DEPARTAMENTO DE ELECTRONICA

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

Informe de Proyecto: “Análisis de Aplicación Móvil basada en Geolocalización”.

Grupo 14:
Daniel Fuentes
Iván Opitz
Néstor Oro

Índice

Resumen Ejecutivo

En el presente informe, se realizó un breve análisis de los requerimientos para el desarrollo de una aplicación móvil que funciona en base a la geolocalización de usuarios.

En el estudio se abarcó el estado del arte de aplicaciones similares, las funciones y protocolos necesarios en cada capa. Luego se estudiaron las diversas metodologías de geolocalización, lo que marca el punto central para la implementación de la aplicación.

Finalmente, se diseñó un algoritmo en Python para calcular distancia entre el ususario y otros clientes a través de sus coordenadas, devolviendo los “Identificadores” de aquellos que se encuentren dentro del radio de búsqueda.

Objetivo

Analizar los requerimientos en materia de redes y algoritmos necesarios, para el diseño e implementación de una aplicación móvil de “catálogo por proximidad”, basada en geolocalización.

Estado del Arte

En la actualidad, existe una gran cantidad de aplicaciones que se basan en la ubicación geográfica del usuario para ofrecerle algún servicio en particular. Tales servicios abarcan un gran abanico de posibilidades, tales como conocer personas, ofertar eventos, planificar rutas de viaje, oferta de productos locales, etc.

Algunos ejemplos de aplicaciones existentes son:

- **Sazoot:** Plataforma que realiza recomendaciones de música a través de un filtro colaborativo con usuarios de preferencias similares.

- **Tinder:** Aplicación para buscar citas a través de diversos filtros, tales como género, edad y distancia entre usuarios. Usa un algoritmo de sugerencias aleatorias para el conjunto de usuarios que cumplen los requisitos especificados en los filtros.

- **Happn:** Aplicación móvil que entrega sugerencias de perfiles de usuarios con los que el cliente se ha encontrado en lugares públicos. Su algoritmo requiere más precisión en cuanto a la distancia entre clientes

- **Foursquare**: Mapa de reseñas colaborativas entre clientes. Los usuarios califican los servicios (restaurant, concierto, cine, etc.) en base a sus propias experiencias y gustos, ayudando a otros a elegir el lugar que quieren visitar.

Aplicación propuesta:

La propuesta del grupo, se basa en fomentar al artista emergente y ayudarle a ofrecer sus productos o compartir sus obras localmente, aumentando sus oportunidades de ventas y/o contratos.

Los clientes podrán tener la oportunidad de encontrar algo original y nuevo, que se acomode a sus gustos y poder de adquisición.

La aplicación consta de dos grandes funcionalidades, dependiendo de si el usuario es un cliente o un artista:

Creación de perfil:

1) Acceder a una porción de memoria en la base de datos.
2) Crear un ID de usuario.
3) Registrar datos del usuario en la memoria.

Búsqueda:

1) Geolocalización: calcular coordenadas del usuario.
2) Almacenar ID, coordenadas y opciones de filtros en base de datos.
3) Aplicar filtros a base de datos.
4) Liberar información de perfiles que cumplen los requisitos.
5) Enviar información al usuario.

Herramientas de desarrollo:

- HTML
- JavaScript
- API - GoogleMaps
- Python

Ejemplos de Protocolos:

En la ***tabla 1***, se muestran las posibles funcionalidades que necesitaría la aplicación, protocolos que las proveen y en qué capa se usan:

Capa	Protocolos	Funcionalidades
Capa Aplicación	SSH HTTP SMTP FTP DNS	Seguridad en la conexión Navegador de internet Perfil de usuario Servicio de correo Transferencia de archivos Acceso a API google-maps
Capa Transporte	TCP	Transmisión confiable de paquetes de datos
Capa Red	IP	Direccionamiento y enrutamiento.
Capa Enlace	PPP	Autentificación conexión Cifrado de transmisión
Capa Física	802.11 WiFi	Conexión inalámbrica

Tabla 1: Análisis de protocolos y funciones de la aplicación.

Requerimientos:

- **Perfiles de usuario:**

  Cada usuario deberá crear un perfil con un número identificador, contraseña, servicio de correo y archivos que desea compartir.

  Se debe contar con una conexión segura, encriptación en la transferencia de archivos, un criterio para liberar la información de otros usuarios, corrección de errores, etc.

- **Metodologías de Geolocalización:**

  La geolocalización es el ***eje o requerimiento central del proyecto***, ya que permite obtener las coordenadas del dispositivo a través del cual el usuario accede a la red.

  Existen 3 grandes metodologías para geolocalización. Cada una tiene ventajas y desventajas, por lo que su elección debe ser en base a las necesidades de la aplicación a desarrollar. El desarrollador debe establecer dichos criterios previamente.

1) Sistemas basados en satélites (GPS)

Esta metodología consiste en la triangulación de 3 o más satélites, donde cada uno transmite tiempo y posición. La diferencia de tiempo permite calcular la distancia y finalmente obtener la ubicación del dispositivo. Un ejemplo más gráfico, se muestra en la ***figura 1.1:***

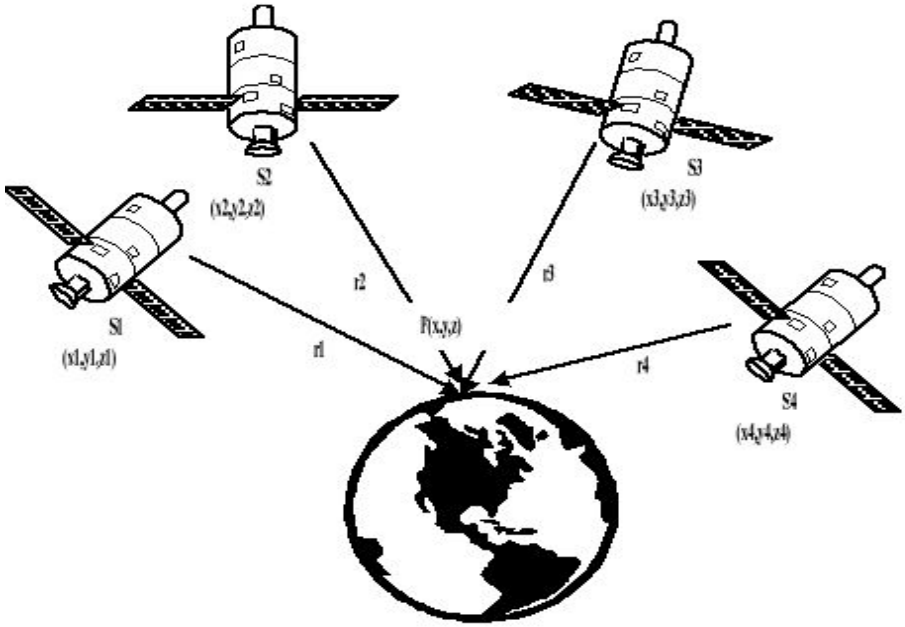

Figura 1.1: Esquema de triangulación GPS.

Ventajas: cuenta con cobertura mundial y gran precisión (hasta 2,5 metros).

Desventajas: no funciona al interior de edificios debido a que las señales de microondas no pueden atravesar paredes. Otra desventaja es que requiere un gran consumo de energía. Por último, no todos los dispositivos móviles los incorporan.

2) Redes de telefonía móvil (Cell ID)

Este mecanismo de geolocalización, está basado en la red de antenas de telefonía. Existe una base de datos con sus ubicaciones geográficas y el usuario tiene conocimiento de cuáles son las antenas más cercanas. La ***figura 1.2*** muestra un esquema de triangulación por antenas:

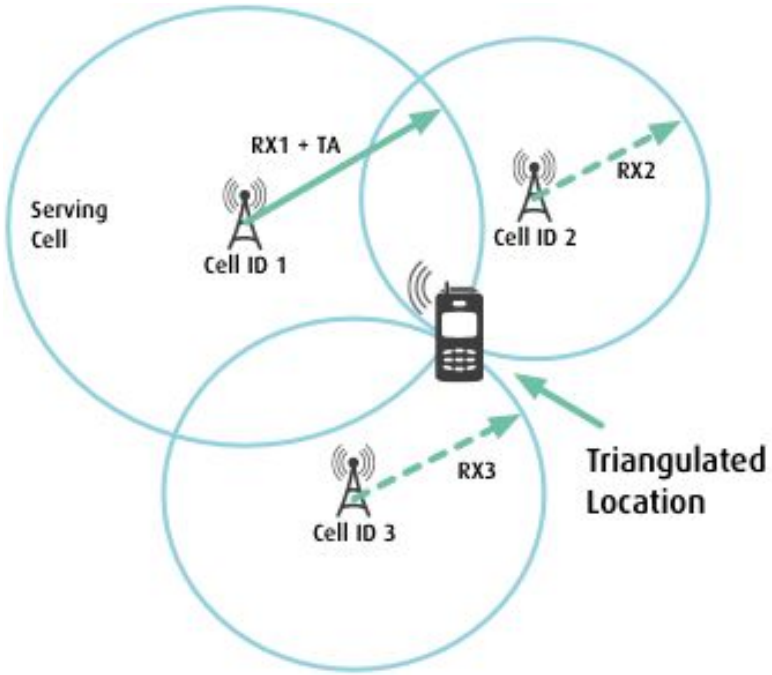

Figura 1.2: Triangulación Cell ID

Ventajas: máxima eficiencia energética, pues casi no requiere energía adicional para su funcionamiento. Otra ventaja es que posee gran cobertura, igual a la de la red telefónica. Finalmente, funciona al interior de edificios.

Desventaja: La precisión es muy baja. Alrededor de 200 metros en zonas urbanas y de 2 a 4 kilómetros en zonas rurales.

3) Redes Wifi

En la actualidad, el número de puntos de acceso WiFi, está creciendo exponencialmente. Cada punto de acceso, posee una dirección MAC única que lo identifica y un nivel de señal, lo que permite al dispositivo estimar la distancia y ubicación al acceder a varios puntos de acceso. Ver ***figura 1.3.***

Los puntos de acceso se han creado sin orden, lo que añade una incógnita extra. Para saber la ubicación de los dispositivos, se usa la información de otros usuarios que cuentan con conexión GPS.

Esta situación, hace indispensable la creación de una base de datos y una fase de entrenamiento para establecer la ubicación física de los puntos de acceso y desarrollar un método probabilístico para el cálculo de coordenadas. También permite estimar qué puntos han cambiado de la ubicación original.

La fase de entrenamiento mencionada, puede consistir en un vehículo con GPS que recorra una localidad almacenando MAC y coordenadas de los routers o puntos de acceso que detecte.

Google liberó una API que facilita el acceso a esta información para los desarrolladores de aplicaciones.

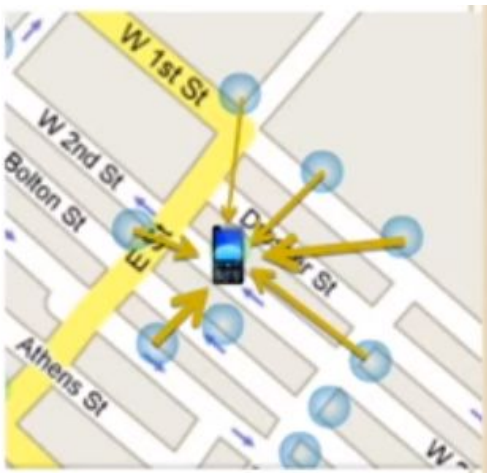

Figura 1.3: Localización por redes Wifi.

Ventajas: está disponible para todos los dispositivos con WiFi, funciona al interior de edificios y requiere un consumo moderado de batería.

Desventajas: la cobertura es sólo para áreas entrenadas. La precisión es variable y depende de los puntos de acceso y el tipo de entrenamiento.

Distancia entre usuarios:

Una vez obtenidas las coordenadas del usuario, se debe proceder a enviar al servidor, un paquete de información que contenga ***ID, Latitud, Longitud, Filtros (distancia, preferencias, etc.).*** El servidor posee un algoritmo para calcular la distancia entre las coordenadas de varios usuarios y también discrimina las preferencias de los filtros. Una vez aplicado el algoritmo, libera la información de los perfiles que cumplen los requisitos y se la envía al dispositivo del cliente.

A continuación, se muestra la ecuación que permite calcular distancia en metros, a partir de las coordenadas geográficas entre dos usuarios:

$$d = D_{TIERRA} \cdot ArcCos(Cos(Lat\ 1)\ Cos(Lat\ 2)\ Cos(Lon\ 2 - Lon\ 1) + Sin(Lat\ 1)\ Sin(Lat\ 2))$$

Donde:

D_{TIERRA} : Diámetro de la tierra en metros
Lat: Latitud expresada en radianes.
Lon: Longitud expresada en radianes.

Para expresar latitud y longitud en radianes, se debe realizar la siguiente conversión:

$$LAT_{RADIANES} = LAT_{GRADOS}\frac{\pi}{180}$$

En las ***figuras 2.1 y 2.2***, se muestra un algoritmo desarrollado en Python, para obtener una lista de usuarios que se encuentren a una distancia en metros, menor a la indicada por el cliente.

En las ***figuras 3.1, 3.2 y 3.3***, se muestran capturas de la demostración, donde se aplica el algoritmo de localización y el algoritmo para obtener ID de usuarios dentro del radio de búsqueda.

123.py - C:\Users\Daniel\Desktop\coordenadas\123.py

File Edit Format Run Options Windows Help

```
#!/usr/bin/python
import math

pi = 3.14159262653589793238 46
LAT1 = 0
LON1 = 0
LAT2 = 0
LON2 = 0

mycoordx=float(raw_input('Latitud: '))
mycoordy=float(raw_input('Longitud: '))
distancia=float(raw_input('Radio de busqueda: '))
contador=0
contador2=0
nombre=""
coordenadas=""
x=""
y=""
f=0

a=open("coord.txt")
lista=list()

for b in a:
    for c in b:
        if c==",":
            contador+=1
        else:
            if contador==0:
                nombre=nombre+c
            else:
                coordenadas=coordenadas+c
    for d in coordenadas:
        if d=="|":
            contador2+=1
        else:
            if contador2==0:
                x=x+d
            else:
                y=y+d
```

Figura 2.1: Primera parte del algoritmo de distancia.

```
    LAT1 = mycoordx*(pi/180)
    LON1 = mycoordy*(pi/180)
    LAT2 = x*(pi/180)
    LON2 = y*(pi/180)

    f=(6378137 * math.acos( math.cos( LAT1 ) * math.cos( LAT2 ) * math.cos( LON2 - LON1 ) + math.sin( LAT1 ) * math.sin( LAT2 )))

    if f <= distancia:
        lista.append(nombre)

    contador=0
    contador2=0
    nombre=""
    coordenadas=""
    x=""
    y=""
    f=0
a.close()
print (lista)
```

Figura 2.2: Segunda parte de algoritmo de distancia entre usuarios.

Ejemplos de uso:

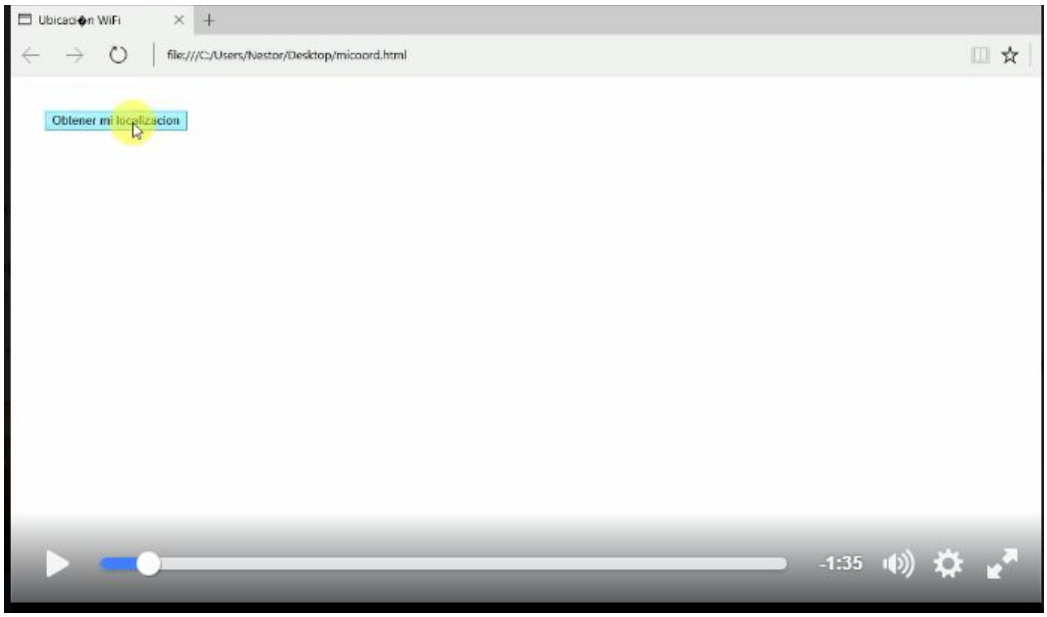

Figura 3.1: Obtener localización.

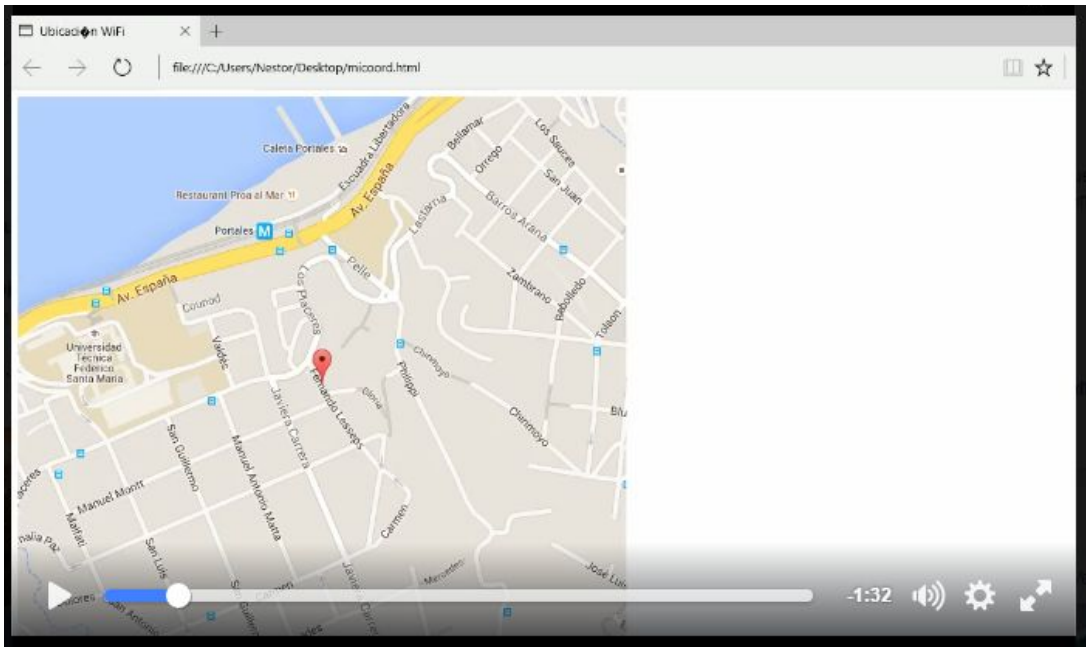

Figura 3.2: Mapa de ubicación a través de API - javascript.

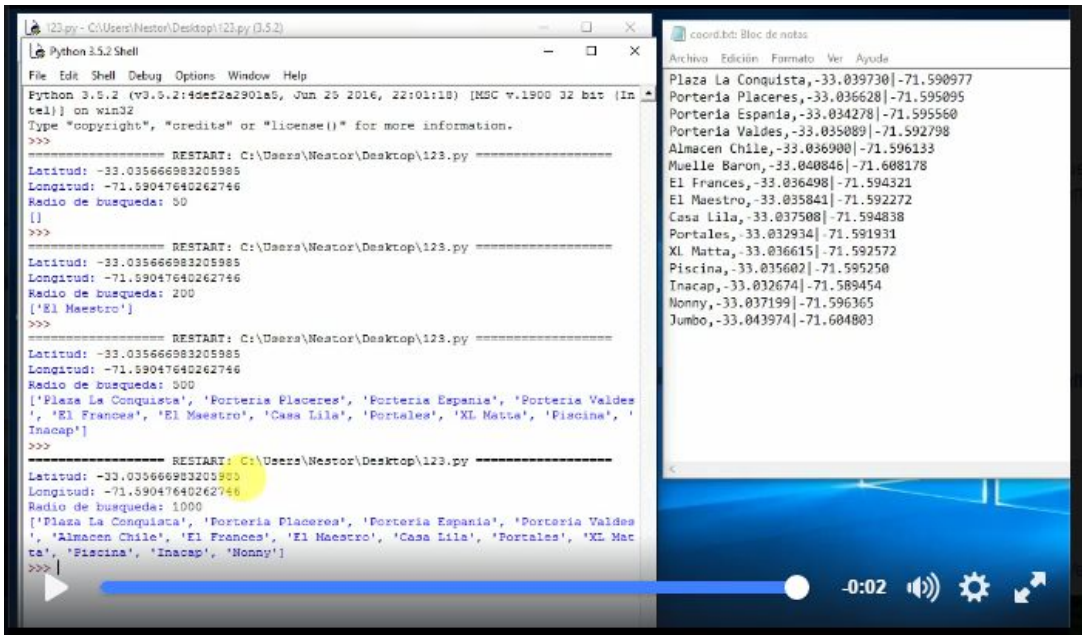

Figura 3.3: Uso de algoritmo para distintas distancias.

Conclusiones

Los contenidos y herramientas vistos en este curso, son de gran utilidad para el equipo que desee aventurarse en el desarrollo de aplicaciones web y móviles, ya que facilitan la base para la comprensión de lo que ocurre en los distintos niveles de la red y cómo establecer criterios en el uso de protocolos.

Gracias a grandes compañías como google, hoy en día se cuenta con grandes bases de datos de acceso público que facilitan la tarea a las personas naturales la tarea a la hora de implementar una idea.

Es indispensable para el desarrollador de aplicaciones, estudiar el comportamiento de las redes de internet y cómo unificar la transferencia de información de todos los niveles.

Con el fin de optimizar el tiempo y recursos, es importante consultar el estado del arte antes de iniciar la programación, ya que hay muchos problemas que ya tienen una solución disponible.

La aplicación propuesta tiene un sinfín de posibles usos, por lo que se transforma en una potente herramienta para la entrega de servicios personalizados.

Referencias

http://www.elandroidelibre.com/2016/03/aplicaciones-de-geolocalizacion.html

http://www.nosolousabilidad.com/articulos/mapas_digitales.htm

https://librosweb.es/libro/javascript/capitulo_8/menu_desplegable.html

http://www.mapanet.eu/Resources/Script-Distance.htm