nuevo software es desarrollado para convivir con el ya existente.

La situación anterior abrió paso a la concepción de una nueva arquitectura para el diseño de softwa-

---

<sup>12</sup>Siglas del inglés *WebSocket Secure*.

re conocida en adelante como Arquitectura Orientada a Servicios o SOA<sup>13</sup> [Intertech, 2014], diseñada para facilitar la construcción de plataformas altamente escalables y flexibles, pero favoreciendo la facilidad de su mantenimiento y actualización.

En ésta, el modelo de datos se encuentra aislado por una capa formada por pequeños componentes de software llamados servicios, los cuales se encargan de un proceso lógico específico y poseen la característica de estar desacoplados y ser autónomos (figura 2.10), por lo cual un servicio puede ser consultado por un proceso sin interferir o afectar a otros. También es posible desarrollar servicios que se comuniquen con otros para concretar una tarea u obtener información que requieran para llegar a su fin.

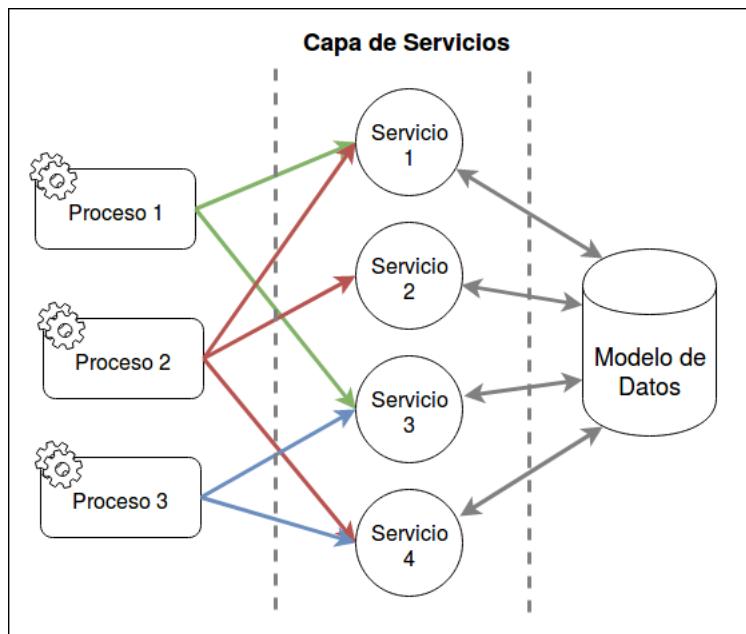


Figura 2.10.- Esquema de un sistema SOA básico que indica el desacoplamiento entre los servicios ofrecidos.

Los procesos o aplicaciones cliente, de acuerdo a la arquitectura, están diseñados para consumir estos servicios y ejecutar acciones CRUD<sup>14</sup> sobre el modelo de datos detrás de ellos. Al igual que esta arquitectura de desarrollo de software, existen otras opciones que se adecuan a otro tipo de necesidades, como el caso de las que siguen el paradigma orientado a objetos o el orientado al diseño de componentes.

<sup>13</sup>Del inglés *Service Oriented Architecture*.

<sup>14</sup>Siglas en inglés de *Create-Read-Update-Delete*

## Servicios web

Los servicios web siguen la conceptualización antes mencionada respecto a “servicio”. Son componentes desacoplados de software e independientes de plataforma que se encuentran disponibles a través del internet y, usualmente, son consultados utilizando el protocolo HTTP. Cada servicio web es identificado por medio de una serie única de caracteres llamada URI<sup>15</sup>, misma que sirve para accederlo o consultarla [Bouguettaya et al., 2014].

Como ejemplo de implementación de servicios web se puede mencionar la creación de interfaces de comunicación para sistemas legados (sistemas informáticos antiguos que pueden ser difícilmente actualizados por desarrollar tareas críticas en la lógica del negocio), ya que estos permiten que un sistema nuevo interactúe con uno obsoleto sin alterar este último [Michal Zemlicka, 2011].

## SOAP

SOAP<sup>16</sup> es un protocolo para el intercambio de mensajes en lenguaje XML que fue definido por el Consorcio WWW<sup>17</sup> siguiendo una filosofía Cliente-Servidor [Paik et al., 2017]. Su uso principal se encuentra en la implementación de arquitecturas de software basadas en servicios web.

La implementación de SOAP logra que la transferencia de mensajes entre pares se realice de manera independiente a la plataforma y al lenguaje de programación. El envío y recepción de mensajes no está limitado al protocolo HTTP, si no que también puede efectuarse sobre los protocolos SMTP<sup>18</sup> y TCP, y utilizando el mecanismo JMS<sup>19</sup>. Un mensaje SOAP se compone por los siguientes elementos:

- **Información del protocolo de transporte:** Es la cabecera del protocolo sobre el que se enviará el mensaje, usualmente HTTP, más una cabecera especial *SOAPAction* que indica que la petición es un mensaje SOAP.
- **<soapenv:Envelope>:** Es el elemento que contiene el mensaje, es conformado por dos bloques:

---

<sup>15</sup>Siglas del inglés *Uniform Resource Identifier*.

<sup>16</sup>Acrónimo del inglés *Simple Object Access Protocol*.

<sup>17</sup>Siglas del inglés *World Wide Web*.

<sup>18</sup>Siglas del inglés *Simple Mail Transfer Protocol*.

<sup>19</sup>Siglas del inglés *Java Message Service*.

- **<soapenv:Header>**: Especifica las directivas para el mecanismo SOAP, estas reglas no son para el mensaje a transferir, si no para su procesamiento en el servidor, tales como seguridad básica, expresiones regulares, configuraciones de la transacción, etc.
- **<soapenv:Body>**: Contiene los datos a enviar o consultar, así como el nombre del servicio web , parámetros necesarios para el procesamiento y nombres de métodos.

## **RESTful**

Los servicios web RESTful se encuentran basados en la ideología REST<sup>20</sup>, propuesta por Roy Fielding, que indica que todo en la red es un recurso o elemento de información que puede ser referenciado por medio de un URI y manipulado a través de peticiones del protocolo HTTP [Fielding, 2000].

En esta conceptualización, algunos de los comandos HTTP son utilizados como operadores sobre el recurso señalado por la dirección URI, como GET, POST y PUT.

Este tipo de servicios web ha tomado gran importancia en los últimos años debido a la implementación de interfaces completas de acceso a información o APIs que grandes empresas han implementado basándose en su ventaja sobre SOAP en cuanto a su velocidad de procesamiento, tal es el caso de Facebook, Twitter y Spotify. A través de dichas interfaces o servicios es posible solicitar recursos como contenido multimedia, documentos de texto, streamings, etc.

### **2.4.2. Patrones arquitectónicos**

La fase del análisis de requerimientos de software es considerada la más importante y decisiva debido a que en ésta se abstrae la arquitectura que el producto debe tener para cumplir con todos los requerimientos y alcanzar sus objetivos. Los Patrones Arquitectónicos de Software son soluciones generales a ciertos problemas o situaciones con características similares. Definen algunas reglas o restricciones cuyo objetivo es asegurar que el producto final contará con características como escalabilidad, desacoplamiento, modularidad, entre otras [Buschmann et al., 2007]. Un patrón arquitectónico se aplica usualmente sobre módulos definidos y la comunicación entre estos. El módulo de interfaces gráficas o de la vista, el de la lógica de negocio, el de servicios web o el del

---

<sup>20</sup>Siglas del inglés *Representational State Transfer*.

modelo de datos son ejemplos de estos.

### Modelo-Vista-Controlador

Usualmente abreviado solo como MVC<sup>21</sup>, es un patrón de diseño de software que se basa en la separación e interconexión de los componentes en tres categorías principales y es típicamente implementado en el desarrollo de software distribuido susceptible al cambio [Buschmann et al., 2007] y que se construye con lenguajes de programación orientados a objetos, como Java o PHP [Pitt, 2012]. La figura 2.11 muestra de manera abstracta la arquitectura MVC, a continuación la descripción de sus componentes:

- **El modelo** es el contenedor de toda la lógica de negocio del sistema, es decir, es en donde los datos que son manipulados en el sistema se almacenan, a demás de que también se especifica como es que dichos datos son almacenados y si es necesario utilizar servicios de terceros para lograr cumplir con todos los requerimientos del negocio al que va dirigido el sistema. Si en el sistema es necesario el acceso a información almacenada en alguna base de datos, el código para implementar esa tarea debe ser parte del modelo.
- **La vista** es el módulo en el que las interfaces gráficas con las que el usuario interactúa son guardadas, por ejemplo los documentos HTML, las hojas de estilo CSS<sup>22</sup>, y los archivos JavaScript. En general todo lo que interactúe y pueda ser utilizado por el usuarios puede ser guardado en el componente de vista.
- **El controlador** es el componente encargado de realizar la conexión entre el modelo y la vista. Su labor principal es el aislamiento de la lógica del negocio incluida en el modelo, de los elementos de la interfaz de usuario incluida en la vista, y la manipulación de las respuestas que el sistema dará ante la interacción del usuario con la vista. Se puede decir que este componente es el principal de los tres, ya que cuando el usuario realiza alguna petición en la vista, esta es pasada en primera instancia al controlador, el cual posteriormente ordenará alguna acción en el modelo y regresará resultados a la vista.

---

<sup>21</sup>Por sus siglas en inglés *Model View Controller*.

<sup>22</sup>Siglas del inglés *Cascading Style Sheets*.

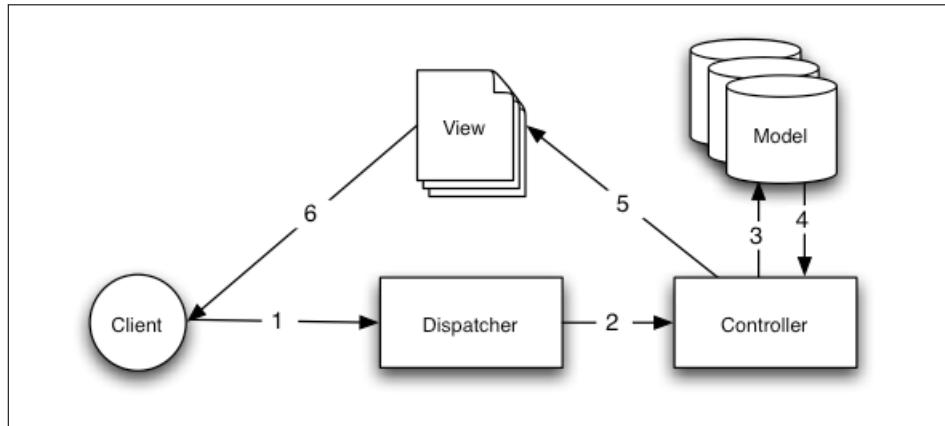


Figura 2.11.- Representación abstracta de la arquitectura MVC.

El uso de este patrón de diseño ayuda a los desarrolladores de software a mantener un código bien estructurado y limpio ya que cada componente está dedicado a ciertas responsabilidades, lo cual aporta mayor facilidad al mantenimiento o hacer pruebas en busca de errores [Pitt, 2012]. Actualmente este patrón de diseño ha tenido gran auge en los sistemas web, habiendo incluso frameworks que ayudan a los desarrolladores a organizar su código siguiendo las reglas MVC sin mucho esfuerzo como el caso de Struts2 en Java y Laravel en PHP.

### Modelo-Vista-Presentador

Es una derivación del patrón MVC que es utilizada principalmente para la construcción de interfaces móviles y el front-end de los sistemas web [Microsoft, 2017]. Se trata de organizar los componentes de código en las capas a continuación explicadas:

- **Modelo:** Usualmente es la capa de acceso a los datos, como el API de conexión con la Base de Datos o con un servidor remoto.
- **Vista:** Es la capa que se encarga de la interacción con el usuario y del paso de eventos a la capa de Presentador.
- **Presentador:** Esta capa contiene la lógica necesaria para responder a los eventos generados en la capa de Vista; también se encarga de controlar las actualizaciones sobre la capa del Modelo y de alterar el estado de la Vista según sea necesario.

El flujo de datos entre dichas capas (mostrado en la figura 2.12) inicia con la generación de eventos por parte del usuario en la capa de la vista; estos son comunicados al presentador, quien se encarga de actualizar el modelo de datos. Una vez actualizado el modelo, una serie de eventos de cambio de estado son disparados y controlados por el Presentador para realizar actualizaciones en la vista según es necesario.

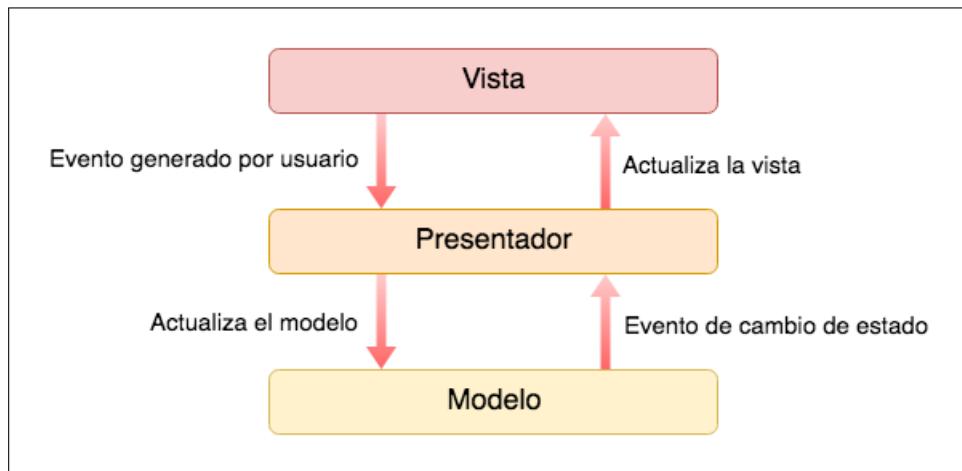


Figura 2.12.- Patrón de diseño Modelo-Vista-Presentador.

### 2.4.3. Patrones de diseño

Un patrón de diseño es un concepto a menor escala que el de patrón arquitectónico, ya que éste se centra en definir reglas que afecten a nivel de subcomponentes y objetos, asegurando un estándar para la creación de instancias y estructuras y la interacción entre estas. En [Gamma et al., 1994] los patrones de diseño son clasificados de acuerdo a su propósito como se detalla a continuación:

- Patrones creacionales: Describen pautas para la construcción de instancias u objetos.
- Patrones estructurales: Formulan reglas para la especificación de clases, sus atributos, métodos y tipos de acceso.
- Patrones de comportamiento: Describen la manera en que los objetos y estructuras deben interactuar entre ellos y la distribución de sus responsabilidades.

A continuación algunos de los patrones de diseño más importantes:

## Singleton

Este patrón de diseño (de tipo creacional) asegura que una clase tendrá solamente una instancia a lo largo de la ejecución del proceso, siendo un acceso global para todos los interactores [Gamma et al., 1994]. Lo anterior se logra configurando la clase de tal forma que sea capaz de interceptar las ordenes para crear nuevas instancias y sea un acceso a la única existente (figura 2.13).

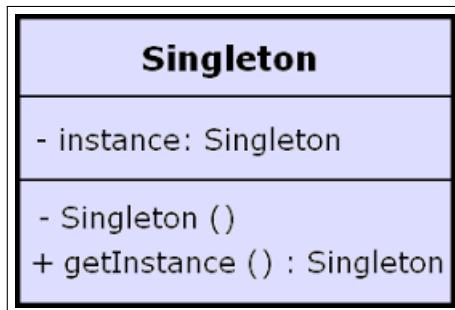


Figura 2.13.- Descriptor UML básico de la implementación del patrón Singleton.

Ejemplo de una situación en la que se puede aplicar el patrón Singleton es la programación de una interfaz para comunicarse a una base de datos, es preferible que exista una sola conexión compartida para todos los usuarios o cliente puesto que permitir conexiones separadas para los mismos puede ser muy costoso computacionalmente.

## Factory method

También conocido como “Virtual Constructor”, es un patrón de diseño de tipo creacional cuyo objetivo es la creación de una interfaz que permita construir objetos cuyo tipo será definido por la clase que la implemente, lo cual favorece que el código sea muy poco acoplado y fácil de extender.

El esquema 2.14 representa el modelo de clases para una situación en la que un proceso quiere crear instancias de diferentes tipos de computadoras (Laptop, Desktop y ServerMachine). Con base en el argumento *tipo*, será la instancia de Computadora que el método *factory()* dará como retorno, ya que las clases *Laptop*, *Desktop* y *ServerMachine* son subclases de *Computadora*.

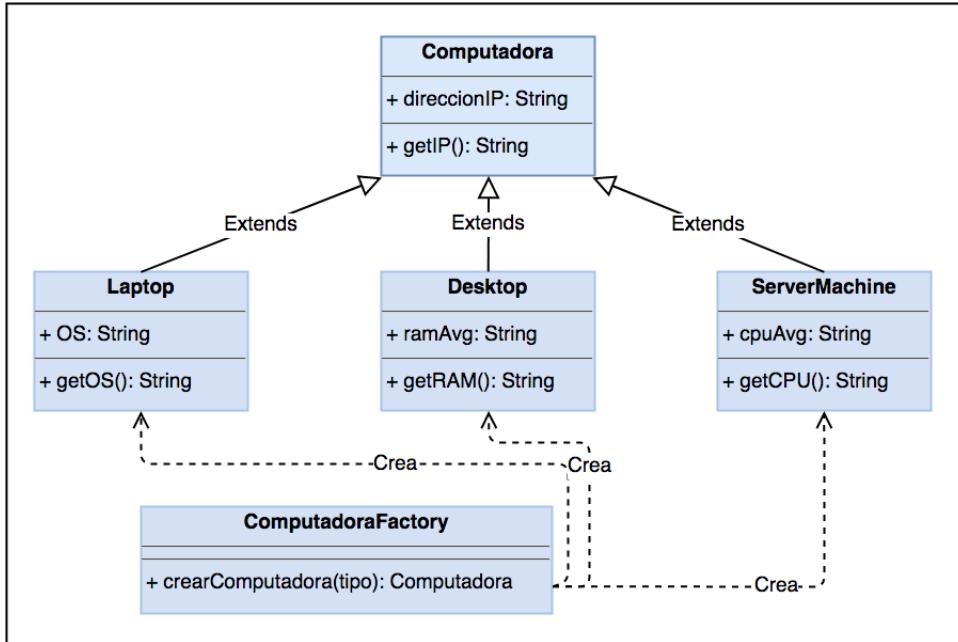


Figura 2.14.- Diagrama de Clases para la situación que ejemplifica el uso del patrón Factory Method.

## Observer

Éste es un patrón de diseño de comportamiento ampliamente utilizado en la construcción de sistemas MVC Programación Orientada a Objetos, especialmente al implementar software en el que se sigue la arquitectura MVC [Shalloway and Trott, 2004]. El objetivo que éste sigue es la definición de una relación de dependencia uno a muchos ( $1 : N$ ) entre las instancias, de tal forma que cuando una instancia cambia de estado, todas las instancias dependientes de esta son notificadas y actualizadas automáticamente [Gamma et al., 1994].

Un caso de implementación es un sistema de gráficas en tiempo real que ofrece a varios usuarios diferentes formas de visualizar los datos que se encuentran en una instancia Tabla (figura 2.15). Cuando un dato de la tabla cambia, se notifica a las gráficas que despliegan los datos al usuario y éstas deben preguntar por la modificación y actualizarse.

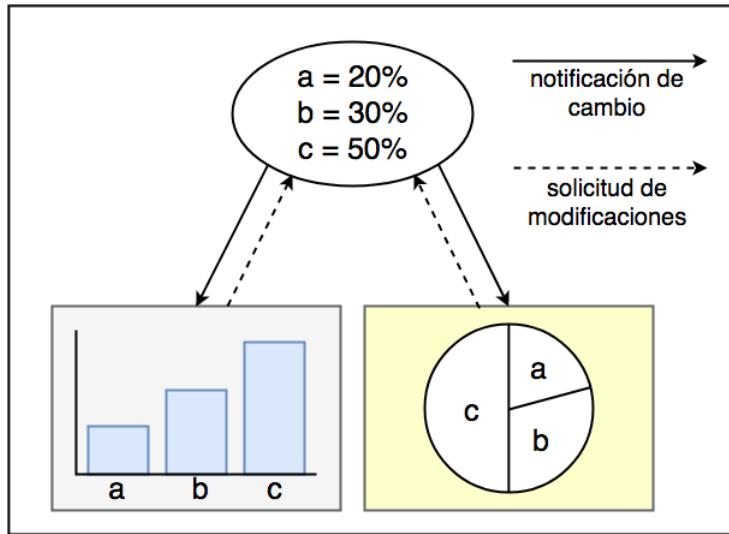


Figura 2.15.- Aplicación del patrón Observer a un sistema de gráficas en tiempo real.

## 2.5. Base de datos

### 2.5.1. Definición y objetivo

Una base de datos es una colección estructurada de datos relacionados que son relevantes para una persona o empresa [Elmasri and Navathe, 2010] [Silberschatz et al., 2006]. Usualmente se utiliza un conjunto de programas para almacenar y recuperar información de ella de manera eficiente, al cual se le llama Sistema Gestor de Base de Datos o SGBD por sus siglas.

Las empresas, los bancos, las universidades, todos utilizan bases de datos para almacenar su información de interés, crucial para su negocio o para el control de la vida laboral, desde ventas y compras, hasta número de teléfono y edades.

Un SGBD debe cumplir con tres características principales:

- Debe ser capaz de mantener la integridad de los datos que entran, a fin de que cuando necesiten ser recuperados no se vean alterados.
- Debe manejar cierto nivel de seguridad para evitar la extracción de información por puertas traseras del sistema.
- La cantidad de información que se almacenará en ella puede variar, por lo que debe contar con los mecanismos y estructuras de datos necesarios para soportar cada caso.

## 2.5.2. Tipos de bases de datos

La conceptualización de una base de datos está dividida, en primera instancia, en dos grupos: Relacionales y No Relacionales.

### Bases de datos relacionales

Las bases de datos que siguen un paradigma relacional, también conocidas como SQL<sup>23</sup> por el lenguaje utilizado para su manipulación, son las más populares en la industria del desarrollo de sistemas de software. En éstas, el almacenamiento de la información se basa en estructuras tabulares en donde cada fila representa un nuevo registro, ejemplar o instancia, y cada columna un atributo o propiedad específica de la misma (figura 2.16) [Silberschatz et al., 2006].

#	nombre	apellido	edad	carrera	nivel
1	Emilio	Hernandez	25	Sistemas Comp.	9
2	José	Pérez	24	Bioquímica	8
3	Alexis	Figueroa	26	Mecánica	10
...	...	...	...	...	...
N	María	Gonzalez	19	Mecatrónica	5

Figura 2.16.- Ejemplo gráfico del almacenamiento de información en una estructura tabular.

El almacenamiento de información y la manipulación de los datos está regido por un esquema o estructura descriptora conocido como “Diagrama Entidad-Relación” o “E-R”. Un diagrama E-R contiene las reglas necesarias para lograr esquematizar la situación real en un modelo lógico de datos, éstas incluyen la descripción de las entidades u objetos que intervienen y sus propiedades e información de interés, a demás de la manera en que dichas entidades se relacionan e interactúan entre sí (figura 2.17) [Date, 2011].

Las bases de datos relacionales presentan ventajas asociadas a la madurez de la tecnología y confiabilidad en la integridad de los datos y operaciones realizadas sobre estos, sin embargo a su vez presentan desventajas en cuanto al rendimiento en ejecución y, la más remarcable, un bajo nivel de tolerancia a cambios y escalabilidad del modelo lógico.

---

<sup>23</sup>Siglas en inglés de *Structured Query Language*.

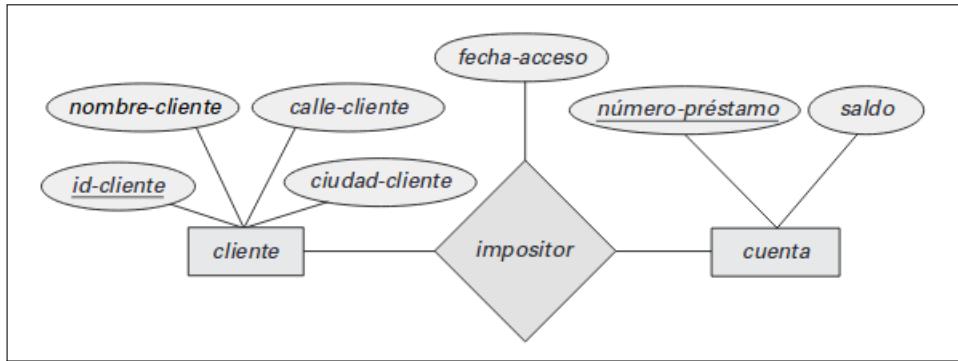


Figura 2.17.- Ejemplo de un diagrama Entidad-Relación.

La implementación de los SGBD es recomendada para situaciones en las que difícilmente se suscitarán cambios en el modelo y cuando los datos deben ser consistentes sin dar posibilidad al error. Ejemplo de dichas situaciones son los sistemas bancarios o de manejo administrativo de las empresas.

### Bases de datos no-relacionales

Son también conocidas como Bases de Datos NoSQL debido a que utilizan otros lenguajes o métodos para el almacenamiento y manipulación de los datos. En este grupo se encuentran aquellas que no se basan en estructuras tabulares para el estructuramiento de la información, si no que utilizan diversos esquemas, a continuación se listan algunos ejemplos:

#### ■ Bases de datos orientadas a pares llave-valor

Basadas en una tabla Hash, estas tecnologías son las más sencillas entre las NoSQL ya que un cliente solamente puede obtener el valor al que apunta una llave, asignarle a ésta un valor o eliminarla de la base de datos. Usualmente, el tipo de valor que almacenan es binario, razón por la cual el mecanismo de almacenamiento guarda todo de la misma manera, logrando un alto nivel de rendimiento y procesamiento [Sadalage and Fowler, 2012]. Riak KV y Apache Cassandra son tecnologías de este tipo.

#### ■ Bases de datos orientadas a documentos

Se basan en el almacenamiento de información en documentos empleando formatos como XML, JSON<sup>24</sup> o BSON (versión binaria de JSON). Los documentos generados son

<sup>24</sup>Acrónimo del inglés *JavaScript Simple Object Notation*.

organizados utilizando una estructura de árbol jerárquico compuesto por mapas de datos, colecciones y valores escalares, y relacionados con identificadores o llaves únicas (figura 2.18) [Sadalage and Fowler, 2012]. En este paradigma, cada documento almacenado representa una instancia de algún objeto lógico y es diferente de otro, a pesar de que los documentos tengan cierto parecido en estructura y formato. MongoDB es actualmente la opción principal cuando se necesita un esquema de este tipo.

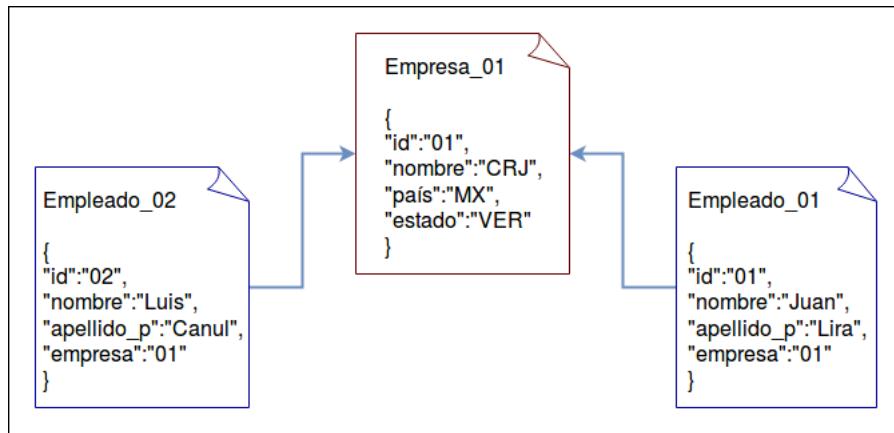


Figura 2.18.- Distribución y conexión entre documentos organizados en una base de datos orientada a documentos.

#### ■ Bases de datos orientadas a grafos

Son sistemas diseñados para organizar la información en estructuras conocidas como grafos, representándola con nodos, aristas y propiedades (figura 2.19). El uso de estas bases de datos está orientado a casos en que la información a almacenar está fuertemente ligada y expuesta a cambios constantes como la que se almacena en las redes sociales. Son empleadas en el análisis de grandes cantidades de datos (BigData), puesto que al ser un grafo es posible aplicar algoritmos optimizados para la búsqueda sobre dichas estructuras [Robinson et al., 2013]. El sistema Neo4J, originalmente diseñado para trabajar con el lenguaje Java, es un excelente referente de un SGBD basado en grafos.

La ventaja principal de los SGBD NoSQL es la flexibilidad que aportan en cuanto a los datos que se almacenan, puesto que no necesitan una serie de reglas que indique estrictamente el tipo de información a almacenar y como se relaciona, propiciando el desarrollo de un sistema mantenable y altamente escalable [Redmond and Wilson, 2012] [Sadalage and Fowler, 2012]. No obstante, son

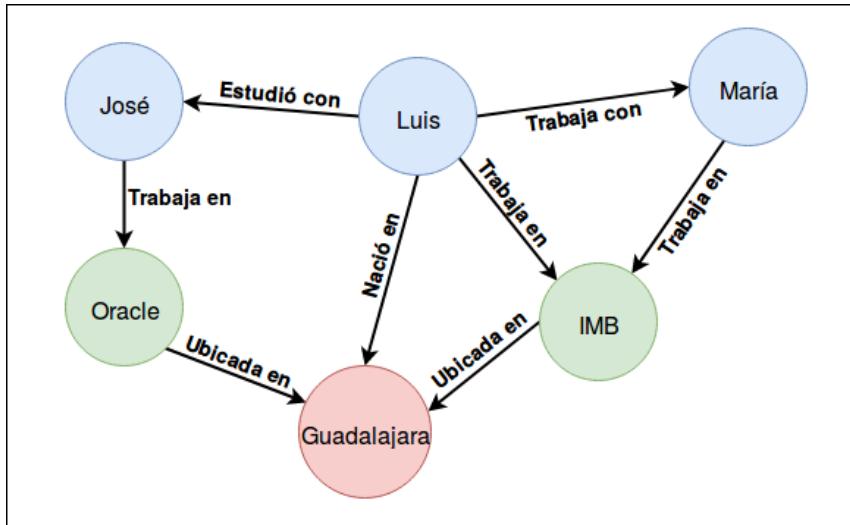


Figura 2.19.- Grafo utilizado para modelar una situación real en una base de datos.

tecnologías y conceptos emergentes comparados con las bases de datos relacionales, el nivel de madurez es mucho menor y algunos casos pueden llegar a presentar un alto nivel de riesgo en cuanto a la integridad de los datos y la seguridad.

## 2.6. Lenguajes y tecnologías

### 2.6.1. Sistema Android

Es un sistema operativo de código abierto orientado principalmente al manejo de smartphones, aunque recientemente ha sido implementado en otros tipos de dispositivos como relojes (smartwatch), televisiones (smart-TV) y controladores domóticos.

Es la plataforma para dispositivos móviles con mayor importancia en el mercado, abarcando más del 85 % y dejando atrás otras plataformas como iOS de Apple, con poco menos del 10 %, y Windows Phone de Microsoft con menos del 5 % [Statista, 2017].

La programación de aplicaciones móviles para esta plataforma se realiza utilizando el lenguaje de programación Java, aunque en el año 2017 se anunció el lenguaje Kotlin como una opción [Android, 2017].

## 2.6.2. Java

Java es un lenguaje de programación orientado a objetos empleado principalmente en el desarrollo de sistemas distribuidos de alta escalabilidad y arquitecturas para servicios y aplicaciones web. El software desarrollado con este lenguaje tiene la remarcable característica de ser independiente de la plataforma o del tipo WORA<sup>25</sup> pues es ejecutado sobre una *Máquina Virtual*, conocida simplemente como JVM<sup>26</sup>, que es una capa de software que se encuentra entre la aplicación y el sistema operativo (figura 2.20), y que está disponible para una gran cantidad de plataformas [Gosling et al., 2015].

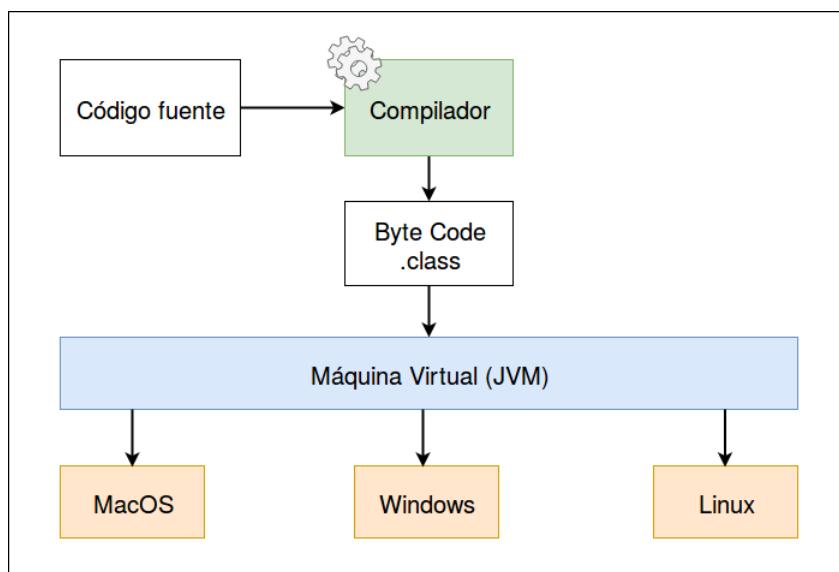


Figura 2.20.- Diagrama que muestra la ubicación de la JVM en el proceso de ejecución de aplicaciones, permitiendo ejecutar la misma aplicación sobre cualquier plataforma.

## Servlet

Un Servlet es un programa escrito en el lenguaje Java que se ejecuta del lado del servidor, actúa como una capa entre las peticiones provenientes de la interfaz gráfica con la que el usuario interactúa y las bases de datos o aplicaciones que se encuentran en el Servidor Web, tal y como se ilustra en la imagen 2.21. Es considerado como un módulo especial que sirve para extender las capacidades del servidor en cuanto al contenido dinámico.

<sup>25</sup>Siglas del inglés *Write Once, Run Anywhere*.

<sup>26</sup>Siglas del inglés *Java Virtual Machine*.

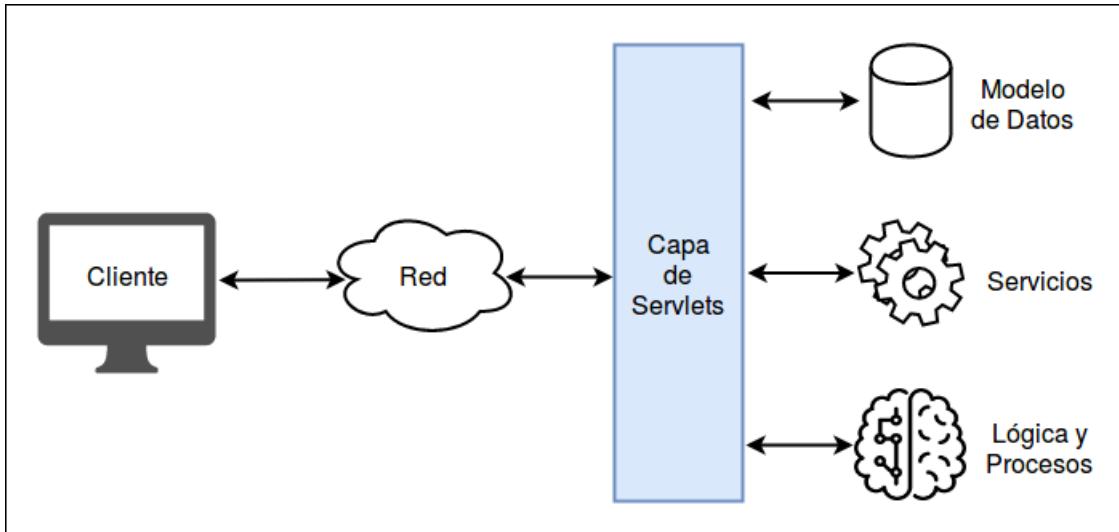


Figura 2.21.- Diagrama acerca del funcionamiento de un Servlet.

## JSP

Java Server Pages es una tecnología para el desarrollo web que permite a los programadores agregar contenido dinámico a las páginas desplegadas desde el lado del servidor, lo cual es muy útil cuando se debe mostrar información dependiente de alguna rutina o proceso lógico del flujo de datos. Un uso común es el manejo de información dependiente de sesiones como las de perfiles, carritos de compras en línea o interfaces administrativas [Zambon, 2012]. El dinamismo del contenido se consigue incrustando líneas de código Java entre el código HTML, cuando el servidor procesa el documento, interpreta las instrucciones Java y las reemplaza por el contenido correspondiente.

### 2.6.3. XML

El Lenguaje de Marcado Extensible, o simplemente XML, es un formato de texto muy simple y flexible que fue originalmente diseñado como una aportación al problema que las publicaciones electrónicas de gran tamaño representaban. Debido a la facilidad de su procesamiento, actualmente juega un papel muy importante como formato para el intercambio de mensajes de información entre procesos sobre la red [World Wide Web Consortium, 2017].

La sintaxis de este formato se muestra con un ejemplo en la figura 2.22 y se describe en las siguientes líneas:

- Un **elemento** estará compuesto por una etiqueta de apertura, el contenido de éste y una etiqueta

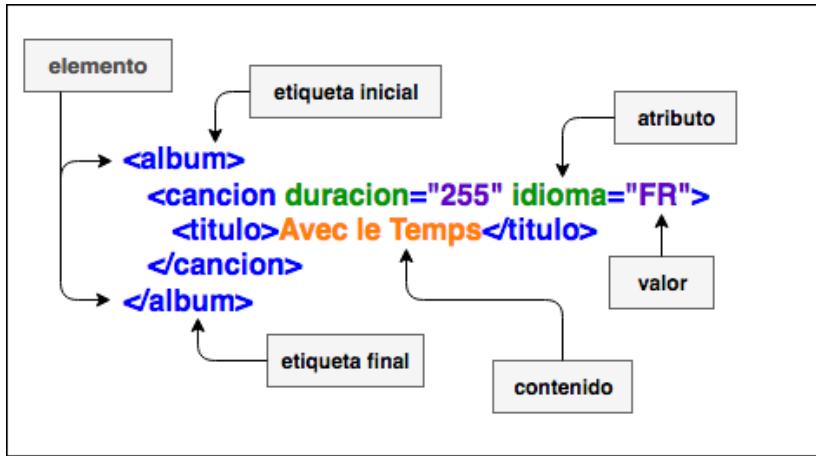


Figura 2.22.- Muestra de los componentes sintácticos del lenguaje XML.

de cierre.

- Siempre debe existir un **elemento raíz** en el documento XML.
- El **contenido** de una etiqueta puede ser una cadena de caracteres, números u otros elementos correctamente anidados.
- Un elemento puede tener uno o más **atributos**, los cuales se ubican dentro de la etiqueta de apertura. Es importante mencionar que el valor del atributo siempre debe ir entre comillas dobles.

Existen dos mecanismos para el procesamiento mensajes con este formato, a continuación su descripción:

#### **Analizador SAX<sup>27</sup>**

Este mecanismo procesa un documento o mensaje XML con base en eventos que se disparan en casos específicos de lectura:

- Inicio del documento: Es disparado solamente al inicio del procesamiento del documento XML
- Inicio de un elemento: Se dispara cada vez que una etiqueta de inicio es leída, aportando información sobre el nombre del elemento y sus atributos.

<sup>27</sup>Acrónimo de *Simple API for XML*.

- Contenido: Solo es disparado si al leer el contenido de un elemento, éste es numérico o una cadena de caracteres.
- Fin de un elemento: Es disparado cada vez que se lee una etiqueta de cierre.
- Fin de documento: Se dispara solamente cuando el procesamiento del documento XML finaliza.

La velocidad de procesamiento por parte de este mecanismo es considerablemente alta si se compara con el procesamiento DOM<sup>28</sup>, y el uso de recursos es contrastantemente bajo en la misma comparación.

## Analizador DOM

Por otro lado, cuando un documento XML es procesado con un analizador DOM, éste utiliza cierto espacio de memoria para generar una representación arborescente del documento (figura 2.23), de ese modo se pueden realizar operaciones como agregar nodos al árbol, editar los existentes o eliminarlos e inclusive obtener información a través de la aplicación de algoritmos especializados. Es importante tomar en cuenta el tamaño de los archivos XML que se procesan, puesto que es probable sobrecargar la memoria de la máquina durante la construcción del árbol.

## KML<sup>29</sup>

Desarrollado por la compañía Google, es un lenguaje de marcado descriptivo basado en la sintaxis del lenguaje XML utilizado para el almacenamiento de información geográfica tal como puntos de geoposición, líneas, capas, objetos en 3D, etc. basándose en el estándar GML<sup>30</sup> definido por el *Open Geospatial Consortium* [J. Du et al., 2009] [Ferreira et al., 2012].

Este formato descriptivo es usualmente procesado por plataformas como Google Earth y Carto, aunque es posible desarrollar *parsers* o intérpretes con facilidad.

---

<sup>28</sup>Acrónimo de *Document Object Model*.

<sup>29</sup>Siglas del inglés *Keyhole Markup Language*.

<sup>30</sup>Siglas del inglés *Geography Markup Language*.

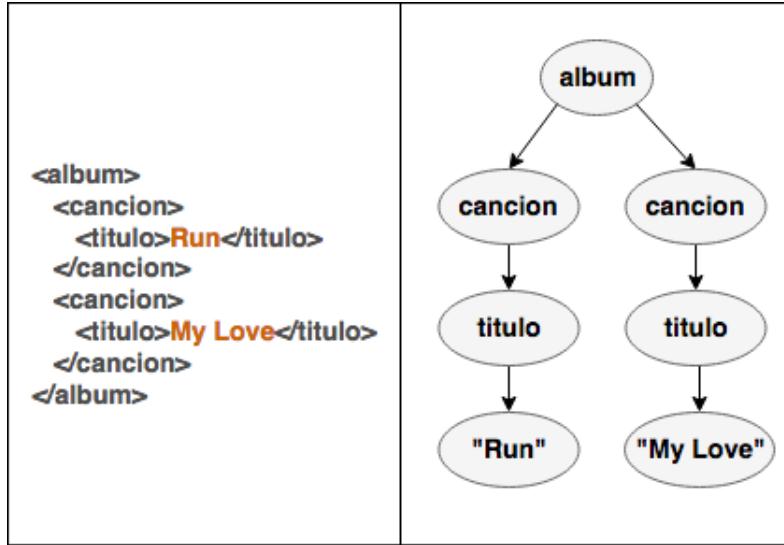


Figura 2.23.- Ejemplo de la conversión de un documento XML a un árbol por medio del modelo DOM.

#### 2.6.4. Formato JSON

Es un formato de texto ligero utilizado, al igual que XML, para el intercambio de información entre mensajes. Al transmitir la información de manera textual, este formato es independiente de la plataforma que envía o recibe el mensaje y del lenguaje de programación [JSON Organization, 2014].

El formato JSON define tres estructuras principales denotadas por la gramática de la figura 2.24, a continuación una breve explicación:

Una cadena JSON puede ser la descripción de la instancia o estado de un **objeto** o un **arreglo** de éstas, la descripción de una instancia está dada por **pares**  $\langle \text{identificador}, \text{valor} \rangle$ , en donde el *identificador* es el nombre de un atributo y el *valor* es el estado de éste al momento; la descripción de un objeto se escribe entre llaves ( { y } ) y un arreglo se escribe entre corchetes ( [ y ] ), finalmente los pares  $\langle \text{identificador}, \text{valor} \rangle$  van separados por una coma ( , ) según el número de atributos a incluir.

#### 2.6.5. HTML

El lenguaje HTML, categorizado dentro de los lenguajes de marcado basados en la estructura sintáctica de XML, es el estándar web utilizado para la descripción de la estructura y el contenido que debe desplegarse ante el usuario [Deitel and Deitel, 2007].

```

< json > ::= < objeto > | < arreglo >
< objeto > ::= '{' < miembros > '}' | '{' '}'
< miembros > ::= < par > ',' < miembros > | < par >
< par > ::= STRING ':' < valor >
< arreglo > ::= '[' < elementos > ']'
< elementos > ::= < valor > ';' < elementos > | < valor >
< valor > ::= STRING | NUMBER | < objeto > |
               < arreglo > | TRUE | FALSE | NULL

```

Figura 2.24.- Gramática descriptora del formato de texto JSON.

Los elementos descritos por el lenguaje HTML se dividen en dos grupos según la interpretación que el explorador web les da durante el procesamiento y despliegue de información [Duckett, 2014]:

- Semánticos: Son aquellas etiquetas que definen claramente su contenido, las etiquetas `<p>` utilizadas para describir párrafos, `<table>` para la descripción de una tabla y `<img>` para especificar una imagen son algunos ejemplos.
- Estructurales: Son las etiquetas que se limitan a funcionar como simples contenedores cuyo contenido no es obvio, por ejemplo las etiquetas `<div>` y `<span>`.

## CSS

El lenguaje CSS es el estándar propuesto por el W3C para la definición visual de la información que se despliega en una página web. Permite a programadores y diseñadores especificar por separado muchas características respecto a la visualización de cada elemento empleando reglas, tales como tamaño, posición, colores, animaciones, eventos, etc. La separación de dichas reglas visuales y de la estructura del contenido facilita en gran medida el mantenimiento y modificación de la página web [Deitel and Deitel, 2007].

Las reglas que describen el aspecto de los elementos de la página se componen de dos partes:

- Selector: Representa el elemento o conjunto de ellos que serán afectados por las especificaciones posteriormente señaladas y pueden ser nombres de etiquetas, clases de elementos, el atributo `id` de un elemento e inclusive nodos del árbol DOM que representa el documento HTML. El nivel de especificidad de un selector juega un papel muy importante cuando el explorador web

procesa las reglas CSS, cuanto más específico es el selector mayor será la prioridad que la regla tendrá [Meyer, 2012]. Por ejemplo, un selector que es el id de un elemento es mucho más específico que un selector que es el nombre de la etiqueta de un elemento, ya que el id señala exactamente el nodo del árbol al que se aplicarán las reglas, sin embargo el nombre de una clase señala un conjunto de nodos.

- Definición: Es una tupla atributo-valor que especifica aspectos como el largo o ancho, color, tipo de letra, etc., sobre el objeto que especifica el selector al que pertenecen.

## JavaScript

JavaScript es un lenguaje de programación que permite agregar contenido dinámico, animaciones, interactividad y efectos visuales a las páginas web [McFarland, 2011]. Es el lenguaje de programación estándar de facto para el lado del cliente debido a su alto nivel de portabilidad, cualidad multiparadigma y a su sencilla curva de aprendizaje[Deitel and Deitel, 2007].

El explorador web ya implementa una gran cantidad de acciones dinámicas de JavaScript por defecto, como el disparo de eventos de botones o del cursor, el llenado y envío de formularios al servidor web, sin embargo son funciones básicas. JavaScript permite personalizar el comportamiento de la clásica página web estática por completo [Robbins, 2012]. A pesar de ser un lenguaje de procesamiento rápido y ligero, es importante mencionar que depende de las características de la máquina cliente puesto que es ésta la que lo interpreta.

## 2.7. Comentarios finales

Derivado de los conceptos y tecnologías explicados, se puede notar que para el desarrollo del proyecto se utilizaron solamente estándares, tanto en el caso de los lenguajes y tecnologías, como en el de aplicaciones teóricas como arquitecturas de software y patrones de diseño. El uso de estándares aporta al proyecto mayor fiabilidad y robustez, debido a que un estándar que ya ha sido blanco de numerosos estudios, pruebas y validaciones de concepto, e implementado en muchas áreas y proyectos.

En el siguiente capítulo se detalla el uso y aplicación de estos conceptos en el desarrollo del sistema, y más específicamente en el desarrollo de la capa de software con la que el usuario interactúa.

# **Capítulo 3**

## **Análisis y Diseño del Sistema**

En este capítulo se introducen las bases del sistema de monitoreo, es decir, los requerimientos que fueron tomados en cuenta para el desarrollo del sistema y el bosquejo o diseño general de éste, también se presenta una breve explicación sobre el papel desempeñado por cada uno de los módulos principales que lo conforman: El módulo de recolección de datos, el de procesamiento de información y el de despliegue de información.

Es importante destacar que los detalles acerca de la implementación del módulo de recolección de datos y del de procesamiento de la información quedan fuera del alcance de este proyecto de tesis.

### **3.1. Requerimientos de desarrollo**

Para el desarrollo del sistema se tomó en cuenta la siguiente lista de requerimientos:

#### **Requerimientos de usuario**

- Se debe mostrar una lista que contenga las rutas que se encuentren disponibles en la ciudad, tal que el usuario pueda seleccionar de ella la que quiere ver.
- El usuario debe ser capaz de ver en un mapa interactivo el dibujo de la ruta que él seleccione, además de los parabuses oficiales que dicha ruta involucre.
- Los autobuses que circulan la ruta de interés para el usuario deben ser representados gráficamente en pantalla, de modo tal que informe también el sentido que el autobús lleva.

- El usuario debe poder ver información extra acerca de los parabuses y los autobuses de una ruta.
- El usuario podrá recibir notificaciones por parte del sistema en ciertas situaciones, por ejemplo el caso en el que un autobús que él espera esté a punto de llegar a la parada de autobús en la que se encuentra.
- Se debe notificar y mostrar al usuario cuando una ruta cambie por algún acontecimiento externo al sistema.

## **Requerimientos funcionales**

- El sistema debe ser capaz de dar servicio a  $N$  usuarios al mismo tiempo.
- La transmisión de datos debe ejecutarse en forma tal que todos los usuarios cuenten con información actual en todo momento.
- La arquitectura del sistema debe permitir su escalamiento a otras ciudades.

## **Especificaciones derivadas**

Derivado del análisis de los requerimientos anteriormente expuestos, se tiene que el sistema debe cumplir con las siguientes características técnicas:

- La arquitectura del sistema debe ser distribuida.
- La transmisión de datos entre los componentes del sistema debe darse en tiempo real.
- El sistema debe ser especialmente robusto en cuanto a los riesgos y fallos que pueden darse al transmitir datos, tales como la latencia en la transmisión y la desconexión inesperada de servidores y clientes.
- La precisión de la información geográfica de los autobuses debe tener un margen de error pequeño (no más de 10 metros).
- El ícono que representa a los autobuses en movimiento debe ser transformado con cada posición nueva para indicar el sentido real de éste.

- El modelo de datos implementado debe ser flexible a cambios frecuentes.

### 3.2. Abstracción del modelo de datos

Los sistemas de transporte urbano de una localidad pueden variar de acuerdo a factores como el nivel socio-económico de ésta, su tamaño y el número de habitantes. Existen ciudades muy grandes como Monterrey, Guadalajara y Ciudad de México, y también ciudades pequeñas como la capital de Guanajuato, pero en todas ellas hay un sistema de transporte urbanos que involucra autobuses. Tomando en cuenta dicha situación, el modelo de datos se basa en afirmaciones lo suficientemente generales para encajar en casi cualquier localidad, que se listan a continuación:

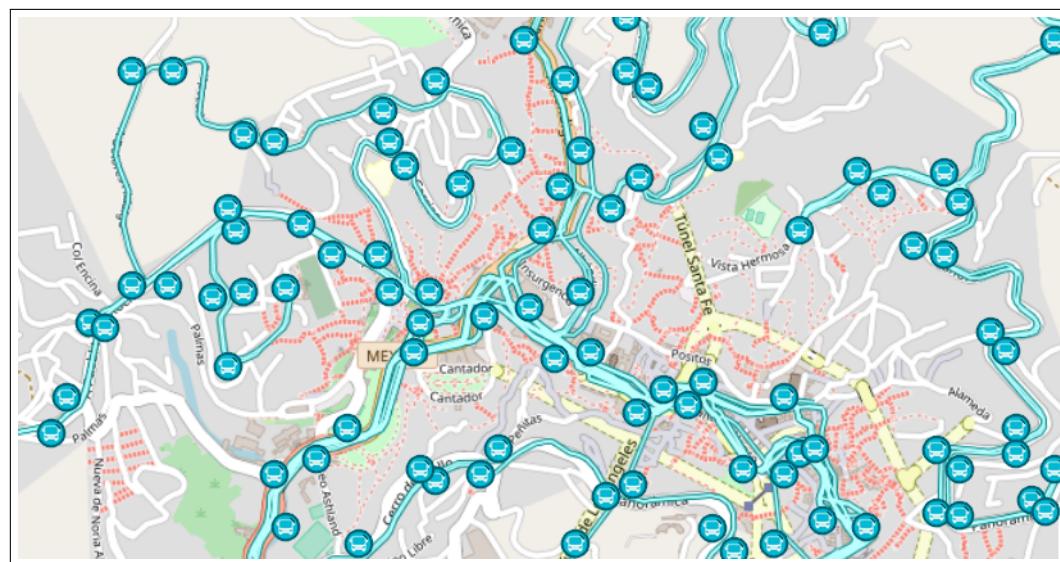


Figura 3.1.- Mapa de una ciudad que remarca la ubicación de los parabuses oficiales.

- Existe un conjunto de parabuses distribuidos por toda la ciudad (figura 3.1).
- Todos los parabuses se encuentran conectados entre sí por un camino que puede ser una calle o una sucesión de éstas, siguiendo un sentido.
- Existe un conjunto de rutas urbanas, y cada una se define como una sucesión de parabuses que forma un circuito cerrado (figura 3.3).

- Existe también un conjunto de autobuses en la localidad, recorriendo una ruta específica de manera permanente o dinámica.

De manera natural, el contexto encaja para ser modelado en un diagrama de tipo grafo dirigido (figura 3.2). En la abstracción, cada nodo representa un parabús cuya propiedad más importante es su posición GPS. Las aristas modelan la calle o camino que lleva de un parabús a otro, el peso y el sentido de éstas son la longitud en metros y el sentido descrito por el sistema vial, respectivamente.

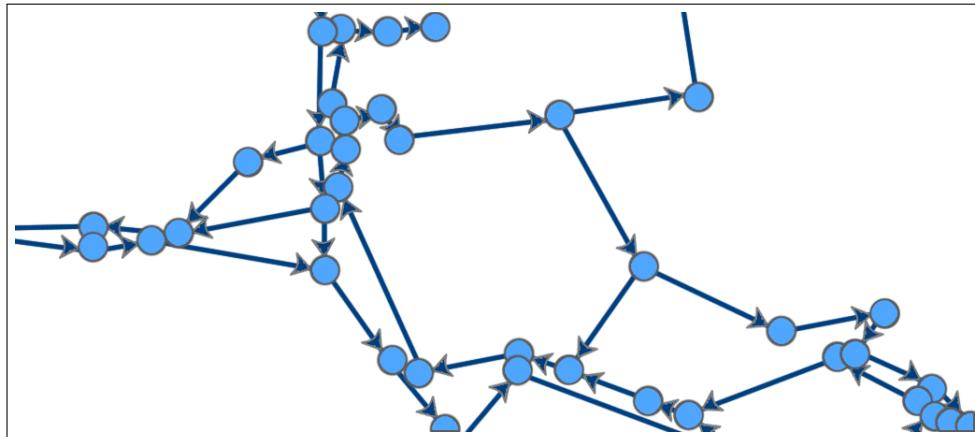


Figura 3.2.- Abstracción gráfica de la información sobre el transporte público expresada en la figura 3.1, según el esquema propuesto.

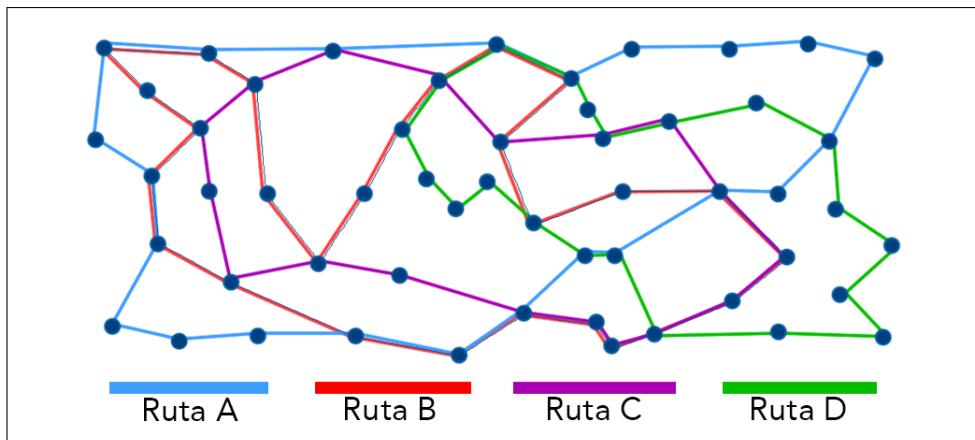


Figura 3.3.- Simulación de la formación de rutas sobre un grafo.

Por su parte, los autobuses que transitan las rutas son concebidos como agentes dinámicos externos a la estructura base del grafo, ya que sus atributos y su posición se encuentran en cambio constante (tiempo-real). Además un autobús no está ligado a una ruta específica, ya que puede ser cambiando de ruta según la demanda de usuarios durante el día.

En cuanto al manejo del modelo de datos descrito, el gestor de base de datos principal almacena instancias de rutas preprocesadas en archivos KML que contienen la definición estructural de una ruta (los parabuses, los caminos entre éstos y el sentido que siguen) para facilitar la entrega de información al cliente (móvil o web) que la solicite.

El modelo también almacena un registro de las unidades de transporte urbano que se monitorean, incluyendo el número de placa y el código de la unidad, y un registro de los conductores de las unidades (nombre completo, número de teléfono, dirección y demás datos que los concesionarios requieren), con el objetivo de enviar algunos datos a los usuarios finales y otros utilizarlos para generar reportes e históricos de utilidad administrativa.

Cabe resaltar que la abstracción del modelo de datos permite que el sistema reaccione a la situación de cambio en la estructura de alguna ruta, como cuando un parabús es eliminado de la ciudad, o cuando se suscita un acontecimiento que obliga al sistema de tránsito a proponer un camino temporal para los autobuses. Las actualizaciones solo involucran un cambio en las propiedades de los nodos o de las aristas según corresponda.

### 3.3. Módulos principales del sistema

Debido a que el sistema debe ser distribuido para dar servicio a muchos clientes, se dan a notar tres procesos principales para lograr el funcionamiento adecuado de éste. La imagen 3.4 ilustra la propuesta, donde se observa que procesos se comunican y comparten información implementando interfaces basadas en protocolos como TCP, UDP y HTTP.

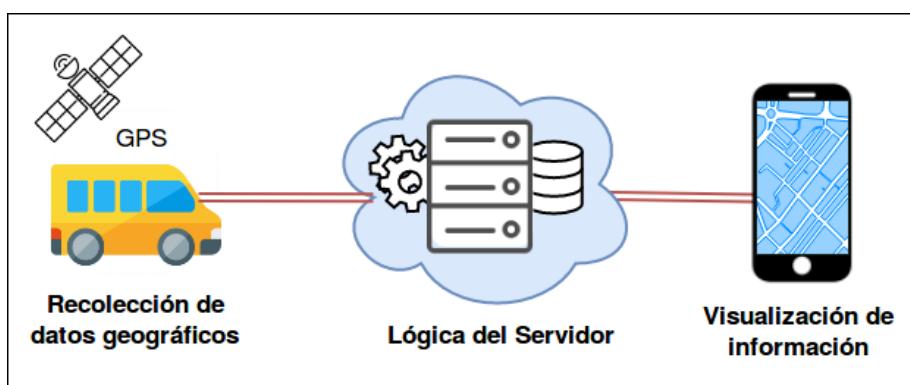


Figura 3.4.- Módulos principales del sistema de monitoreo.

### 3.3.1. Recolección de datos geográficos

El Módulo de Recolección de Datos Geográficos es el primero en la cadena de flujo de información y por tanto el origen de los datos, consiste en una aplicación móvil instalada en un smartphone Android que va montado en el autobús que se desea rastrear. Esta aplicación está dedicada a realizar tres tareas ilustradas en el diagrama de flujo de la imagen 3.5 y explicadas en las líneas siguientes:

La tarea principal que desempeña esta aplicación es notificar al servidor la ubicación en tiempo real de los autobuses que circulan una ruta específica del sistema de transporte de la ciudad o localidad, también es responsable de realizar cálculos basados en la geoposición de la unidad y detectar el caso en que ésta salga de ruta (figura 3.5).

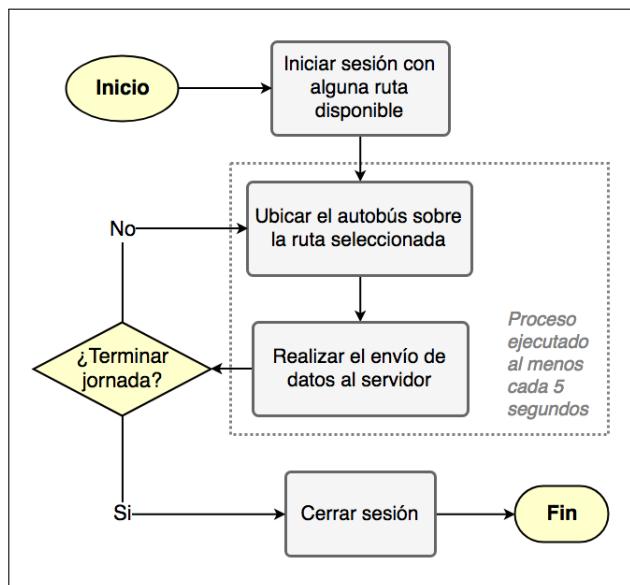


Figura 3.5.- Diagrama de flujo general de los pasos que sigue la aplicación rastreadora.

Por medio del proceso de inicio de sesión se define la ruta que el autobús circulará durante la jornada de trabajo, declarando también la clave de la ruta como encabezado especial de la información enviada al servidor en tiempo real.

Una vez que se estableció la ruta a circular, y para evitar el problema de la variante precisión en las lecturas del sensor GPS de los dispositivos móviles, la aplicación se encarga de ubicar la unidad sobre algún punto o segmento de línea del camino que hay entre las paradas de autobuses implementando cálculos basados en la fórmula conocida como “Haversine”<sup>1</sup> (fórmula 3.1), la cual

<sup>1</sup>Contracción del inglés *Half Versed Sine*.

es utilizada para calcular la distancia entre dos puntos sobre una superficie esférica (en este caso la Tierra) con muy buena aproximación.

$$\begin{aligned} a &= \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \\ c &= 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{(1-a)}) \\ D &= R \cdot c \end{aligned} \quad (3.1)$$

donde:

$R$  = Radio terrestre acordado en 6371 km según [Williams, 2017].

$\varphi$  = Latitud de un punto.

$\lambda$  = Longitud de un punto.

$D$  = Distancia aproximada en metros.

Cuando se obtiene una posición GPS se calcula el segmento de línea o punto más cercano, teniendo así que el autobús se aproxima a esa parte de la ruta (figura 3.6). Un mecanismo similar es empleado para calcular la distancia que le falta al autobús para llegar al siguiente parabús, puesto que la línea es una serie de puntos y su longitud es la suma de las distancias que hay entre ellos (formula 3.2), por lo que la distancia faltante es la diferencia que hay entre la longitud total y la recorrida.

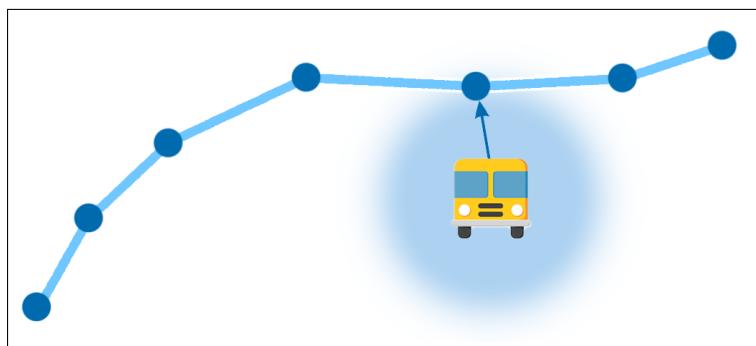


Figura 3.6.- Ubicación de una unidad de transporte sobre un segmento de línea de la ruta.

$$D_T = \sum_{i=0}^{n-1} \text{haversine}((\varphi_i, \lambda_i), (\varphi_{i+1}, \lambda_{i+1})) \quad (3.2)$$

Aparte de la información de interés para el usuario final mencionada en líneas anteriores, la aplicación también calcula la velocidad aproximada a la que se desplaza con base en la distancia

que recorre cada cinco segundos y el tiempo que tardará en llegar a al siguiente parabús. Toda esta información está ligada a una ruta y a una unidad de transporte específicos, y es actualizada al menos cada cinco segundos para enviarla al servidor.

### 3.3.2. Lógica del servidor

Este módulo funge como intermediario entre los datos recolectados en el módulo anterior y la información que se despliega al usuario final (figura 3.4). El servidor realiza tres tareas concretas que se describen a continuación:

#### Interfaz REST

El servidor deja a disposición de los clientes un API de tipo REST para el acceso vía HTTP a la información de las rutas de autobuses y unidades de transporte registrados en el sistema (figura 3.7). La información que los clientes obtienen puede encontrarse en dos formatos diferentes según los siguientes casos:

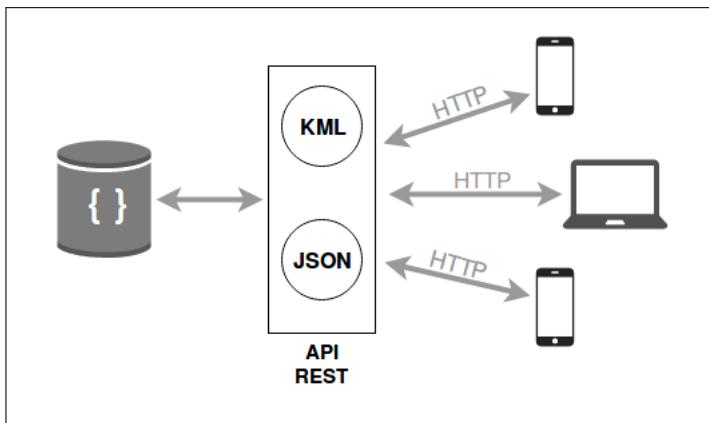


Figura 3.7.- Interfaz REST entre el modelo de datos y los clientes que entrega información en formatos KML y JSON, según sea necesario.

- **Tipo KML:** Al ser un formato estándar para la presentación de información geográfica, se implementa para brindar la información que describe la estructura de las rutas registradas en el sistema, incluyendo la ubicación de los parabuses oficiales o nodos que la delimitan y la dirección de cada camino o arista que los autobuses recorren.

- **Tipo JSON:** La información transferida en este formato es la descripción básica de las instancias que interactúan en el modelo de datos (rutas, nodos, autobuses, ciudades, etc.), ofreciendo reportes como la lista de rutas disponibles en cierta ciudad, la lista de parabuses de dicha ruta, los autobuses que la circulan al momento, entre otros. Mediante este formato también se envían notificaciones del sistema, como el caso en que una ruta cambia y es necesaria su actualización en el resto de los módulos.

## Validación de datos

La tarea de validación de información se implementa con el fin de asegurar un nivel de seguridad en cuanto a la confiabilidad de la información que se distribuye por el sistema. Ésta se realiza en primera instancia al nivel de origen de la información, en el que por medio de un protocolo propio y de la adición de cabeceras, se asegura que esta proviene de una aplicación rastreadora válida. Posteriormente se valida el formato de las cadenas o mensajes recibidos, ya que estos deben contar con una cantidad de caracteres específica y tener una estructura tal que puedan ser interpretados por el servidor. Finalmente algunas credenciales son comparadas con la base de datos para asegurar la legitimidad del mensaje.

## Distribución de la información

Una vez iniciada la transmisión de datos entre las aplicaciones rastreadoras activadas y el servidor, éste procede a la distribución de información filtrada y formateada a los usuarios finales utilizando el protocolo WebSocket, organizando el envío de datos de manera tal que el dispositivo móvil del usuario solo recibe la información de los autobuses activos en la ruta deseada (figura 3.8).

El filtrado se implementa en primera instancia para reducir el tráfico de información en red, evitando así la sobrecarga y congestión en el lado del servidor, y también para evitar que el usuario se vea económicamente afectado a causa del uso de internet móvil cuando no se cuenta con Wi-Fi.

## Manejo de conexiones

Un tema muy importante en cuanto al desarrollo de sistemas distribuidos es el manejo de las conexiones en red. Al estar basado en el protocolo TCP, el protocolo WebSocket también presenta complicaciones para notar cuando una conexión se terminó de manera inesperada y pasó a ser una

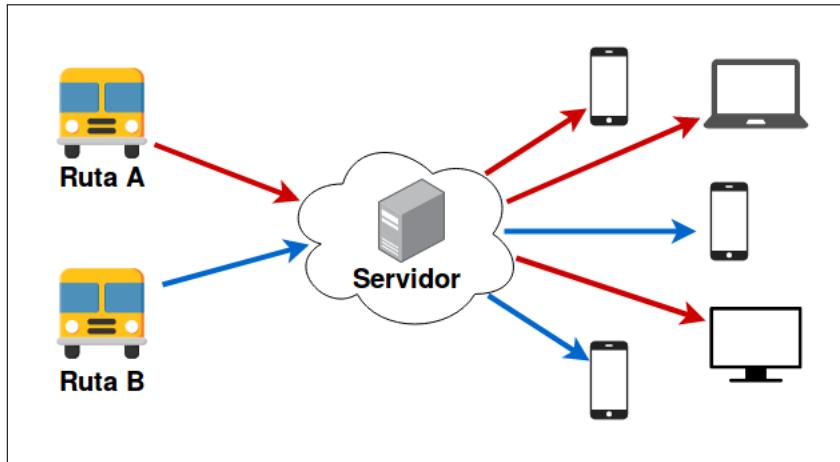


Figura 3.8.- Esquema de la entrega de información bajo demanda por rutas de transporte.

*conexión zombie*, es decir que el canal de comunicación se cerró a causa de un error o excepción y uno de los procesos no recibió el paquete ACK que lo notifica, conservando el apuntador como una conexión activa.

Enfocando dicha situación, el cliente web es afectado en menor grado que los dispositivos que cargan la aplicación móvil, puesto que el cliente web (que es usualmente una computadora con un conveniente tamaño de pantalla) puede perder conexión de manera inesperada en muy esporádicos casos, como la caída de la red Wi-Fi. Los clientes móviles, por el contrario, son más susceptibles a generar conexiones zombies en el servidor a causa de las conexiones dinámicas a la antena celular más cercana o a la pérdida de recepción en zonas de difícil acceso.

En un esquema de conexión en el que un proceso envía información (escribe en el canal) y otro la recibe (lee del canal), el proceso que lee arroja al instante una excepción al perder la comunicación, dando pauta a reaccionar de acuerdo al protocolo establecido en el desarrollo. El proceso que escribe no puede reaccionar de la misma manera debido a que en el sistema no hay nada que le avise que se perdió la conexión, por lo que una de las medidas recomendadas en el lado del envío de datos es la limitación del tamaño del *buffer* temporal, asegurando que al llenarse arroje una excepción que notifique al proceso de escritura que el canal de comunicación es inválido.

Java maneja un buffer de 8 kb para cada conexión WebSocket, sin embargo en este caso se considera que éste debe ser reducido a 3 kb para para disminuir la probabilidad de saturar el procesador con conexiones zombies, y así lograr que el sistema reaccione oportunamente.

### 3.3.3. Visualización de la información

Finalmente, el módulo para el despliegue o visualización de la información comprende una aplicación para dispositivos móviles Android y una aplicación web, que funcionan como la interfaz que presenta gráficamente la información proveniente del servidor e interactúa con el usuario final. Ambas aplicaciones satisfacen el mismo objetivo primario: Mostrar al usuario la ubicación de los autobuses que circulan una ruta de interés (figura 3.9).

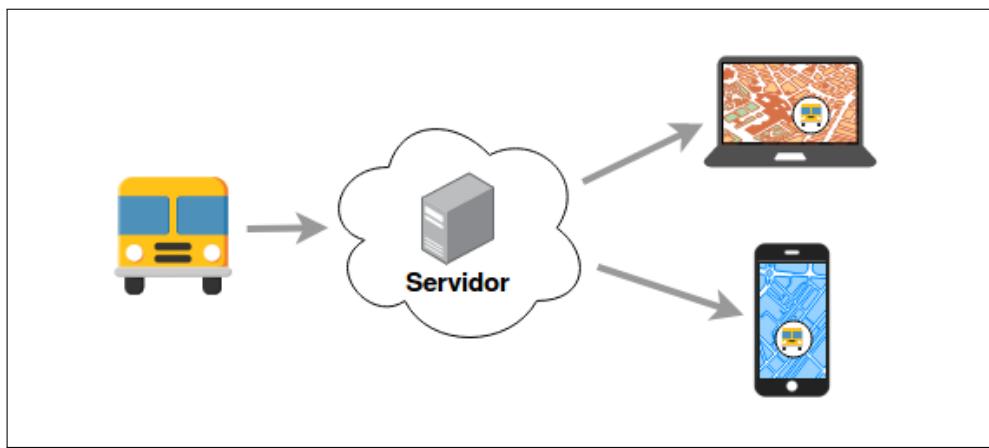


Figura 3.9.- Despliegue de información geográfica en tiempo real sobre la web y móviles.

Ambas aplicaciones visuales obtienen la información en tiempo real de los autobuses al conectarse al módulo de lógica de servidores utilizando el protocolo WebSocket y enviando una serie de cabeceras que lo acreditan como un dispositivo cliente confiable. Entre dichas cabeceras se encuentra un identificador que le indica al servidor la ruta que el cliente pide en pantalla y, por consiguiente, los datos de los autobuses que son necesarios, a medida que los datos son obtenidos van siendo graficados (figura 3.10).

La función secundaria del módulo en cuestión, es la de notificar al usuario o interactor cuando un autobús se acerca a algún punto de interés sobre la ruta (dígase un parabús de su preferencia). Paralelamente al despliegue de los autobuses en pantalla, el software está habilitado para realizar cálculos basados en la información que el servidor envía y resolver los autobuses que se encuentran más cerca de el parabús seleccionado y si alguno se encuentra lo suficientemente cerca (referencia acordada a 80m o menos) como para activar la notificación al usuario.

Cabe mencionar que, a pesar de que las funciones de estas aplicaciones son las mismas, algunas

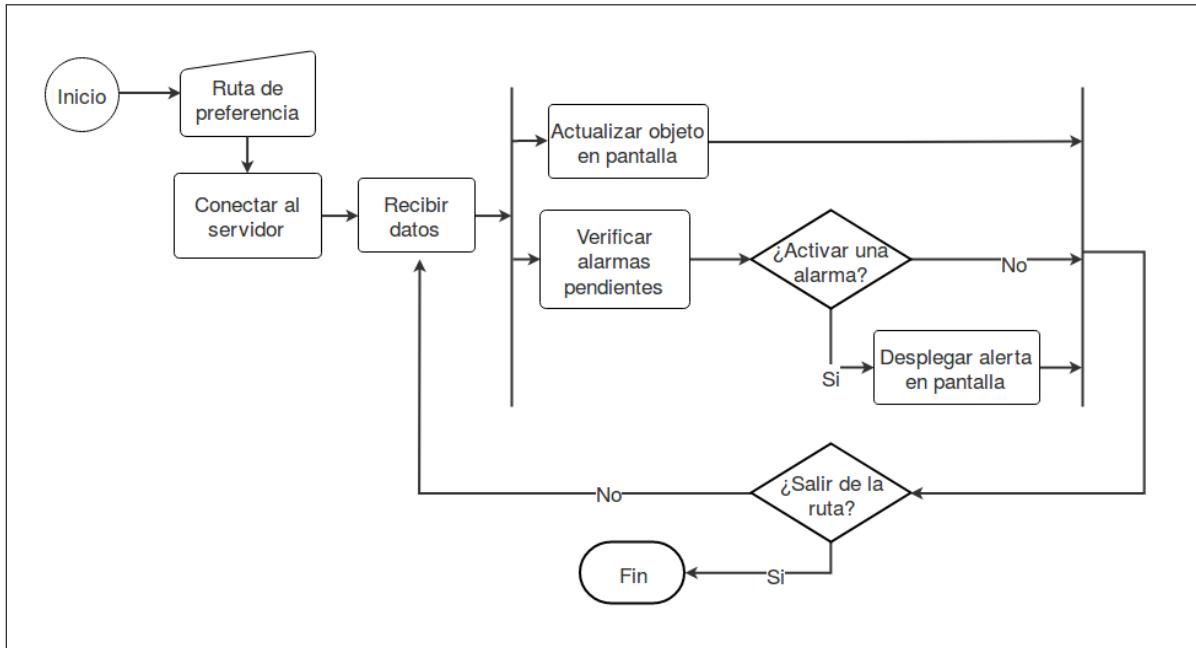


Figura 3.10.- Diagrama del funcionamiento general de las aplicaciones que interactúan con el usuario final del sistema.

de éstas no fueron implementadas de la misma manera debido al uso de diferentes lenguajes de programación y tecnologías. Esto se describe más a detalle en el siguiente capítulo.

### 3.4. Comentarios finales

A pesar de ser descrita de manera introductoria es posible expresar que el sistema fue diseñado desde su base para ser fácilmente actualizable y escalable, por lo que no se utilizaron Frameworks de terceros. Iniciando por el uso de un esquema de datos más orgánico y natural que el modelo tabular clásico, el cual captura muy acertadamente la situación real de un sistema de transporte público. Continuando por el uso de servicios web para dar acceso a la información de manera independiente por parte de los procesos externos, y permitiendo aumentar funcionalidades o mejorar las ya existentes sin interferir con las que se encuentran activas. Finalmente en cuanto al flujo de datos propuesto, se pone gran atención al rendimiento del servidor, ya que se toman medidas importantes para evitar, por ejemplo, problemas de sobrecarga por procesamiento de peticiones entrantes y conexiones persistentes.

El siguiente capítulo detalla la procedencia y objetivo de dichas conexiones y peticiones, la

información que se transfiere y su procesamiento hasta la pantalla del usuario final.

## **Capítulo 4**

# **Implementación del Módulo de Visualización (Móvil y Web)**

A continuación se explican detalladamente aspectos clave para el desarrollo de la aplicación móvil que sirve al usuario final como visor de información de interés, algunos de estos son el modelo lógico seguido para su interacción con el usuario, el esquema de conexión e intercambio de información con módulo de Lógica del Servidor y su arquitectura interna. Cabe mencionar que la arquitectura presentada contempla la restricción impuesta por el Sistema Operativo Android en la que se designa al hilo o proceso principal como el único capaz de alterar la interfaz gráfica [Android, 2017].

Posterior a esto se habla acerca de la implementación de la interfaz para la web, la cuál se construye bajo los mismos lineamientos de funcionalidad e interacción

### **4.1. Interacción con el usuario final**

Como se mencionó anteriormente, el objetivo del módulo de visualización de datos es proveer al usuario final con información de interés, cubriendo los siguientes requerimientos:

- El usuario podrá conocer si el sistema de monitoreo está activo en su ciudad o localidad.
- Podrá saber cuales son las rutas habilitadas para su ciudad, en caso de que ésta esté dada de alta en el sistema.

- Podrá acceder a información acerca de las rutas activas y sobre los autobuses o unidades de transporte que las circulan.
- Podrá activar una alerta que le indicará cuando una unidad de transporte esté por pasar un determinado parabús.

Para ello se diseñó un esquema de interacción (figura 4.1) compuesto por solamente tres escenas principales que promueven la facilidad de uso y comprensión por parte del usuario.

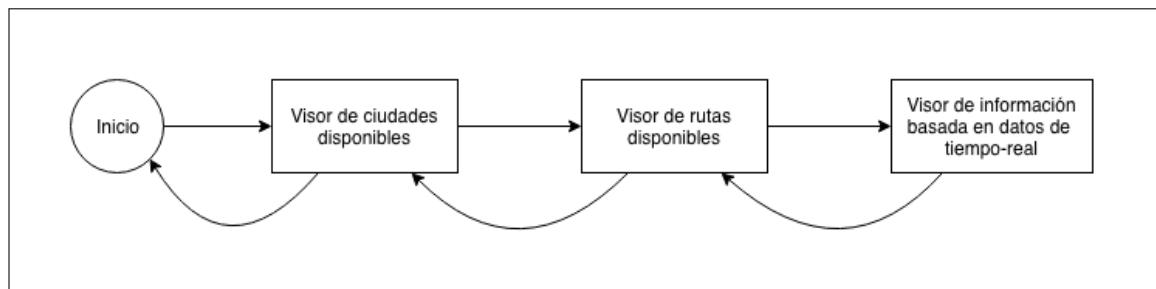


Figura 4.1.- Esquema de interacción entre la aplicación móvil y el usuario final.

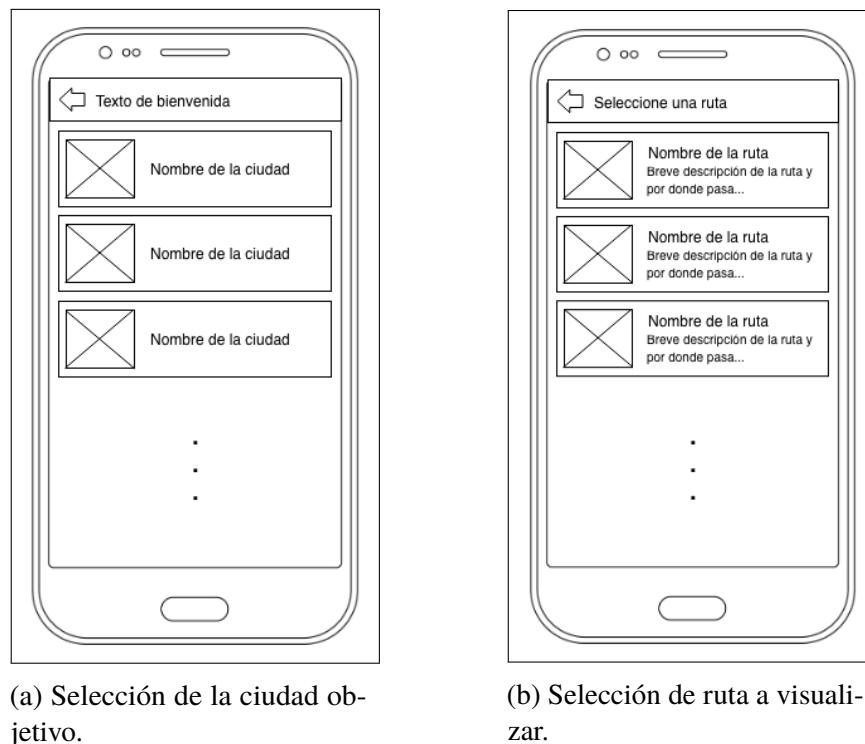


Figura 4.2.- Esquemas *Wireframe* móviles que plasman el contenido y estructura de las primeras escenas para el usuario.

Derivado del esquema de interacción, se tiene que el usuario debe establecer la ruta que quiere ver antes de iniciar con la transmisión de datos en tiempo-real, cuyas vistas se muestran en la figura 4.2.

La pantalla de inicio de la aplicación incluye un listado de las ciudades (figura 4.2a) que se encuentran registradas en la plataforma, al seleccionar una ciudad se transita a la vista en la que se muestran las rutas urbanas disponibles en la ciudad seleccionada (figura 4.2b).

Una vez seleccionada la ruta a visualizar, el dispositivo móvil despliega en pantalla la última escena de interacción (figuras 4.3). Esta escena muestra la estructura de la ruta seleccionada (los parabuses que incluye y los caminos que los conectan) y la información dinámica que proviene de los autobuses o unidades en circulación (figura 4.3a).

Otra de los objetivos de esta escena es mostrar información bajo demanda al usuario respecto a cada uno de los parabuses que conforman la ruta (figura 4.3b), tal como su nombre y la distancia y tiempo de llegada estimados de las tres unidades de transporte que se encuentran más cercanas, tomando como cercanía la distancia que les falta por recorrer siguiendo el sentido de la ruta.

Finalmente también habilita al usuario para activar alertas en los parabuses que él deseé con la finalidad de ser notificado cuando una unidad de transporte esté próxima a pasar por ellos (figura 4.3c).

## 4.2. Obtención de los datos

Para lograr mostrar información de interés en la pantalla del dispositivo móvil, la lógica que rige la aplicación móvil necesita obtener algunos datos provenientes del módulo de Lógica del Servidor. Esto se logra a través del uso de dos canales de comunicación: Peticiones HTTP a servicios web y una conexión persistente con el protocolo WebSocket.

Las peticiones a servicios web se emplean para obtener aquellos bloques de datos que no necesitan ser actualizados constantemente, tal es el caso de la lista de ciudades que se encuentran registradas en la base de datos, el listado de rutas de transporte urbano por ciudad, el conjunto de autobuses registrados y la ruta que circulan, y la estructura de una ruta específica.

Por otra parte, la conexión persistente WebSocket sirve como canal para aquella información que surge en tiempo-real, como el caso de una actualización en la información de las rutas y ciudades o,

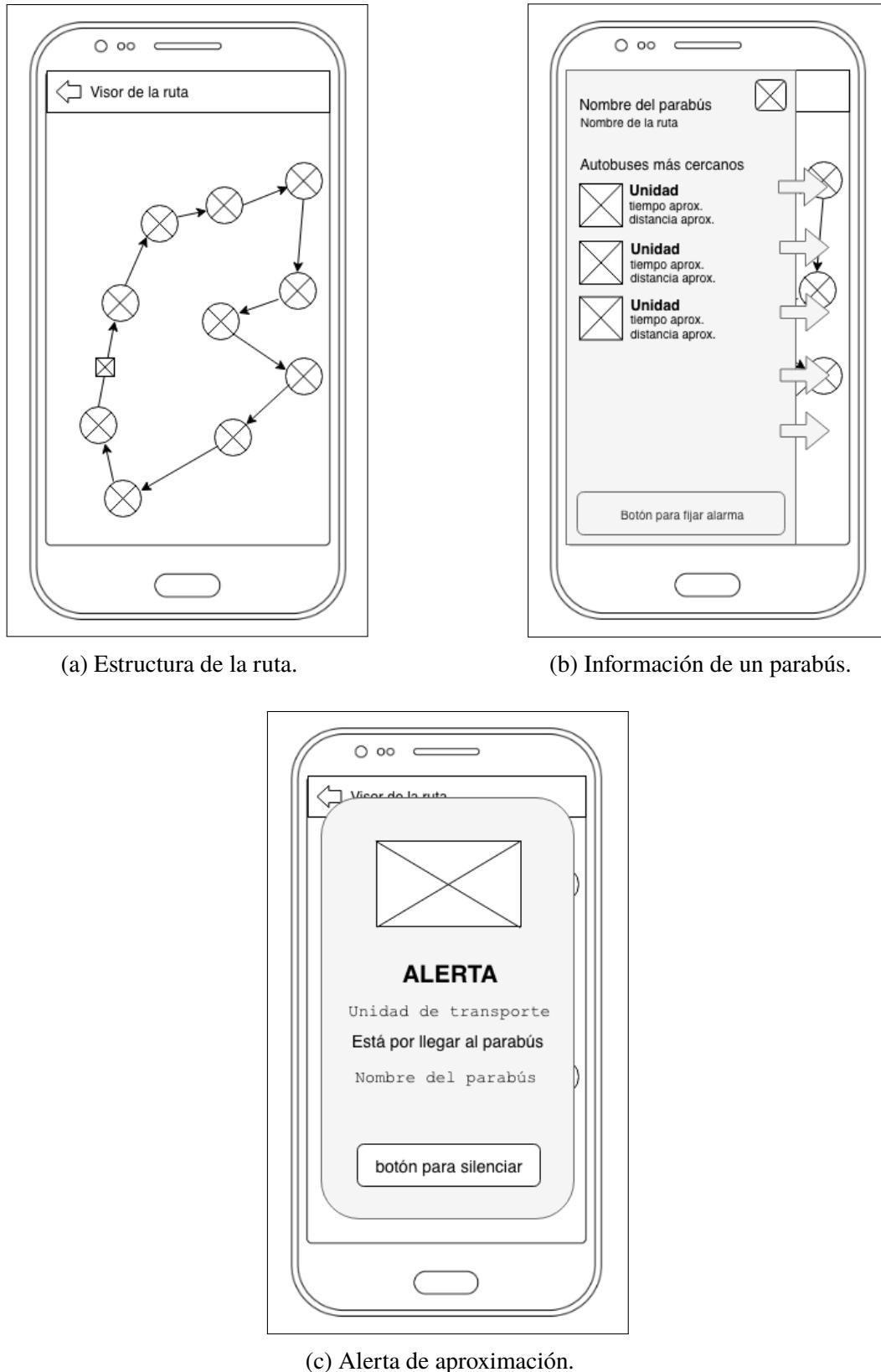


Figura 4.3.- Esquemas Wireframe móviles que representan el contenido de la última escena de interacción con el usuario.

por supuesto, las actualizaciones de posición GPS de los autobuses que circulan una ruta determinada.

Una de las opciones que se revisaron para lograr la funcionalidad de tiempo-real fue el esquema AJAX, sin embargo éste se descartó debido a la sobrecarga de peticiones HTTP que presenta y a que es más lento para transmitir datos en comparación con el protocolo WebSocket, puesto que este último establece un solo canal bidireccional en lugar de abrir muchas conexiones consecutivas.

El protocolo WebSocket también trae consigo una desventaja al ser utilizado bajo un esquema de conexión intermitente, caso de los dispositivos móviles. Un dispositivo móvil que utiliza internet emplea constantemente un algoritmo para estar siempre conectado a la antena celular más cercana con base en el cálculo de triangulaciones, por lo que el dispositivo realiza el cambio de conexión si sale del rango de una u otra ofrece mayor calidad de señal. Cuando esto sucede, el protocolo WebSocket indica que el dispositivo debe terminar la conexión establecida e intentar enlazarse alEndPoint (servidor) con una conexión nueva.

La situación anterior genera una conexión zombie en el servidor que, en combinación con la espera programada que el protocolo TCP tiene, utiliza un espacio y recursos por un tiempo indefinido y usualmente prolongado. Una forma de reducir el problema fue la reducción del buffer de escritura para cada conexión WebSocket, así el paso continuo de paquetes lo satura y la excepción arrojada deja en claro que conexión está caída y se procede a la liberación de recursos.

### 4.3. Arquitectura de la aplicación móvil

A pesar de que por el momento los controles ofrecidos al usuario en la aplicación móvil no son muy complejos, internamente se siguió una arquitectura de desarrollo basada en un modelo monolítica de datos y su interacción con otros tres módulos de control y procesamiento de datos, mostrada en la figura 4.4.

El sistema operativo ya cuenta con una restricción en cuanto al uso del proceso principal, la cual indica dos puntos muy importantes que evitan que la interfaz gráfica se vea lenta o congelada en algún momento, estos son:

- El proceso o hilo principal no puede ejecutar tareas relacionadas a operaciones I/O<sup>1</sup> locales o en red.

---

<sup>1</sup>Expresión utilizada en inglés (*Input/Output*) para señalar operaciones externas de entrada y salida de datos.

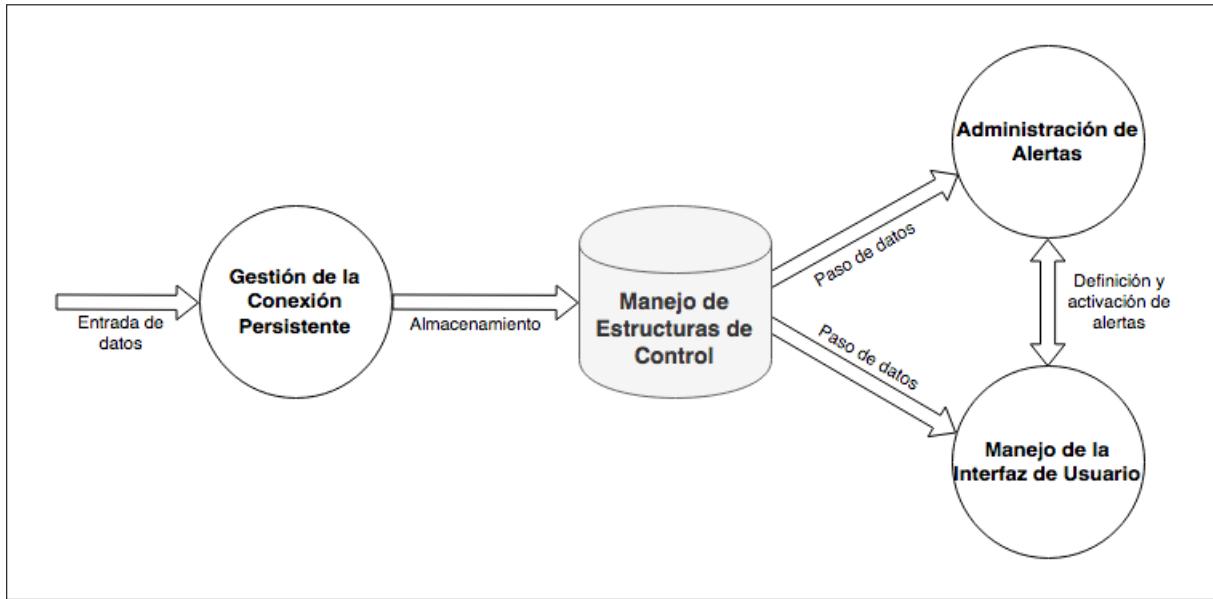


Figura 4.4.- Arquitectura de la aplicación móvil.

- Tampoco es recomendable que ejecute procesos que involucren muchos cálculos o llamados a funciones en pilas grandes.

Por ello, el módulo Administrador de Alertas y el Manejador de la Interfaz de Usuario son separados en procesos secundarios que se ejecutan en segundo plano. Estos subprocessos fueron implementados utilizando la biblioteca *Services* definida en el núcleo de Android, ya que esto aporta la valiosa característica de poder continuar en ejecución aunque el usuario cambie de aplicación.

De acuerdo al orden de activación de procesos (figura 4.5), cuando el usuario entra a la vista en la que se despliega la información de los autobuses el proceso principal se encarga de activar los servicios que se ejecutan en segundo plano. Al concluir el uso de esta vista, el sistema recibe la señal de destrucción de la escena o actividad y ésta ordena la destrucción de los servicios ligados.

### Gestor de conexión persistente

El Servicio Gestor de la Conexión es el encargado de iniciar y mantener un enlace persistente entre el dispositivo móvil y la lógica del servidor utilizando el protocolo WebSocket, actuando en caso de errores en la conexión causados principalmente por la pérdida de conexión móvil. Su tarea principal es recibir y verificar los paquetes de datos recibidos para almacenarlos posteriormente.

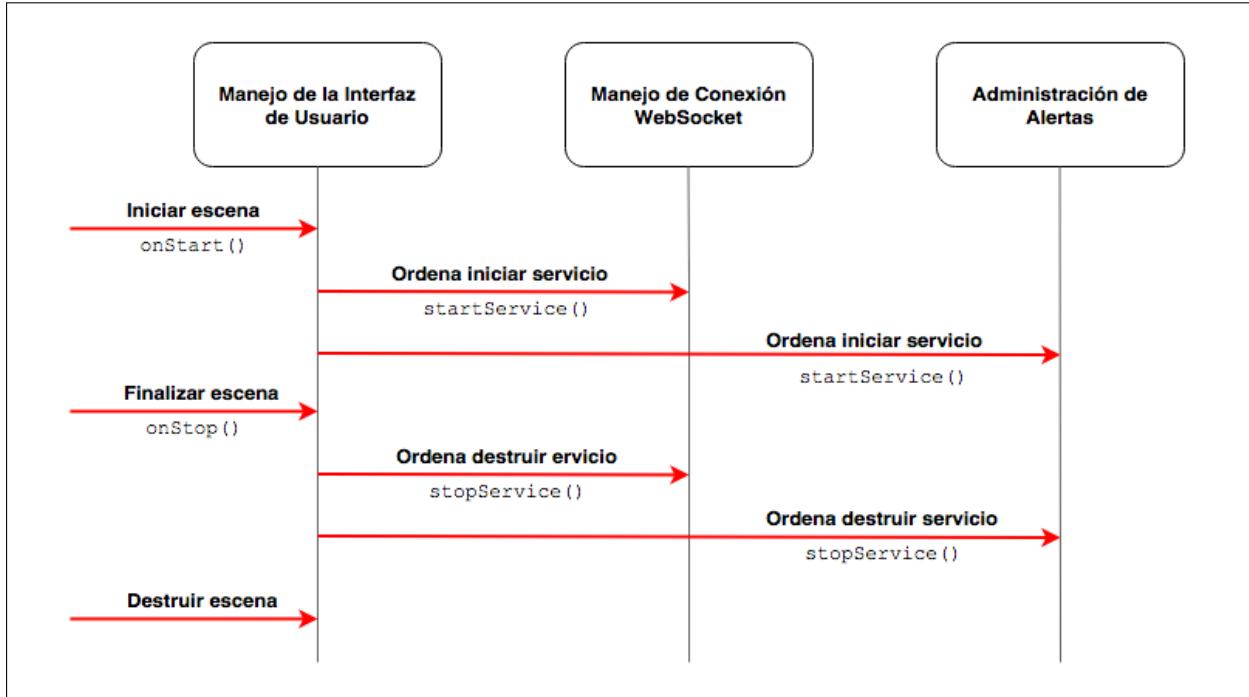


Figura 4.5.- Esquema que plasma el proceso de activación de procesos de acuerdo a los eventos disparados por el ciclo de vida definido por Android.

### Administrador de estructuras de control

Son las estructuras tabulares centrales que se encargan de almacenar la información estructural de la ruta y los datos provenientes de la conexión WebSocket. También se encargan de notificar actualizaciones a los procesos de Administración de Alarmas y al proceso de Manejo de la Interfaz para que calculen cercanías y actualicen los componentes gráficos, respectivamente. Es importante destacar que estas instancias manejan una referencia estática para simular un repositorio de datos que puede ser consultado independientemente del proceso.

### Administrador de alertas

Es el proceso en segundo plano que se encarga, en primera instancia, de registrar y eliminar alarmas según la interacción del usuario con la interfaz gráfica. Conforme se registran actualizaciones en las tablas de datos centrales, el proceso calcula si hay una cercanía esperada por el usuario entre alguna alerta y algún autobús, en caso de haberla envía una señal al proceso principal para que se lo muestre en pantalla al usuario.

Para decidir si un Autobús se encuentra próximo a un parabús con alerta, el algoritmo 1 plantea

el flujo de operaciones seguidas para obtener una distancia entre el autobús y el parabús al que se le asignó la alerta, si está los suficientes metros cerca, ésta se activará. Es de mencionar que para calcular dicha distancia existen tres casos (figura 4.6):

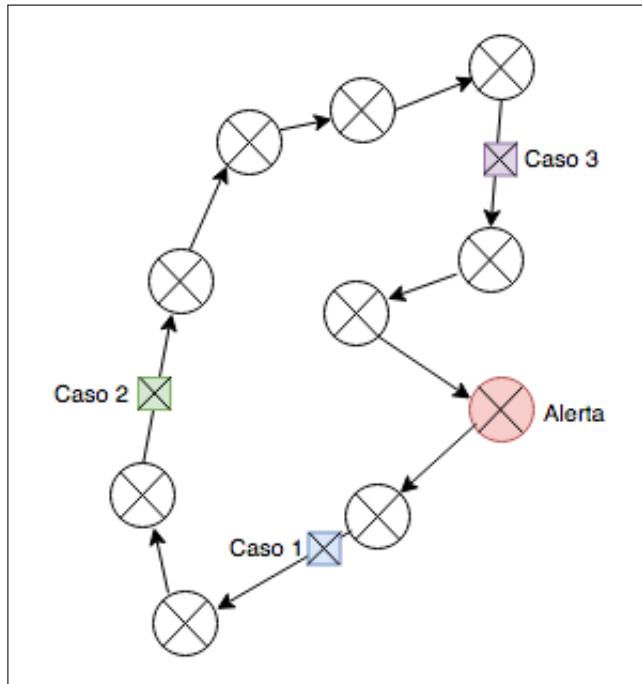


Figura 4.6.- Casos a tomar en cuenta al calcular la distancia desde la posición de un autobús a un parabús con alerta.

- El autobús está en el mismo sentido que el parabús de alerta, pero se encuentra después en el orden (caso 1 de la figura 4.6) por lo que para llegar debe recorrer el circuito casi por completo.
- El autobús se encuentra en algún punto del sentido contrario al parabús (caso 2 de la figura 4.6), debe completar el sentido actual, y luego recorrer el contrario hasta llegar al parabús.
- El autobús se encuentra en el mismo sentido que el parabús y antes de éste (caso 3 de la figura 4.6), solo debe llegar al parabús de alerta.

### Manejador de la interfaz de usuario

Este proceso es ejecutado por el hilo principal de la aplicación. Se encarga primeramente de mostrar en pantalla los componentes gráficos que describen la estructura de la ruta de autobuses, es decir, los parabuses y las conexiones entre estos que la describen. También es el encargado de mostrar

---

**Algoritmo 1** Con cada actualización de la tabla observada se ejecuta el cálculo de proximidad

---

```

1: procedure NOTIFICARACTUALIZACION(autobus)           ▷ Notifica en caso de proximidad.
2:   alertas                                         ▷ Una tabla con las alertas programadas.
3:   for alerta : alertas do
4:     distacia  $\leftarrow$  calcDistanciaTotal (alerta, autobus)
5:     if distancia  $\leq$  80 then
6:       enviarAlerta (alerta, autobus)                ▷ Envía al proceso principal.
7:       return
8:     end if
9:   end for
10:  end procedure
11: procedure CALC DISTANCIA TOTAL(alerta, autobus)      ▷ Calcula proximidad en ruta.
12:   distacia  $\leftarrow$  0
13:   alertS  $\leftarrow$  alerta.sentido
14:   alertI  $\leftarrow$  alerta.indice
15:   busS  $\leftarrow$  autobus.sentido
16:   busI  $\leftarrow$  autobus.indice
17:   if busS = alertS  $\wedge$  busI  $\geq$  alertI then
18:     distacia += calcDistanciaSentido(busS, busI, null)
19:     busI  $\leftarrow$  0
20:   end if
21:   if busS  $\neq$  alertS then
22:     distacia += calcDistanciaSentido(alertS, busI, null)
23:     busI  $\leftarrow$  0
24:   end if
25:   distacia += calcDistanciaSentido(busS, busI, alertI)
26:   return distacia
27: end procedure
28: procedure CALC DISTANCIA SENTIDO(sentido, inicio, fin)    ▷ Calcula longitud de segmento.
29:   if fin == null then
30:     fin  $\leftarrow$  0
31:   end if
32:   d  $\leftarrow$  0                                              ▷ Distancia calculada por sentido.
33:   segmentos = sentidos[sentido]
34:   for i  $\leftarrow$  inicio, fin do
35:     d +=  $\sum_{i=0}^{n-1}$  haversine(( $\varphi_i, \lambda_i$ ), ( $\varphi_{i+1}, \lambda_{i+1}$ ))
36:   end for
37:   return d
38: end procedure

```

---

componentes gráficos dinámicos que representen a los autobuses en circulación, cambiando su ubicación y orientación de acuerdo a las actualizaciones registradas en las tablas de datos. Finalmente se encarga de mostrar alertas en pantalla respecto a notificaciones de conexión o activación de alertas de proximidad.

#### 4.3.1. Flujo de datos basado en eventos

Con el objetivo de lograr un buen desempeño de la aplicación móvil en tiempo de ejecución, la comunicación entre la administración de las tablas de datos y los procesos de Administración de Alertas y Manejo de la Interfaz de Usuario se implementó bajo del patrón de diseño Observer-Observable (figura 4.7), teniendo al primero como una entidad independiente y a los otros como dependientes.

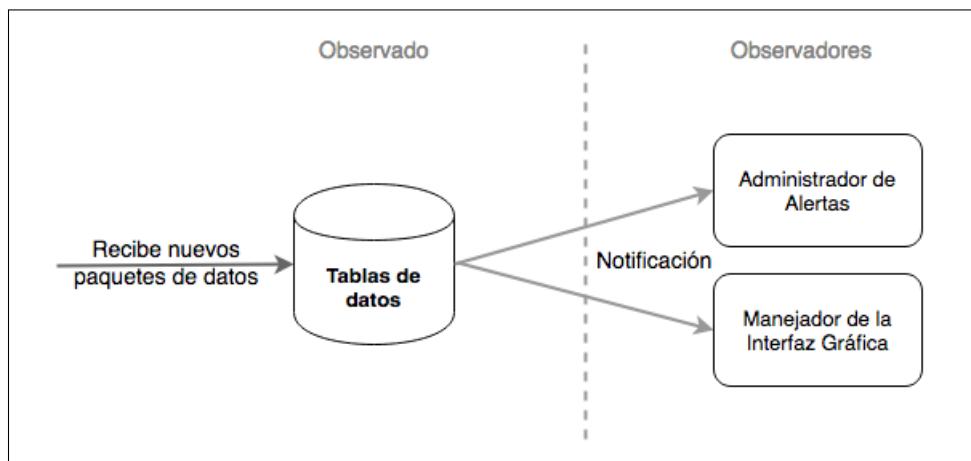


Figura 4.7.- Comunicación entre procesos basada en el patrón de diseño Observer-Observable.

Al diseñarlo así, cada que llega un nuevo paquete de datos y es almacenado en las tablas se dispara un evento que notifica una actualización a los procesos dependientes para que estos realicen las acciones pertinentes.

#### 4.4. Implementación en el ambiente web

El desarrollo de una versión web de esta aplicación surgió con la idea de abarcar el mayor número de usuarios posibles, lo cual incluye aquellos casos en los que por cualquier razón el usuario no cuenta

con un dispositivo móvil, pero es un usuario frecuente del transporte urbano y, por ende, el sistema de rastreo le sería de gran utilidad, y los casos en los que el usuario no puede instalar el aplicativo por no contar con sistema operativo Android o por falta de espacio en memoria.

La versión web está basada en las tecnologías estándar HTML5, CSS3 y JavaScript con la biblioteca JQuery, y utiliza técnicamente el mismo mecanismo de funcionamiento que la versión móvil, sin embargo el diagrama de interacción de ésta con el usuario es más sencillo porque toda la pre-configuración se realiza en la misma página o escena (figura 4.8), pero implementando transiciones para llevar orden en la pre-configuración.

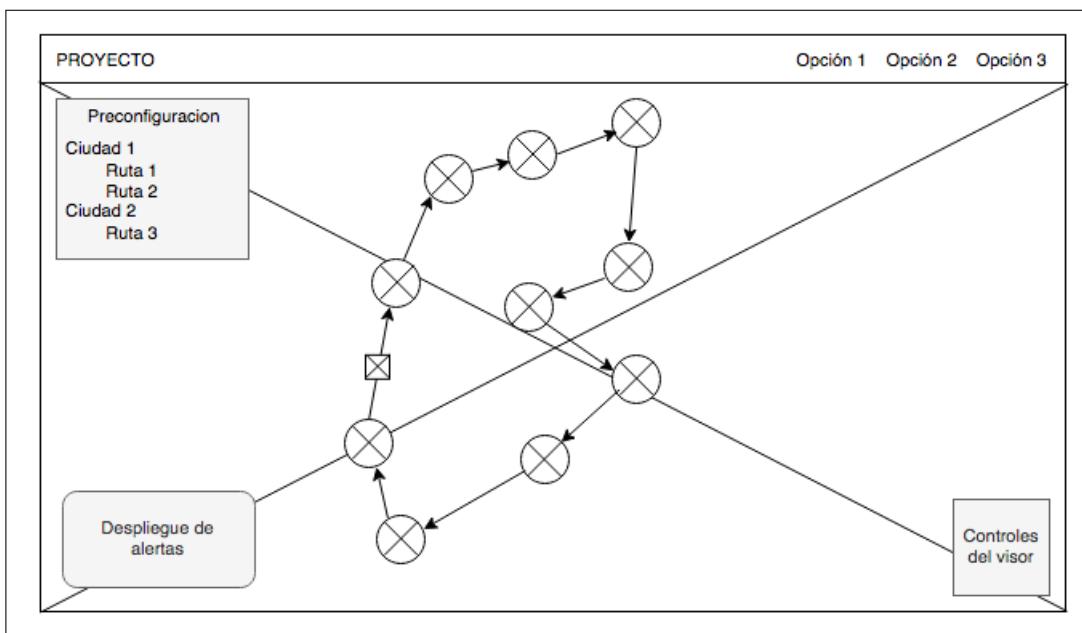


Figura 4.8.- Modelo wireframe básico de la interfaz web equivalente a la aplicación móvil.

En cuanto al diseño arquitectónico de esta versión web, el manejo de los componentes o interactores en pantalla está definido por una combinación entre el modelo DOM y el patrón de diseño Observer-Observable, la cual se explica a continuación:

El modelo DOM maneja dos árboles lógicos, uno de éstos es la representación estructural (HTML) de la interfaz web, en la que el usuario final interactúa con menús, botones y barras de desplazamiento para ordenar cambios y obtener información. El otro árbol que DOM maneja es el que representa la estructura KML de la ruta de autobuses solicitada por el usuario (figura 4.9), teniendo nodos para los parabuses y ligas para los caminos.

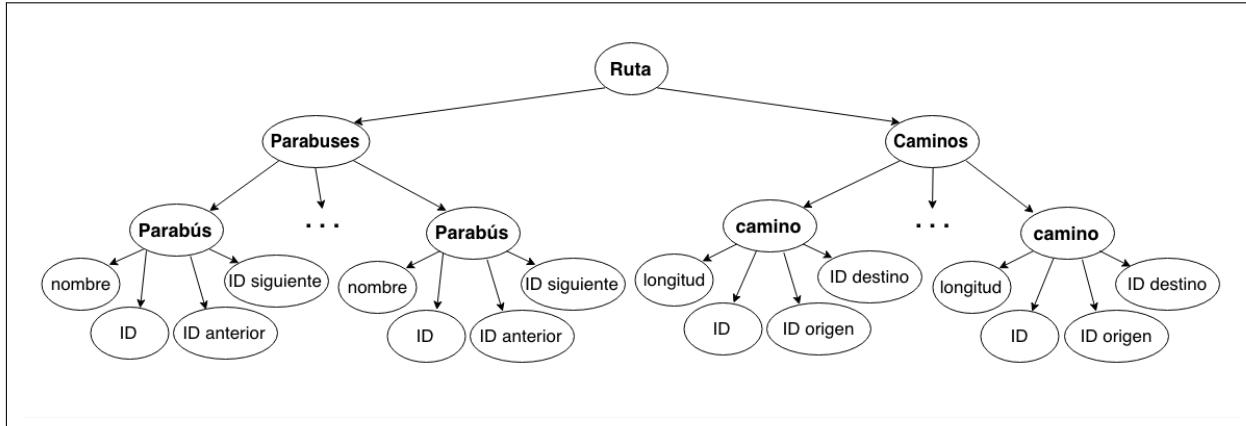


Figura 4.9.- Árbol generado por el modelo DOM al procesar el archivo descriptor KML.

El patrón Observer es el que funge como código de enlace o *pegamento* entre ambos árboles lógicos, definiendo eventos en sus nodos que al dispararse ordenan cambios en la información que se le pide al servidor y que se despliega al cliente. Tal es el caso del evento disparado por un click sobre un nodo parabús, que ordena un cambio en el componente gráfico de la barra de desplazamiento.

Una de las ventajas que la versión web presenta es la baja probabilidad de pérdida de conexión, ya que un equipo de cómputo, tanto de escritorio como portátil, está usualmente conectado a una red Wi-Fi fija y no suele salir del alcance de ésta, por lo que un cambio de red no es tan frecuente con en el caso de los dispositivos con internet móvil. Esta característica trae consigo que los clientes web no generan el mismo número de conexiones zombie que los clientes móviles.

También, debido a que el estándar WebSocket fue originalmente desarrollado en conjunto con la versión 5 del lenguaje HTML, la biblioteca WebSocket presenta mayor estabilidad en el entorno web en comparación con su adaptación a su entorno móvil. Aunado a esto, JavaScript facilita mucho el desarrollo debido a que el uso de procesos paralelos es casi automático, siendo innecesario tener que especificar diferentes hilos y mecanismos de comunicación entre ellos.

## 4.5. Implementación del módulo de visualización (móvil y web)

Esta sección muestra el resultado de la implementación para ambas plataformas: Android y Web. El despliegue de imágenes está ordenado de acuerdo al modelo de interacción con el usuario, y se muestran capturas equivalentes de ambas plataformas.

Es importante resaltar que en el aspecto gráfico de ambas aplicaciones se implementó el patrón de diseño gráfico *Material Design* propuesto por Google. También hay que mencionar que para implementar una base dinámica que funciones como un mapa de fondo de pantalla se utilizó la biblioteca y el API RESTful de Google Maps, aunque debido a las políticas de uso se propone una segunda iteración utilizando la biblioteca y el API de OpenStreetMaps, lo cual permitiría implementar un modelo de negocio sin pasar por alto ninguna licencia.

### Selección de ciudad/población

Las imágenes en la figura 4.10 muestra la primera interacción entre el usuario y el software, es decir la selección de la ciudad de interés por parte del usuario.

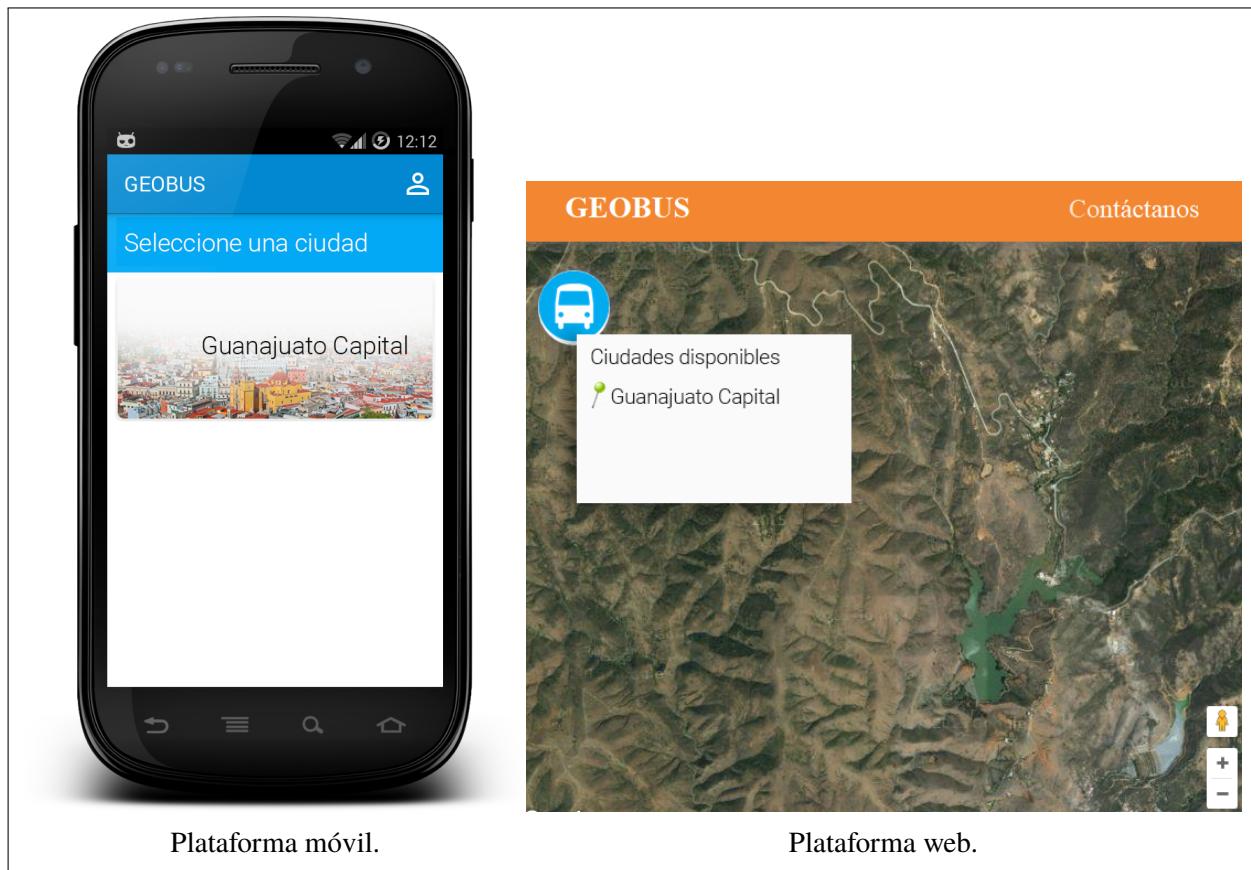


Figura 4.10.- Vistas de la selección de la ciudad de interés.

## Selección de ruta deseada

La figura 4.11 muestra la escena en la que el usuario selecciona la ruta que quiere visualizar en pantalla y de la que quiere recibir información en tiempo-real.

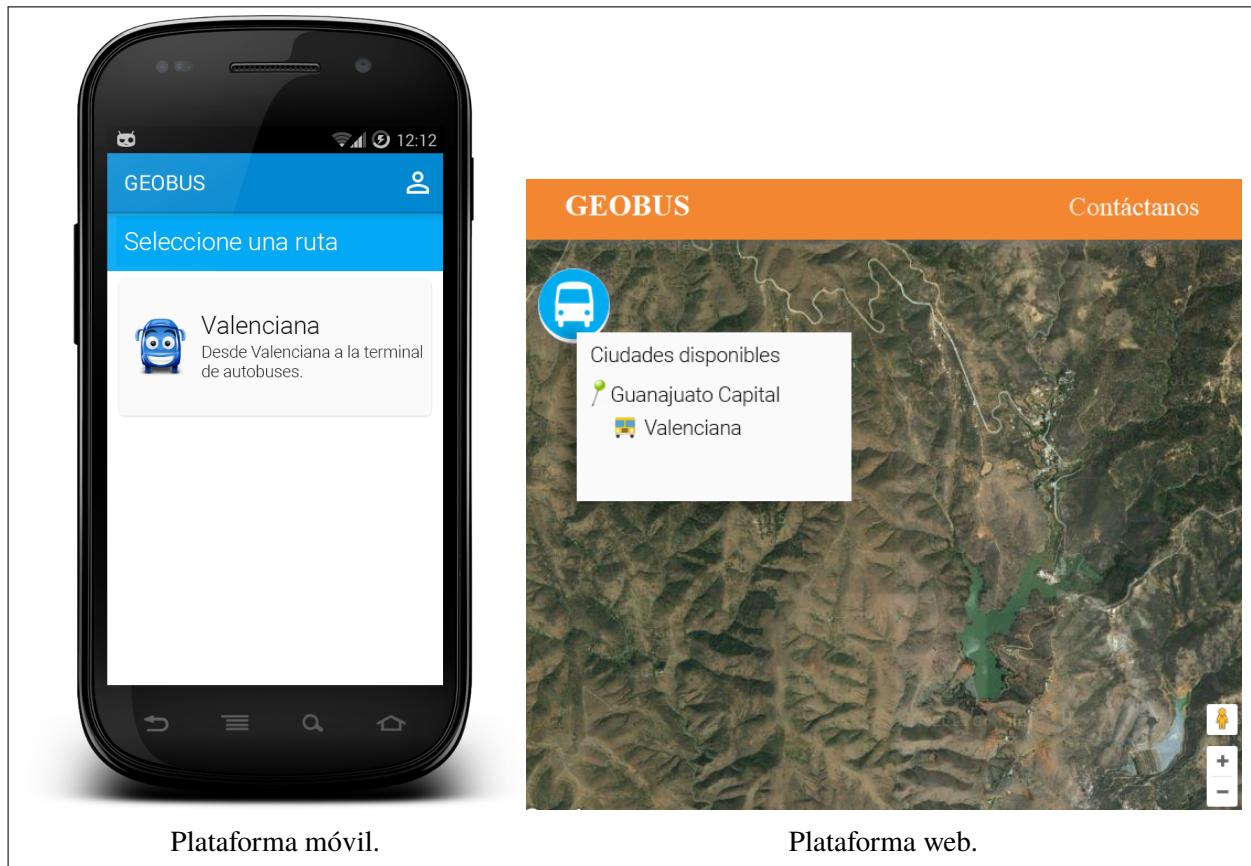


Figura 4.11.- Vistas de la selección de la ruta a visualizar.

## Despliegue de la ruta

La figura 4.12 muestra la escena en la que se construye la ruta con componentes gráficos.

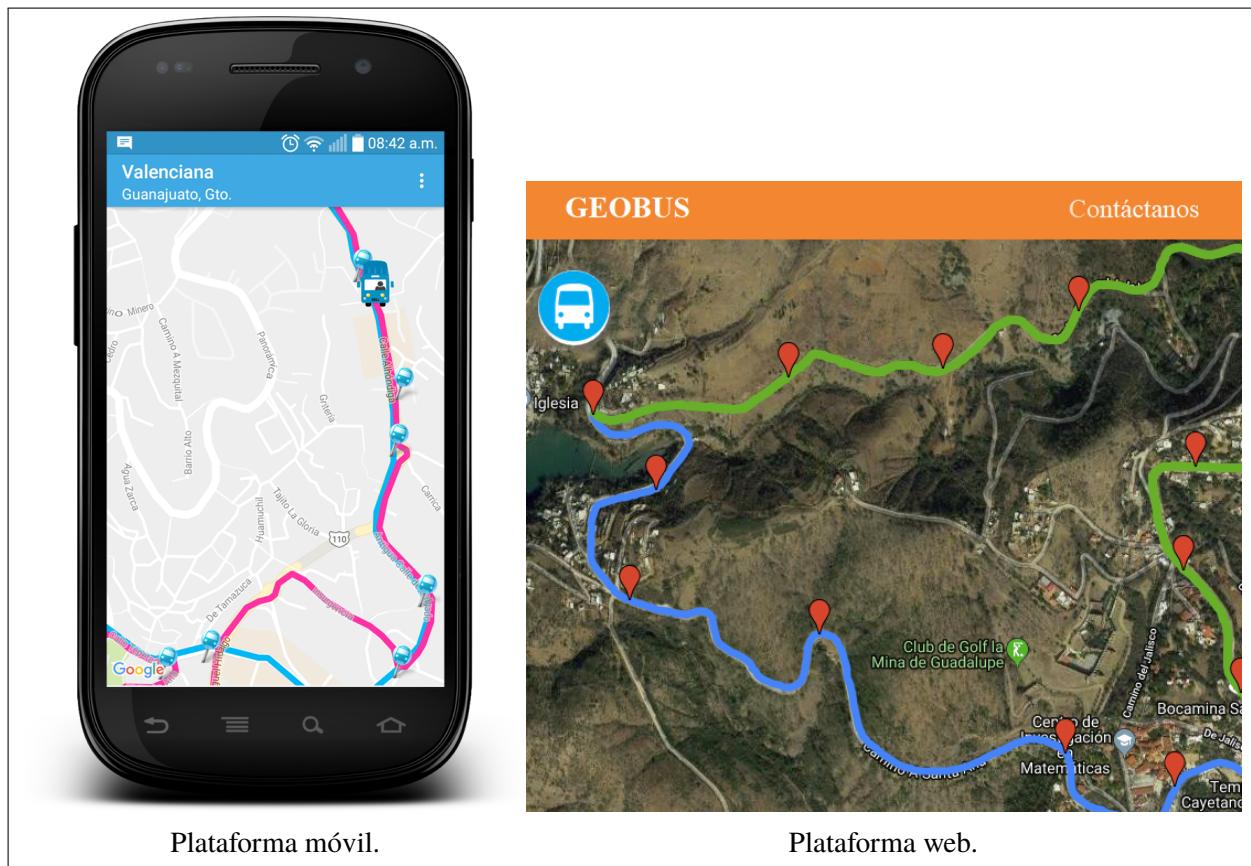


Figura 4.12.- Vistas que muestra la construcción de una ruta sobre un mapa dinámico basado en GoogleMaps.

## Información y opciones de un parabús

La figura 4.13 muestra la forma en la que se despliega la información relacionada a un parabús seleccionado y las opciones pertinentes.

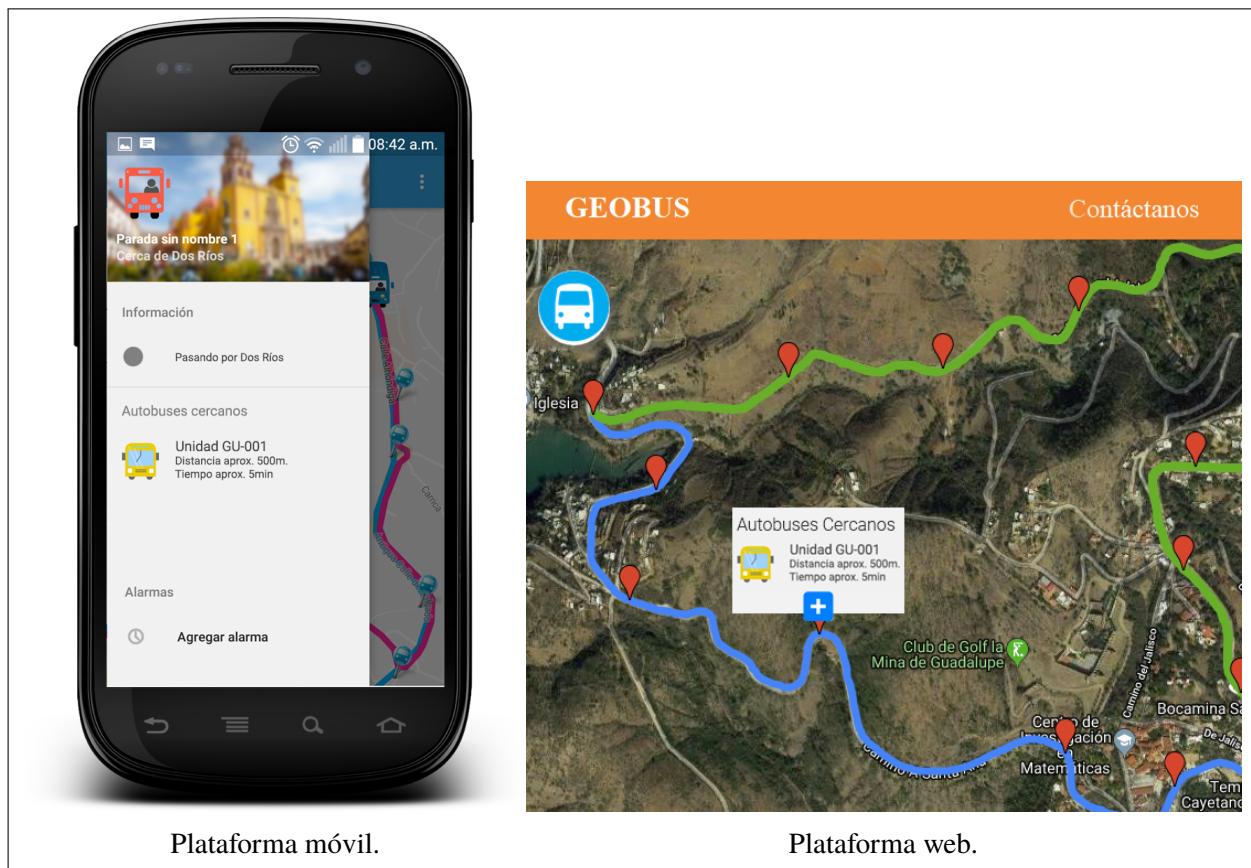


Figura 4.13.- Pantallas en las que se plasma la manera en que se muestra la información y opciones de un parabús seleccionado.

## Despliegue de la alerta

La figura 4.14 muestran el despliegue de una alerta activada en un parabús al momento en que se activa.

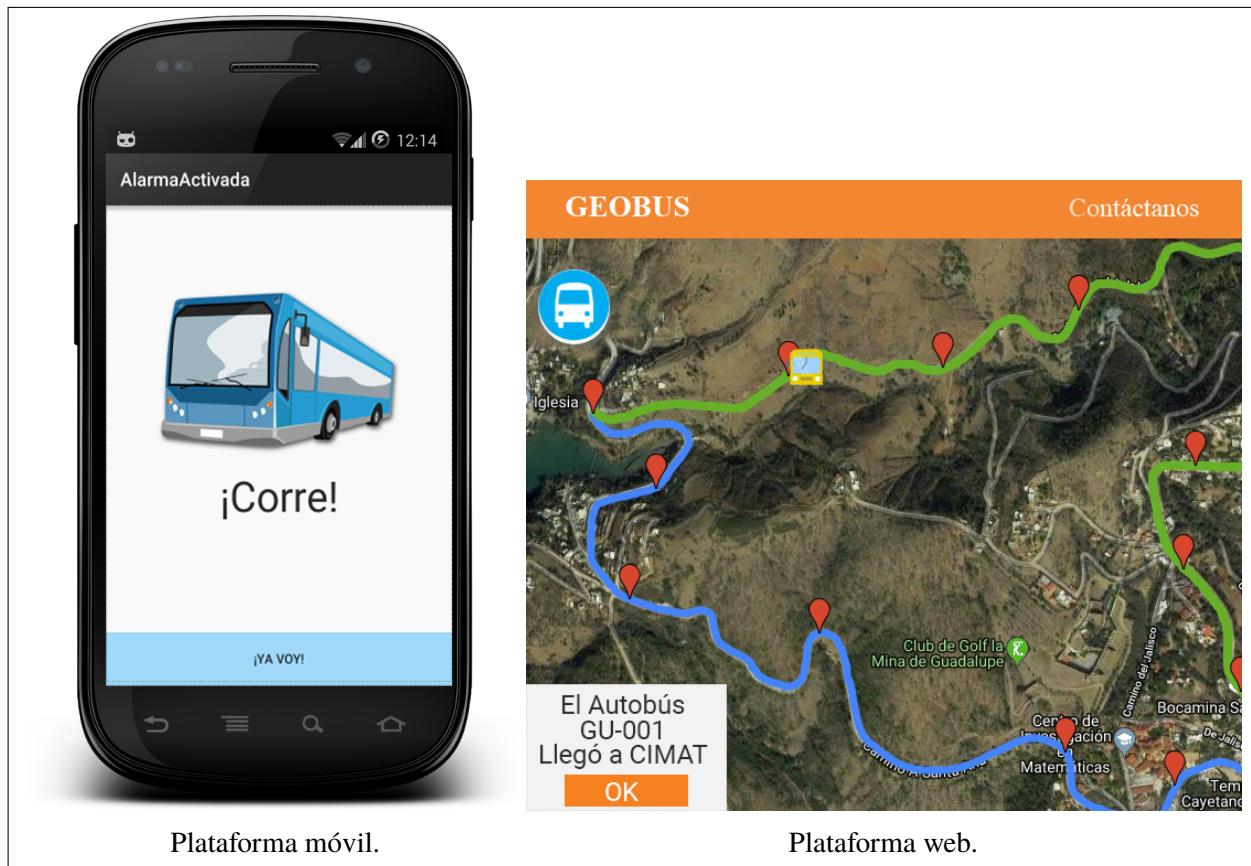


Figura 4.14.- Vistas que muestran la manera en que se despliega una alerta en ambas plataformas.

## 4.6. Comentarios finales

Para finalizar el capítulo se puede agregar que ambas implementaciones fueron diseñadas con el objetivo principal de no afectar el desempeño del dispositivo, puesto que al tomar en cuenta el más extremo de los casos, cuando existe recepción de una gran cantidad de datos en tiempo real y estos deben ser procesados y desplegados en pantalla, especialmente el dispositivo móvil es susceptible a sobrecargarse y alentarse. Es importante mencionar que el área que estudia la experiencia del usuario

o UX<sup>2</sup> señala éste como uno de los aspectos de mayor cuidado.

---

<sup>2</sup>Siglas del inglés *User-Experience*.

# Conclusiones

Para finalizar este escrito es importante mencionar que la fase de pruebas en ambiente de producción se está realizando actualmente en colaboración con la Universidad de Guanajuato, empleando los rastreadores en las unidades de transporte institucional de la misma y desplegando la información de estas a sus alumnos en ciertos horarios del día. Con esto se pretende mejorar un poco la calidad de vida de los alumnos, principalmente, al darles una herramienta extra que les ayude a evitar tener que utilizar transporte propio o un servicio privado (Taxi o UBER) al desplazarse de un campus universitario a otro.

Con base en lo antes mencionado, es viable concluir que la hipótesis y los objetivos planteados en un inicio fueron cubiertos. Definitivamente la razón principal por la que las personas no optan por utilizar el servicio de transporte público es la incertidumbre en tiempo que éste conlleva, la cual es aminorada al tener información en tiempo real del estado del sistema de transporte.

También es posible expresar que, puesto que el sistema de rastreo en cuestión fue desarrollado siguiendo el modelo más abstracto posible, representa una posibilidad muy grande de automatización en uno de los servicios más utilizados por la población mexicana, y la base para el mejoramiento del servicio al grado de países de primer nivel como Japón, Estados Unidos, Alemania y Canadá, pero con una diferencia muy importante en cuanto a la economización de recursos y presupuestos, y a la robustez y escalabilidad del proyecto.

Cabe recalcar que se prevé que esta integración con las tecnologías de la información puede también traer mejoras en otras problemáticas sociales derivadas del uso masivo de autos particulares, como el tráfico ya definido por las *horas pico* y la excesiva emisión de gases contaminantes por uso de combustibles fósiles.

El desarrollo de este sistema continua en manos de un equipo de desarrolladores de software en el CIMAT, puesto que se busca aportar otras utilidades a los usuarios, como el caso de una extensión

que permita planear las rutas de autobuses a utilizar para llegar de un lugar a otro, implementando algoritmos para encontrar el camino más corto en el grafo, tomando en cuenta que el usuario puede necesitar tomar dos rutas de autobuses no consecutivas.

Del mismo modo, se está contemplando la posibilidad de agregar la función de indicar al usuario si la siguiente unidad de transporte en pasar por el parabús de su preferencia se encuentra a su máxima capacidad de pasajeros, utilizando técnicas de visión por computadora y *crowdsourcing*<sup>3</sup>.

---

<sup>3</sup>Técnica para la obtención masiva de datos provenientes de un numeroso grupo de individuos.

# Referencias Bibliográficas

- [Alani, 2014] Alani, M. M. (2014). *Guide to OSI and TCP/IP Models*. Springer Publishing Company, Incorporated.
- [Android, 2017] Android (2017). Develop Apps | Android Developers. <https://developer.android.com/develop/index.html>. Online; accessed 20 December 2017.
- [Baset and Schulzrinne, 2005] Baset, S. and Schulzrinne, H. (2005). An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol.
- [Bouguettaya et al., 2014] Bouguettaya, A., Sheng, Q. Z., and Daniel, F., editors (2014). *Web Services Foundations*. Springer.
- [Buschmann et al., 2007] Buschmann, F., Henney, K., and Schmidt, D. C. (2007). *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Wiley, Chichester, UK.
- [Buschmann et al., 1996] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1996). *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing.
- [Cerf and Kahn, 1974] Cerf, V. and Kahn, R. (1974). A Protocol for Packet Network Intercommunication. *IEEE Transactions on Communications*, 22(5):637–648.
- [Chung, 2000] Chung, L. (2000). Client-Server Architecture. Computer Science Program, The University of Texas.
- [Cowley, 2013] Cowley, J., editor (2013). *Communications and Networking*. Springer, 2 edition.

- [Date, 2011] Date, C. J. (2011). *SQL and Relational Theory: How to Write Accurate SQL Code*. O'Reilly Media, Inc.
- [Deitel and Deitel, 2007] Deitel, P. and Deitel, H. (2007). *Internet & World Wide Web: How to Program , Fourth Edition*. Prentice Hall Press, Upper Saddle River, NJ, USA, fourth edition.
- [Duckett, 2014] Duckett, J. (2014). *HTML and CSS: Design and Build Websites*. Wiley Publishing, 1 edition.
- [Dye et al., 2007] Dye, M., McDonald, R., and Rufi, A. (2007). *Network Fundamentals, CCNA Exploration Companion Guide*.
- [Elmasri and Navathe, 2010] Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Addison-Wesley Publishing Company, USA, 6th edition.
- [Esri team, 2016] Esri team (2016). Esri - GIS Mapping Software, Solutions, Services, Map Apps, and Data. <http://www.esri.com/>.
- [Fall and Stevens, 2011] Fall, K. and Stevens, W. (2011). *TCP/IP Illustrated*. Number v. 1 in Addison-Wesley professional computing series. Addison-Wesley.
- [Ferreira et al., 2012] Ferreira, K. R., Vinhas, L., Monteiro, A. M. V., and Camara, G. (2012). Moving Objects and KML Files. In *Data Engineering Workshops (ICDEW), 2012 IEEE 28th International Conference on*, pages 355–359.
- [Fette and Melnikov, 2011] Fette, I. and Melnikov, A. (2011). The WebSocket Protocol. RFC 6455, Internet Engineering Task Force. <http://www.rfc-editor.org/rfc/rfc6455.txt>.
- [Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). RFC 2616, Hypertext Transfer Protocol – HTTP/1.1.
- [Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California. Doctor of Philosophy in Information and Computer Science.

- [Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 edition.
- [Gomaa, 2011] Gomaa, H. (2011). *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*. Cambridge University Press, New York, NY, USA, 1st edition.
- [Google, 2014] Google, I. (2014). Google Maps reference. <https://developers.google.com/maps/>.
- [Gosling et al., 2015] Gosling, J., Joy, B., Steele, G., Bracha, G., and Buckley, A. (2015). The Java Language Specification JSE8. Technical report, Oracle America, Inc., Redwood City, California.
- [Hunt, 1992] Hunt, C. (1992). *TCP/IP Network Administration*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.
- [INEGI, 2017] INEGI (2017). Estadísticas a Propósito del Día Mundial del Internet.
- [Intertech, 2014] Intertech (2014). SOA and Web Services Introduction.
- [j. Du et al., 2009] j. Du, Y., c. Yu, C., and Liu, J. (2009). A Study of GIS Development Based on KML and Google Earth. In *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pages 1581–1585.
- [Jia and Zhou, 2004] Jia, W. and Zhou, W. (2004). *Distributed Network Systems: From Concepts to Implementations*. Network Theory and Applications. Springer US.
- [Jo et al., 2003] Jo, M. H., Kim, J. B., Jo, Y. W., and Shin, D. H. (2003). Constructing the Forest Fire Extinguishment Helicopter Management System by integrating GPS and GIS. *Journal of the Korean Association of Geographic Information Studies*, 6(1):48 – 58.
- [JSON Organization, 2014] JSON Organization (2014). Introducing JSON. <https://www.json.org>.
- [Kerrisk, 2010] Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press, San Francisco, CA, USA, 1st edition.

- [Kirch, 2008] Kirch, W., editor (2008). *Geographic Information System (GIS)*, pages 494–495. Springer Netherlands, Dordrecht.
- [Kurose and Ross, 2012] Kurose, J. F. and Ross, K. W. (2012). *Computer Networking: A Top-Down Approach*. Pearson, 6th edition.
- [Li and Zhijian, 2010] Li, H. and Zhijian, L. (2010). The study and implementation of mobile GPS navigation system based on Google Maps. In *2010 International Conference on Computer and Information Application*, pages 87–90.
- [Liduo and Yan, 2010] Liduo, H. and Yan, C. (2010). Design and implementation of Web Content Management System by J2EE-based three-tier architecture: Applying in maritime and shipping business. In *2010 2nd IEEE International Conference on Information Management and Engineering*, pages 513–517.
- [McFarland, 2011] McFarland, D. (2011). *JavaScript & JQuery: The Missing Manual*. Missing Manual. O'Reilly Media.
- [Meyer, 2012] Meyer, E. (2012). *Selectors, Specificity, and the Cascade*. Oreilly and Associate Series. O'Reilly Media, Incorporated.
- [Michal Zemlicka, 2011] Michal Zemlicka, J. K. (2011). Legacy Systems and Web Services. Technical report, Charles University, Prague, Czech Republic.
- [Microsoft, 2017] Microsoft (2017). The Model-View-Presenter (MVP) Pattern. <https://msdn.microsoft.com/en-us/library/ff649571.aspx>. Online; accessed 10 December 2017.
- [MiTAC Intl, 2011] MiTAC Intl (2011). GPS Technology Explained. <http://www.mio.com/technology.htm>. 2 de febrero del 2016.
- [Neteler and Mitasova, 2008] Neteler, M. and Mitasova, H. (2008). *Open Source GIS: A GRASS GIS Approach*. Springer, 3rd edition.
- [Neustaedter et al., 2013] Neustaedter, C., Tang, A., and Judge, T. K. (2013). Creating scalable location-based games: lessons from Geocaching. *Personal and Ubiquitous Computing*, 17(2):335–349.

- [Paik et al., 2017] Paik, H.-y., Lemos, A. L., Barukh, M. C., Benatallah, B., and Natarajan, A. (2017). *Web Service Implementation and Composition Techniques*. Springer Publishing Company, Incorporated, 1st edition.
- [PANDA SECURITY, 2010] PANDA SECURITY (2010). ¿QUÉ ES PEER - TO - PEER (P2P)?  
<http://www.pandasecurity.com>.
- [Papinski and Scott, 2011] Papinski, D. and Scott, D. M. (2011). A GIS-based toolkit for route choice analysis. *Journal of Transport Geography*, 19(3):434 – 442. Special Issue : Geographic Information Systems for Transportation.
- [Pitt, 2012] Pitt, C. (2012). *Pro PHP MVC*. Apress, Berkely, CA, USA, 1st edition.
- [Pressman et al., 2003] Pressman, R., Martín, R., Jareño, I., Aguilar, L., and Galaup, V. (2003). *Ingeniería del Software: Un Enfoque Práctico*. Área de informática y computación. McGraw-Hill.
- [Quispe and René, 2011] Quispe, M. and René, N. (2011). Latitud, Longitud y Altitud. <http://nestorgeografia.blogspot.mx/2011/05/latitud-longitud-y-altitud.html>.
- [Redmond and Wilson, 2012] Redmond, E. and Wilson, J. R. (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. Pragmatic Bookshelf.
- [Robbins, 2012] Robbins, J. (2012). *Learning Web Design, 4th Edition*. O'Reilly Media, Incorporated.
- [Robinson et al., 2013] Robinson, I., Webber, J., and Eifrem, E. (2013). *Graph Databases*. O'Reilly Media, Inc.
- [Sadalage and Fowler, 2012] Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, 1st edition.
- [Sample and Ioup, 2014] Sample, J. T. and Ioup, E. (2014). *Tile-Based Geospatial Information Systems: Principles and Practices*. Springer Publishing Company, Incorporated.
- [Shalloway and Trott, 2004] Shalloway, A. and Trott, J. (2004). *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley Professional, 2 edition.

- [Shklar and Rosen, 2003] Shklar, L. and Rosen, R. (2003). *Web Application Architecture : Principles, Protocols and Practices*. John Wiley and Sons Ltd, England, 1 edition.
- [Silberschatz et al., 2006] Silberschatz, A., Korth, H., and Sudarshan, S. (2006). *Database Systems Concepts*. McGraw-Hill, Inc., New York, NY, USA, 5 edition.
- [Sommerville and Galipienso, 2005] Sommerville, I. and Galipienso, M. (2005). *Ingeniería del Software*. Fuera de colección Out of series. Pearson Educación.
- [Statista, 2017] Statista (2017). Mobile OS market share 2017. <https://www.statista.com/statistics/266136>.
- [Stephens, 2015] Stephens, R. (2015). *Beginning Software Engineering*. Wrox Press Ltd., Birmingham, UK, UK, 1st edition.
- [Stevens, 1993] Stevens, W. R. (1993). *TCP/IP Illustrated (Vol. 1): The Protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- [Taczanowska et al., 2014] Taczanowska, K., González, L.-M., Garcia-Massó, X., Muhar, A., Brandenburg, C., and Toca-Herrera, J.-L. (2014). Evaluating the structure and use of hiking trails in recreational areas using a mixed GPS tracking and graph theory approach. *Applied Geography*, 55:184 – 192.
- [Tanenbaum and Wetherall, 2010] Tanenbaum, A. S. and Wetherall, D. J. (2010). *Computer Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.
- [U.S. Air Force, 2016] U.S. Air Force (2016). Official U.S. Government information about the Global Positioning System (GPS) and related topics. <http://www.gps.gov/>. 2 de febrero del 2016.
- [Wang et al., 2013] Wang, V., Salim, F., and Moskovits, P. (2013). *The Definitive Guide to HTML5 WebSocket*. Apress, Berkely, CA, USA, 1st edition.
- [Williams, 2017] Williams, D. R. (2017). Earth Fact Sheet. <https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html>. NASA Official.

[World Wide Web Consortium, 2017] World Wide Web Consortium (2017). XML and JSON.

[https://www.w3schools.com/js/js\\_json\\_xml.asp](https://www.w3schools.com/js/js_json_xml.asp).

[Yu et al., 2018] Yu, F., Hu, X., Dong, S., Liu, G., Zhao, Y., and Chen, G. (2018). Design of a low-cost oil spill tracking buoy. *Journal of Marine Science and Technology*, 23(1):188–200.

[Zambon, 2012] Zambon, G. (2012). *Beginning JSP, JSF and Tomcat: Java Web Development*. Apress, Berkely, CA, USA, 2nd edition.

[Zeimpekis et al., 2002] Zeimpekis, V., Giaglis, G. M., and Lekakos, G. (2002). A taxonomy of indoor and outdoor positioning techniques for mobile location services. *ACM SIGecom Exchanges*, 3(4):19–27.