Desarrollo de estacionamiento en el centro de Tandil

Integrantes: Gerez, Agustin
Lacoste,Yago
Materia : Programacion Orientada a Objetos(POO)
Profesor : Luis Berdun

Introducción

En el mundo de los smartphones, bien es sabido que cualquier sistema operativo que se aprecie debe contener una serie de aplicaciones esenciales,para ello creamos una aplicación que se basa en la localización de estacionamiento en el centro de Tandil.En ésta,aplicamos conceptos de programación orientada a objetos,como también recurrimos a la utilización de servicios de Google maps,Geolocalización,trazado de ruta, y el uso de XAMPP y MySQL para el manejo de datos.

Desarrollo

Android studio

Para el desarrollo del proyecto utilizamos el entorno de Android Studio por su excelente integración de el sdk, por su uso sencillo de manejo de archivos e interfaz y la utilización del lenguaje java.Este es un compilador para el desarrollo de aplicación que utiliza tres tipos de archivos para formar las app:Uno de los archivos que se encuentra en el paquete “Manifest” contiene archivos .XML donde se involucran los permisos que utiliza para el uso de servicios y actividades.Otro de los archivos que se encuentra en el paquete “Java” donde este contiene los código fuente de la aplicación y por el último el paquete “Res” que contiene archivos .XML que permite un diseño de la aplicación.

Cabe destacar que para la aplicación requerimos un smartphone con android 6.0 y un minsdk 23 o API: 23.

CLASES UTILIZADAS EN EL PROYECTO

Como dijimos en el paquete Java se iban a encontrar los archivos con código fuente de la app expresado en lenguaje JAVA.Para esto implementamos distintas clases:

- **MainActivity.java**: Se podría decir que esta es la clase principal del proyecto donde se encuentran dos botones, uno de ellos llamado “Estacionar”que se va a dirigir a otra actividad llamada “DatosEstacionamiento”,y el otro botón llamado “Mi auto” con la misma funcionalidad de dirigirse hacia otra actividad. Está extiende de AppCompatActivity que utilizan las funciones de la barra de acciones de la biblioteca de soporte.
  Dentro de la clase se puede encontrar el método onCreate() que su funcionamiento va a ser asignar y conectar el .XML .
  DatosEstacionamiento.java: Esta clase tiene el siguiente comportamiento: Le asignamos “editext”(texto de edición) donde el cliente coloca las calles

donde desea estacionar.Al hacer click en el botón “buscar” se inicia el comando que contiene al archivo “consultarCuadra.php” y los datos obtenido a partir del editext.A partir de los valores obtenidos, se van a colocar en un listView donde la persona podrá ver la cantidad de lugares disponibles en cada cuadra.
Al seleccionar una de las opciones del listView, se inicia el comando que contiene a “consultarLugar.php” junto al idCuadra .A partir de esta consulta se elige una latitud y longitud al azar dentro del id correspondiente que permite tener la coordenada destino requerida. Tomamos como variable estática el idCuadra ya que dicho id se asigna en esta clase y no cambia por esta razón decidimos a pesar de cierta desventajas dejarla con esta característica.

- **MapsActivityEstacionar.java:** EL comportamiento de esta clase fue mostrar el mapa completo con el trazado de la ruta desde el punto donde se encuentra la persona que quiere estacionar hasta el estacionamiento.Su implementacion fue obtener la latitud y longitud enviados de la clase “DatosEstacionamiento” e insertarlo con un marcador en el mapa.Para el problema del trazado de ruta en mapa creamos el método trazarRuta() donde por medio de una variable Json devolvía las rutas y coordenadas por donde tenía que transitar.Utilizamos los servicios de Google maps, y habilitamos la API direccion para que realice esta operación.
También esta clase implementa OnMapReadyCallback que es una Interfaz de devolución para operar sobre el mapa cuando está listo ,donde reimplementamos el método onMapReary(),a su vez extiende de FragmentActivity.
A esta clase también le agregamos un botón llamado “Guardar” que la persona al llegar al lugar debe apretarlo para guardar las coordenadas en un archivo loca llamado “ARCHIVOSP”.Dicho archivo se encuentra en modo privado, por lo tanto, solo lo puede utilizar la aplicación para su escritura y lectura. Decidimos utilizar un archivo para la persistencia de los datos cuando la app se cierre. Decidimos utilizar un botón para guardar la ubicación estacionada, hacerlo de forma manual, por el hecho que el lugar este ocupado o distintas problemáticas cotidianas.
A su vez, actualizamos los datos correspondiente de la base de datos mediante el archivo “actualizarBase.php”.
- **MapsActivity.java:** Esta clase hereda de FragmentActivity e implementa OnMapReadyCallback aplicando el método onMapReady(), ya que vamos a mostrar un mapa con un marcador sobre el punto donde la persona estaciono y el área que monitorea la app, determinando si el auto se movió. A su vez, mostramos la ubicación de la persona.

También implementamos en esta clase,es el monitoreo de una vehículo de forma automatica.Establecimos un radio de 100 metros, de acuerdo a su coordenada de estacionamiento. Además, implementamos un Hashtable<String,LatLng> como constante, con la finalidad de que se agreguen todos los parquímetros de SUMO para poder monitorear la entrada y salida de vehículos.

La API de geofencing le permite definir perímetros, también conocidos como *geofences*, que rodean las áreas de interés.Creamos una instancia de Google API Client en el método onCreate () de la actividad . Usamos la GoogleApiClient.Builder clase para agregar la LocationServicesAPI.La aplicación debe conectarse a los servicios de ubicación y luego realizar una solicitud de ubicación. Una vez que haya una solicitud de ubicación, podrá iniciar las actualizaciones regulares llamando al requestLocationUpdates().La aplicación necesita crear y agregar geofences usando la clase de constructor de la API de ubicación para crear objetos de Geofence, y la clase de conveniencia para agregarlos.

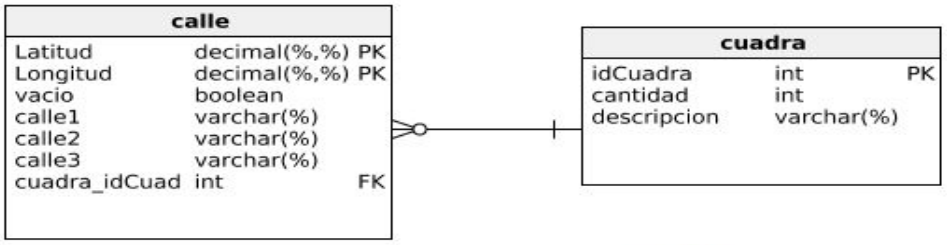
- **GeofenceRegistrationService.java:**Cuando los Servicios de ubicación detectan que el usuario ha ingresado o salido de un geofence, envía el Intent contenido en el PendingIntent que incluyó en la solicitud para agregar geofences. Esto Intent es recibido por un Service que obtiene el evento de geofencing de la intención, determina el tipo de transición de Geofence y determina cuál de las geofences definidas se activó.Esta clase lanza una notificación avisando que se fue del lugar, además de actualizar los datos de la base mediante el archivo “RestaurarDatos.php”.

Diseño de la base de datos

La base de datos debe permitir almacenar y recuperar las calles donde el cliente pueda estacionar y saber la cantidad de lugar disponible que ofrece dicha cuadra.

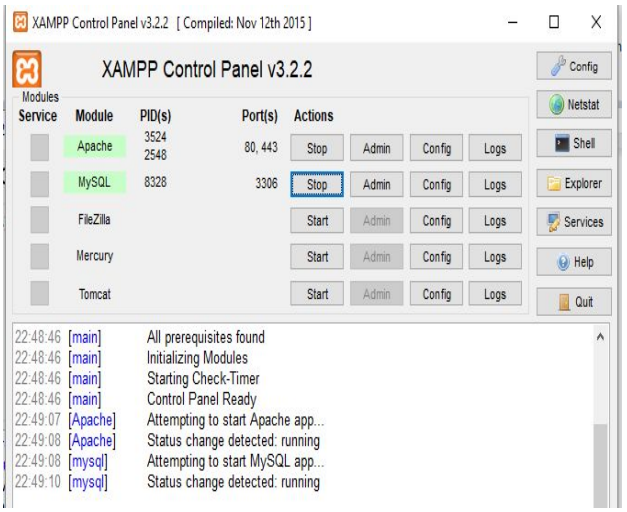
Creamos una tabla llamada “calle ”donde va a contener los atributos latitud, longitud, vacio, calle1, calle2 , calle3 y idCuadra. Dentro de esta tabla los identificadores principales primarios serían la latitud y longitud ya que son valores únicos y que no se repiten.También poseemos una clave extranjera que hace referencia a la tabla “cuadra”, donde dicha tabla contiene los atributos:idCuadra como clave primaria,cantidad y descripción,haciendo referencia a esa cuadra la cantidad de lugares libre que el usuario posee para estacionar.

Para realizar esta base utilizamos el lenguaje MySQL y junto al lenguaje PHP hacemos la conexión entre la base de datos y la aplicación mediante internet.Esta base estaría contenida en un servidor que contendrá la empresa “Sumo”.



Utilización de servidor XAMPP

Para este proyecto utilizamos un servidor virtual llamado XAMPP que es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP. Dentro de este servidor para asignar la consulta de MySQL creamos cinco archivos de .php llamados:



- functions.php: .tiene dos métodos, el primero llamado ejecutarComandoSQL donde permite de acuerdo a una sentencia SQL ejecutarla en la base del servidor conectado y el metodo conectarBase permite la coneccion entre el servidor(apache) y los comando MySQL. Todos los demas archivo, incluyen este archivo con la finalidad que puedan utilizar estos metodos.
- actualizarBase.php : a partir de los paramentros: latitud,longitud y idCuadra. Se ocupa un lugar determinado. Y se descuenta un lugar disponible en la cuadra correspondiente.
- consultarCuadra.php: a partir de tres paramentros: calle1,calle2 y calle3. Se realizan las busquedas efectivas para determinar distintos lugares donde el cliente puede estacionar, mostrandole la informacion por cuadra.
- consultarLugar.php: elige un lugar disponible para estucionar de acuerdo a un idCuadra.
- RestaurarDatos.php: realiza la funcion inversa al archivo “actualizarBase.php”, ya que cuenta un lugar mas disponible, y a su vez lo libera de estar ocupado.

Uso de xamp

Al emular en xamp, debemos tener en cuenta que la direccion ip que nos asigna como localhost cambia de acuerdo a la red donde estemos conectados. Ademas, si utilizamos Windows, debemos desactivar el firewall, en algunos casos puede ocasionar problemas.

Utilización de API de google maps

Directions API: Google Directions calcula las direcciones entre ubicaciones mediante una solicitud REST. Las indicaciones pueden especificar orígenes, destinos y puntos de paso como cadenas de texto.

SDK for android:Con el SDK de Google Maps para Android, puede agregar mapas basados en los datos de Google Maps a su aplicación. La API controla automáticamente el acceso a los servidores de Google Maps, la descarga de datos, la visualización del mapa y la respuesta a los gestos del mapa. También

puede usar llamadas a la API para agregar marcadores, polígonos y superposiciones a un mapa básico, y para cambiar la vista del usuario de un área de mapa en particular. Estos objetos proporcionan información adicional para las ubicaciones del mapa y permiten la interacción del usuario con el mapa.

Uso de volley

Volley es un cliente Http creado para facilitar la comunicación de red en las aplicaciones Android. Está totalmente enfocado en las peticiones, evitando la creación de código repetitivo para manejar tareas asíncronas por cada petición o incluso para parsear los datos que vienen del flujo externo.Tambien sirve para la obtención de una página de resultados de búsqueda como datos estructurados. Se integra fácilmente con cualquier protocolo y sale de la caja con soporte para cadenas en bruto, imágenes y JSON.

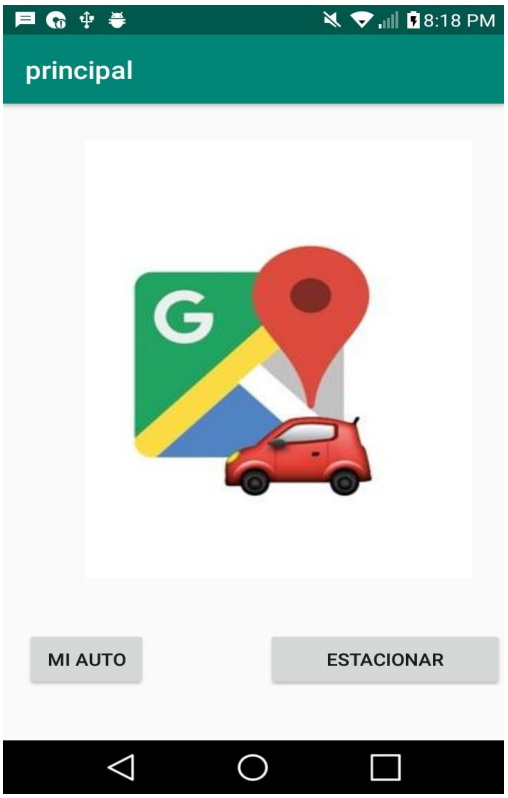
El uso de volley en la app nos facilito el acceso a la base de datos para poder trabajar con ella y su información.

Uso de location

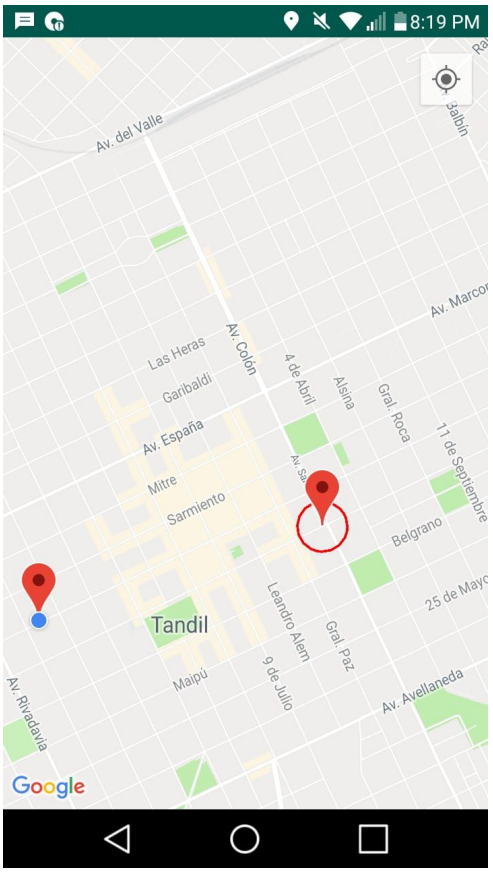
Android le da a sus aplicaciones acceso a los servicios de ubicación compatibles con el dispositivo a través de las clases en el `android.location` paquete. El componente central del marco de ubicación es el `LocationManager` que es el servicio del sistema, que proporciona API para determinar la ubicación y el rumbo del dispositivo subyacente.Al igual que con otros servicios del sistema, no crea una instancia `LocationManager` directamente. Más bien, usted solicita una instancia del sistema llamando `getSystemService(Context.LOCATION_SERVICE)`. El método devuelve un identificador a una nueva instancia `LocationManager`.

Fotos del experimento

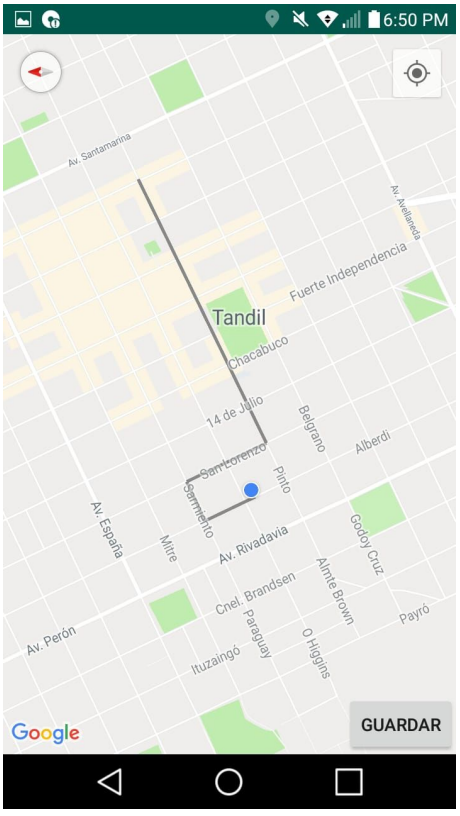
DISEÑO DE MAINACTIVITY



DISEÑO DE MAPACTIVITY



MAINACTIVITYESTACIONAR



Conclusión

Concluimos con respecto al proyecto es que no solo aplicamos los concepto de programación orientada objeto que son mucho más eficientes en el ámbito de la informática,sino que también aprendimos el manejo de android studio, la creación de aplicaciones, uso de API de google, trabajar con Google maps y sus recursos como monitoreo,geoferencia.También aplicamos conceptos de bases de datos y manejo de un servidor como XAMPP para acceder a datos mediante internet.
Otro objetivo u opinión que tuvimos es que es algo interesante trabajar con esto de la geolocalización y el mapeado tanto en aplicaciones como en programas y poder aplicarlo en la vida cotidiana.Además esta app podría servir para los parquímetros de la ciudad de Tandil ya que reconocería si alguien se fue de la ubicación sin pasar la tarjeta verificando la salida del vehículo y no solo eso sino seria mas rapido encontrar un lugar cuando se requiera.Otra idea que se podría implementar en base a esto utilizar haciéndole algunas modificaciones para utilizarlo en un centro comercial cuando se requiera ver si hay lugar de estacionamiento.