Universidad Politécnica

de Madrid

Escuela Técnica Superior de
Sistemas Informáticos

Máster Universitario En Desarrollo De Aplicaciones Y
Servicios Para Móviles

Trabajo Fin de Máster

Aplicación móvil Android basados en Google Map para facilitar los viajeros

Autor(a): Xi Chen
Tutor(a)s: Miguel Á. Manso Callejo
Ramón Pablo Alcarria Garrido

Madrid, junio 2023

Este Trabajo Fin de Máster se ha depositado en la ETSI Sistemas Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario En Desarrollo De Aplicaciones Y Servicios Para Móviles
Título: Aplicación móvil Android basados en Google Map para facilitar los viajeros
Junio 2023

Autor(a): Xi Chen
Tutor(a)s:

Miguel Ángel Manso
Ingeniería Topográfica y Cartografía
ETSI Topografía, Geodesia y Cartografía
Universidad Politécnica de Madrid

Ramón Pablo Alcarria Garrido
Ingeniería Topográfica y Cartografía
ETSI Topografía, Geodesia y Cartografía
Universidad Politécnica de Madrid

Resumen

El turismo registra un rápido crecimiento en todo el mundo y España es uno de los destinos turísticos que atrae a un gran número de visitantes. Sin embargo, viajar por una ciudad desconocida puede hacer que los visitantes se pierdan. Por ello, esta tesis de máster ha optado por explorar este tema.

El objetivo de esta investigación es profundizar en la aplicación de la API de Google Map en una aplicación Andrioid. Para ello, se desarrolló una aplicación llamada SmartMap, diseñada para ayudar a los visitantes a geolocalizar al usuario, crear una ruta entre dos puntos, buscar un lugar y buscar sitios cercanos. Además, los archivos KML importados proporcionan información a los visitantes incluso cuando no hay acceso a Internet.

Al adoptar una metodología de desarrollo ágil, el objetivo es crear consenso y entendimiento entre equipos y partes interesadas para adaptarse rápidamente a los cambios y desarrollar software con eficacia.

Para llevar a cabo este proyecto, aplicaremos los conocimientos adquiridos durante el máster, tales como Geoinformática, Desarrollo de Aplicaciones Android e Ingeniería de Software.

Abstract

Tourism is growing rapidly all over the world and Spain is one of the tourist destinations that attracts a large number of visitors. However, travelling in an unfamiliar city can lead to visitors to get lost. Therefore, this Master's thesis has chosen to explore this issue.

The aim of this research is to explore the application of the Google Map API in an Android application. For this purpose, an application called SmartMap was developed, designed to help visitors to geolocate the user, create a route between two points, search for a place and search for nearby places. In addition, imported KML files provide information to visitors even when there is no Internet access.

By adopting an agile development methodology, the goal is to build consensus and understanding between teams and stakeholders to adapt quickly to changes and develop software efficiently.

To carry out this project, we will apply the knowledge acquired during the master's degree, such as Geoinformatics, Android Application Development and Software Engineering.

Tabla de contenidos

1. Introducción

1.1. Contexto y justificación del trabajo

Más de 1.400 millones de personas viajaron por el mundo en 2018 según la Organización Mundial del Turismo (OMT). España es el segundo país del mundo que más turistas recibe: 82,7 millones en 2018. Por encima está Francia y por detrás de España se sitúan EEUU, China e Italia.

Madrid atrae a decenas de millones de turistas cada año, aquí existen algunos de los espacios abiertos más apetecibles de la capital. Suelen estar muy concurridos, estrechas y con vida. Para personas de todo el mundo, viajar por Madrid puede ser toda una aventura, ya que existe la posibilidad de que se pierdan. Por eso, nació mi interés por elegir este tema como Trabajo Final de Máster.

Por otra parte, había 5.190 millones de usuarios únicos en dispositivos móviles en el 2020, de los cuáles un 74% utilizan el sistema operativo Android, mientras que el otro 25% usan iOS. Así que Android es el sistema operativo de mayor popularidad.

A través de este proyecto, pretendo entenderlas API de Google Maps para la creación de aplicaciones para Andrioid, profundizar en cómo se utilizan en las aplicaciones.

Con el fin de adaptarse rápidamente a los cambios y ofrecer software utilizable de forma continua, los métodos de desarrollo ágil se utilizan a menudo en los procesos de gestión y desarrollo de proyectos de software. Inception Deck[1] es una herramienta de gestión ágil de proyectos que se utiliza para iniciar nuevos proyectos o fases y para crear consenso y entendimiento entre equipos y partes interesadas. En Inception Deck hay diez pasos.

1.1.1. Descripción general

SmartMap es una aplicación diseñada para turistas cuya función es ayudarles a pie en cualquier ciudad que tengas sus calles definidas en Google Maps. Para ello, la aplicación puede mostrarte restaurantes, alojamientos u otros puntos de interés cerca de donde te encuentres.

Se ha elegido utilizar Google Maps porque es una de las aplicaciones de mapas más utilizadas. Además, ofrece un servicio de geolocalización más preciso.

2. Metodología

Este trabajo de fin de máster es un proyecto con mucho trabajo detrás y, por lo tanto, es importante adoptar una metodología correcta y eficaz para el desarrollo de software con el fin de completar con éxito el proyecto en el plazo requerido. Al final decidí adoptar la metodología Mobile-D[2].

2.1. Mobile-D

Mobile-D es una metodología ágil, especialmente para el desarrollo de aplicaciones para dispositivos móviles, admite interactuar constantemente entre el equipo de trabajo con el cliente, así como de responder rápidamente a los cambios que se puedan producir durante la etapa de desarrollo del proyecto. Se basa en metodologías para el desarrollo de aplicaciones móviles conocidas pero aplicadas de forma estricta como: extreme programming, Crystal Methodologies y Rational Unified Process.

Esta metodología es adecuada para ciclos de desarrollo muy rápidos (es decir, menos de 10 semanas) en equipos pequeños. Tiene diferentes fases: exploración, inicialización, producto, estabilización y pruebas. Cada una tiene un día de planificación y otro de entrega. Por estas razones he adoptado esta metodología para este trabajo.

3. Inception Deck

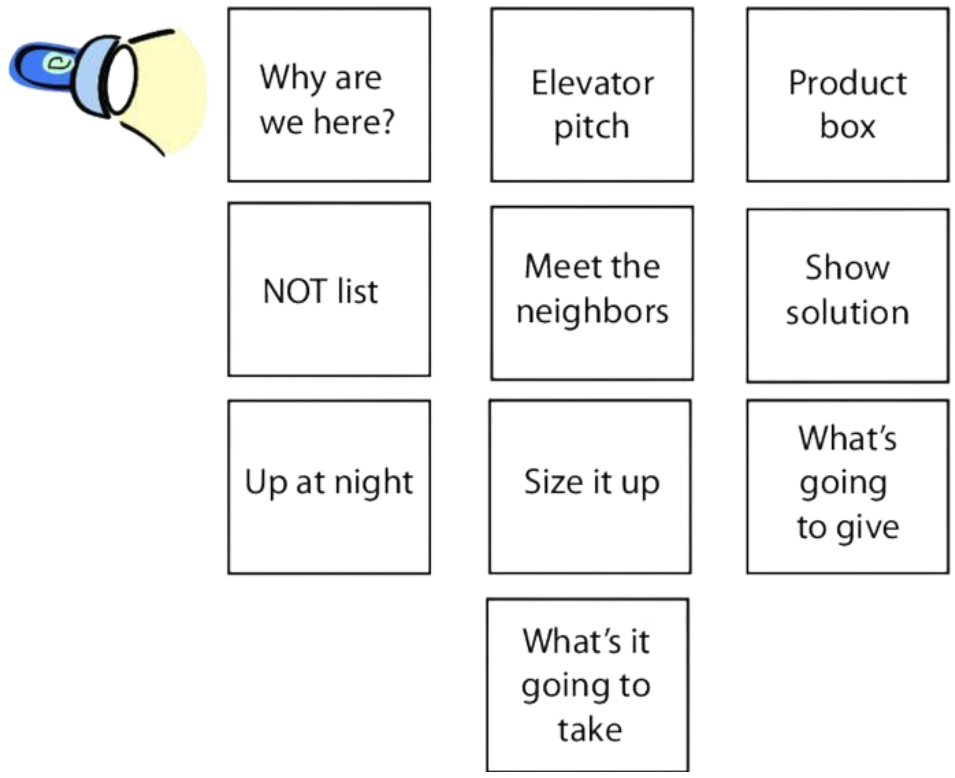

Figura 1 Recorrido por las 10 dinámicas de *Agile Inception*

3.1. Why are we here

El objetivo es el desarrollo de una aplicación destinada a dispositivos Andriod con la cual los usuarios sean capaces de geolocalizar, crear ruta en el mapa y Buscar puntos de interés en un radio de 5 km. También, los archivos KML importados pueden visualizarse en el mapa.

3.1.1. Objetivos de la aplicación

Se han identificado unos objetivos que nuestra aplicación debe alcanzar, que son siguientes:

- Importación mapas en la aplicación
- Geolocalizar al usuario
- Buscar un lugar
- Crear una ruta en el mapa
- Buscar sitios cercanos
- Leer archivos KML y mostrarlos en el mapa

3.1.2. Análisis de mercado

El método de análisis se ha realizado utilizando Google Play para buscar palabras clave que coincidieran con las introducidas en el cuadro de búsqueda. A continuación, se presentan los resultados y una tabla comparativo.

3.1.2.1. APP 1: Madrid Guía de viaje

Se trata de una aplicación que te permite consultar tu ubicación en un mapa, aunque no estés conectado a internet. Encuentra calles, direcciones y puntos de interés, y averigua en qué dirección se encuentra el lugar al que quieres ir. Debido a que el mapa y el contenido de las guías se almacenan en tu dispositivo para que los tengas siempre disponibles, incluso sin conexión a Internet. Así mismo, el posicionamiento GPS también se puede utilizar sin conexión.

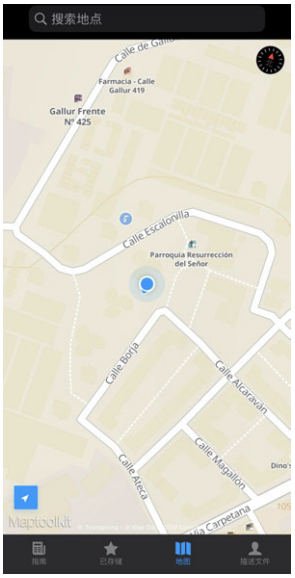

Figura 2 Interfaz Madrid Guía de viaje

3.1.2.2. APP 2: Madrid Turismo y Ocio

Se trata de una aplicación basada en OpenStreetMap que le permite descubrir de forma inmediata la ubicación e información de hoteles, campings, restaurantes, oficinas de turismo, pisos, etc. Una vez que seleccione el punto que le interesa en el mapa, obtenga las indicaciones y la ruta para llegar a su destino sólo pulsar un botón.

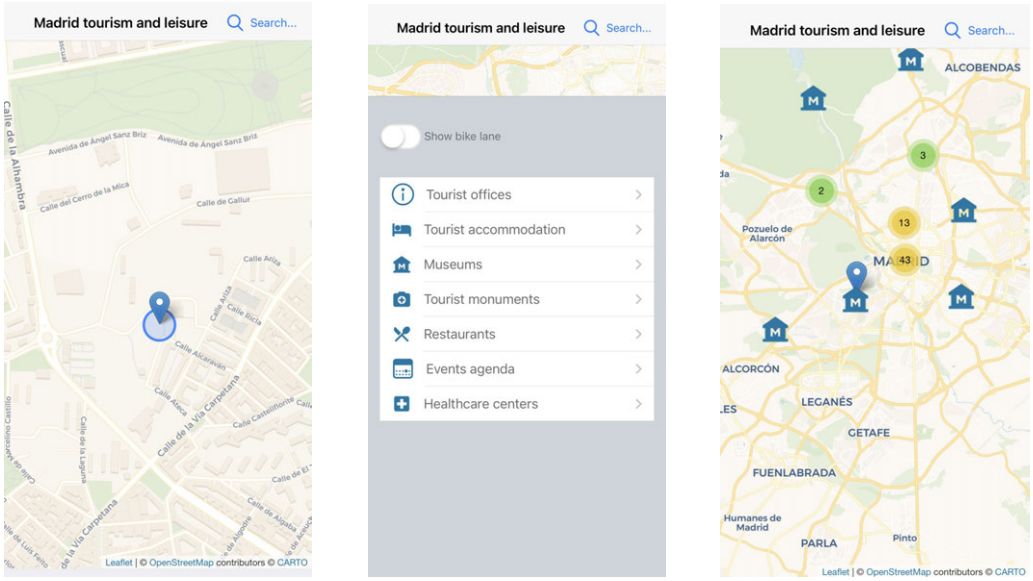

Figura 3 Interfaz Madrid Turismo y Ocio

3.1.2.3. APP 3: Madrid Guía de Viaje Offline

Se trata de una aplicación que puede utilizarse incluso sin conexión a Internet. También ofrecemos consejos para viajeros sobre cómo llegar a la ciudad, cómo desplazarse por ella, dónde dormir, dónde ir por la noche, los lugares más populares, consejos de seguridad y mucho más... La guía turística viene con un mapa regional offline de la ciudad y trae la función de GPS en los mapas.

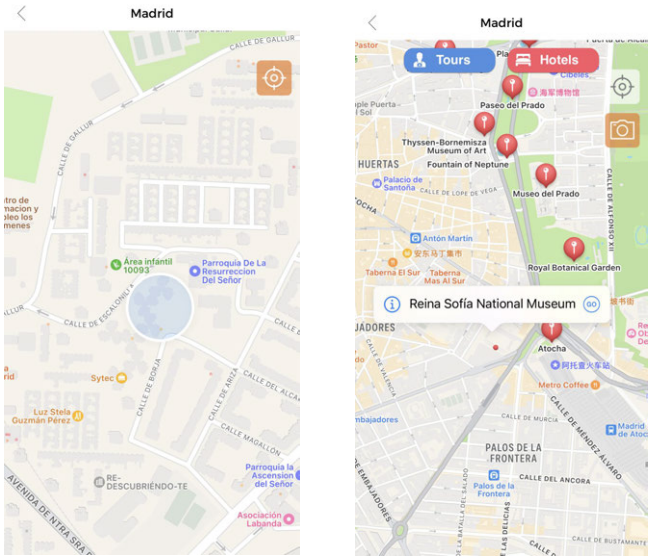

Figura 4 Interfaz Madrid Guía de Viaje Offline

3.1.2.4. Tabla comparativa

Tabla 1 Comparativa de las APP del Mercado

	APP 1	APP 2	APP 3
Posicionamiento	√	√	√
Precisión de posicionamiento	Baja	Media	Baja
Muestra el punto de su interés	×	√	√
Navegación	×	×	×
Buscar un lugar	×	×	×

3.1.3. Problema a resolver

Madrid atrae a numerosos visitantes cada año. En el centro existen algunos de los espacios abiertos más apetecibles de la capital. Suelen estar muy concurridos y por ellos pasa, a un tiempo, la vida de barrio y la de la gran ciudad. El resto de sus calles son estrechas y con vida, algunas peatonalizadas, perfectas para callejear. Por lo tanto, se necesitaba una geolocalización más precisa para completar el proyecto. La función de navegación también es esencial. Debe proporcionar una navegación muy precisa. Madrid es una ciudad llena de historia y cultura, por lo que los visitantes también deben saber qué ver cerca cuando visiten en el barrio histórico. La aplicación importa información sobre museos de todo Madrid y los visitantes pueden verlos en un mapa. También es posible encontrar sus puntos de interés a través de la función de búsqueda.

3.1.4. Solución

A diferencia de la mayoría de las aplicaciones de este tipo que utilizan GPS para la geolocalización, SmartMap utiliza Google Map Api para completar la aplicación. En cuanto a la navegación, la mayoría de las aplicaciones saltan a otras aplicaciones de navegación. Por esta razón, es capaz de crear rutas dentro de la propia aplicación. Mostrar todos los puntos de interés no favorece una visión rápida y sencilla del mapa. La aplicación también puede mostrar puntos de interés en un radio de cinco kilómetros. A diferencia de la mayoría de las aplicaciones de este tipo que utilizan GPS para la geolocalización, SmartMap utiliza Google Map Api para completar la aplicación. En cuanto a la navegación, la mayoría de las aplicaciones saltan a otras aplicaciones de navegación. Por esta razón, es capaz de crear rutas dentro de la propia aplicación. Mostrar todos los puntos de interés no favorece una visión rápida y sencilla del mapa. La aplicación también puede mostrar puntos de interés en un radio de cinco kilómetros. Importaré el archivo KML para poder visualizar puntos de

interés en el mapa. De este modo, los visitantes también pueden encontrar puntos de interés basándose en el mapa.

3.2. Elevator Pitch

Elevator pitch resuelve las siguientes cuestiones:

For: Para el turista que visita la Comunidad de Madrid

Who: Que necesita creación de rutas de alta precisión

The: SmartMap

is: Es una aplicación mapa

That: Que necesita Utilizar la función de navegación cómodamente en cualquier lugar

Unlike: Al contrario del resto de aplicaciones que requieren saltar a apps de terceros para crear la ruta

Our product: Nuestra aplicación puede crear la ruta y también calcular el tiempo y la distancia necesarios

3.3. Product Box

Una vez definido el Elevator Pitch, se realizó una lluvia de ideas sobre las características principales y los beneficios de nuestro producto. Tras ello, se definió un eslogan que representara de una manera clara qué va a ofrecer.

El eslogan es el siguiente:

"Disfruta del viaje"

A continuación, se definen los principales beneficios de SmartMap:

- Geolocalizar al usuario
- Buscar lugar
- Crear ruta en el mapa
- Buscar sitios cercanos

Por último, se enumeran algunas de las características más importantes:

- Calcular la distancia de la ruta y el tiempo necesario
- Leer archivos KML y mostrarlos en el mapa

3.3.1. Logotipo

Figura 5 Logotipo de SmartMap

Figura 6 Product Box

3.4. NOT List

Tabla 2 Una lista de Noes

In Scope	Out of Scope
Geolocalizar el usuario Buscar lugares Crear ruta en el mapa Calcular distancia de la ruta y tiempo necesario Buscar sitios cercanos Leer archivos KML y mostrarlos en el mapa	Crear y gestionar de perfil de usuario Gestionar de Cloud Compartir la ubicación con otras aplicaciones
Sin decidir	
hacer promociones publicitarias Crear y gestionar de listas Mostrar las coordenadas de su ubicación	

3.5. Meet the neighbors

En nuestro caso vamos a identificar a todos los actores involucrados en el proyecto.

- Equipo de Desarrollo
- Equipo de Diseño
- Equipo de UX
- Dueño del producto
- Equipo de Analítica
- Departamento de Marketing

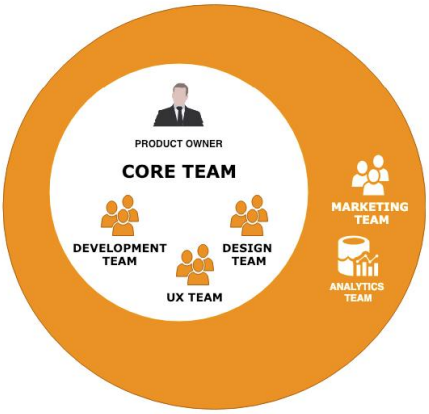

Figura 7 Los actores involucrados

3.6. Show solution

La solución como se ha dicho previamente va a ser una aplicación móvil en Andriod mediante el lenguaje de programación Java en el entorno de desarrollo Andriod Studio.

Se construye una URL de solicitud que cumpla con las especificaciones de la API de Google Maps, según las funcionalidades y parámetros deseados. Se envía la solicitud HTTP URL construida al servidor de la API de Google Maps. El servidor de la API de Google Maps recibe la solicitud del cliente y la procesa según los parámetros y funcionalidades especificados. El servidor verifica la clave de API de la solicitud, analiza los parámetros de la solicitud, realiza las operaciones correspondientes y genera los datos de respuesta. El servidor genera los datos de respuesta en formato JSON.

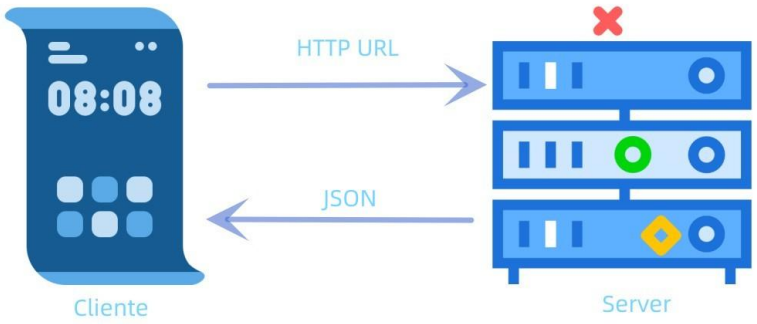

Figura 8 La solución

El ejemplo de búsqueda de sitios cercanos:

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=40.3958131,-3.7407044&radius=500&type=museum&sensor=true&key=YOUR_API_KEY

En el siguiente ejemplo, se muestra una respuesta de Nearby Search:


```
{
   "html_attributions" : [],
   "results" : [
      {
         "business_status" : "OPERATIONAL",
         "geometry" : {
            "location" : {
               "lat" : 40.393011,
               "lng" : -3.740870499999999
            },
            "viewport" : {
               "northeast" : {
                  "lat" : 40.3943392802915,
                  "lng" : -3.739493219708498
               },
               "southwest" : {
                  "lat" : 40.3916413197085,
                  "lng" : -3.742191180291502
               }
            }
         },
         "icon" : "https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/museum-71.png",
         "icon_background_color" : "#13B5C7",
         "icon_mask_base_uri" : "https://maps.gstatic.com/mapfiles/place_api/icons/v2/museum_pinlet",
```

```
            "name" : "Yacimiento Paleontol\u00f3gico Metro Carpetana",
            "opening_hours" : {
               "open_now" : true
            },
            "photos" : [
               {
                  "height" : 4000,
                  "html_attributions" : [
                     "\u003ca
href=\"https://maps.google.com/maps/contrib/109344479921080933892\"\u003eCarlos
Uzcategui\u003c/a\u003e"
                  ],
                  "photo_reference" :
"AZose0nhPUmLfLtC7n4kOI-H9iMxeZ0gy0i8gxUPSQOBS7tpwmvMt1v7twSHRREmLBj32KyVFqFKeiU-AdM
uaS9_wn9nbTCgXPaega0ccn1prhtVn9ukMbMmIbtfwg1DgdhaPtn6_kuktc_xUhysbxymQHOMWp5lG_SQYpN
FSPK_UXvByPDd",
                  "width" : 3000
               }
            ],
            "place_id" : "ChIJY5kwjN0nQg0RP2hwD8trIPM",
            "plus_code" : {
               "compound_code" : "97V5+6M \u897f\u73ed\u7259\u9a6c\u5fb7\u91cc",
               "global_code" : "8CGR97V5+6M"
            },
            "rating" : 4,
            "reference" : "ChIJY5kwjN0nQg0RP2hwD8trIPM",
            "scope" : "GOOGLE",
            "types" : [ "museum", "point_of_interest", "establishment" ],
            "user_ratings_total" : 8,
            "vicinity" : "Calle V\u00eda Carpetana, 141, Madrid"
         }
      ],
      "status" : "OK"
}
```


3.7. Up at night

En este caso, se ha hecho una clasificación de los riesgos que podemos encontrarnos en la realización de nuestra aplicación. Para ello, se han ordenado por prioridad, tal y como se puede ver en la tabla siguiente:

Tabla 3 Una matriz de riesgo

RIESGOS	SIN RIESGOS	
Mala precisión de posicionamiento	Es interesante para el Mercado	+
Navegación incorrecta	Costo del producto elevado	prioridad
Poco conocimiento de las Tecnologías	Mal Análisis de Requisitos	
Mala Gestión de Tiempo	Pueden aparecer distintos rivales que compiten con nosotros	−

3.8. Size it up

Para llevar a cabo esta técnica se ha construido un Gantt en el que han colaborado todos los equipos marcando ciertos hitos del proyecto con entregas funcionales.

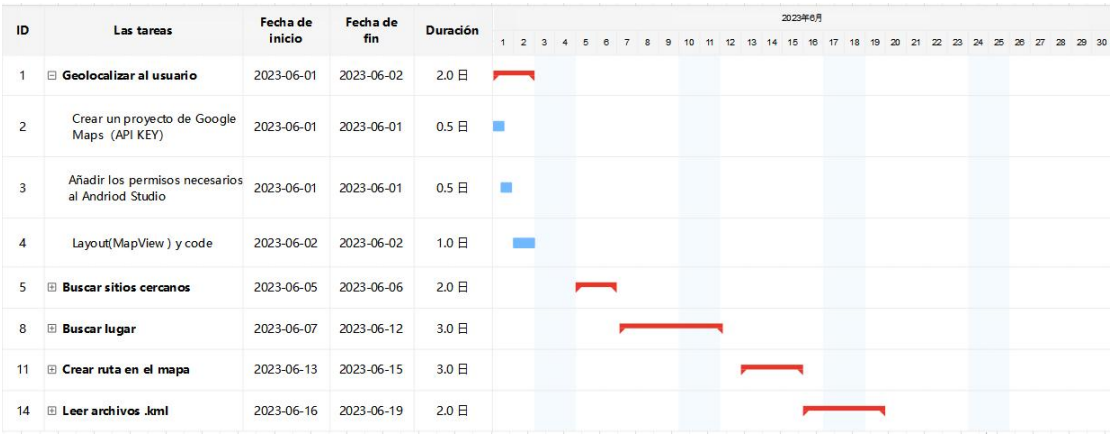

Figura 9 Diagrama Gantt

3.9. What's going to give

A continuación, se muestra un gráfico de aquellos factores que influyen en el proyecto y su importancia:

Se ha decidido que la prioridad será asignada a través del reparto de 100 puntos en diferentes recursos resultando en que a cuanta mayor puntuación mayor prioridad.

Tabla 4 Algunas prioridades

Seguridad	20
Alcance	5

Tiempo	15
Presupuesto	10
Calidad	15
Facilidad de uso	15
Comunicación	10
Riesgos	10
Total	100

3.10. What's going to take

En esta dinámica se trata de calcular aproximadamente el número de personas involucradas para cumplir con las necesidades en el tiempo establecido y estimar el coste que pueda conllevar.

Para la realización del proyecto se ha estimado un presupuesto en función del

coste por personal * la duración del proyecto.

Así pues, se ha estimado que el cobro por personal de desarrollo en 1800 €/mes. Como puede verse en el diagrama de Gantt, el proyecto tarda 12 días en completarse.

Coste Total estimado: 12/22*1800*1=990.

4. Historias de usuario

4.1. Geolocalizar al usuario

4.1.1. Card

Como usuario,
quiero ser capaz de geolocalizar la ubicación del usuario,
para poder marcar la ubicación del usuario y mostrar un mensaje que indica la ubicación actual. cuando entra a utilizar la APP.

- **Título**: “Geolocalizar al usuario”

- **Prioridad**: 10 （10 es la máxima prioridad）

- **Esfuerzo de trabajo estimado (días ideales)**: 2

4.1.2. Conversation

4.1.2.1. Texto de la conversación

Cuando el usuario entra en la aplicación y se le pregunta si permite que esta aplicación acceda a la ubicación de este dispositivo. Una vez seleccionado el consentimiento, la ubicación del usuario se marca inmediatamente. Y en la parte inferior de la pantalla aparecen un mensaje que indica la ubicación actual.

4.1.2.2. Wireframe

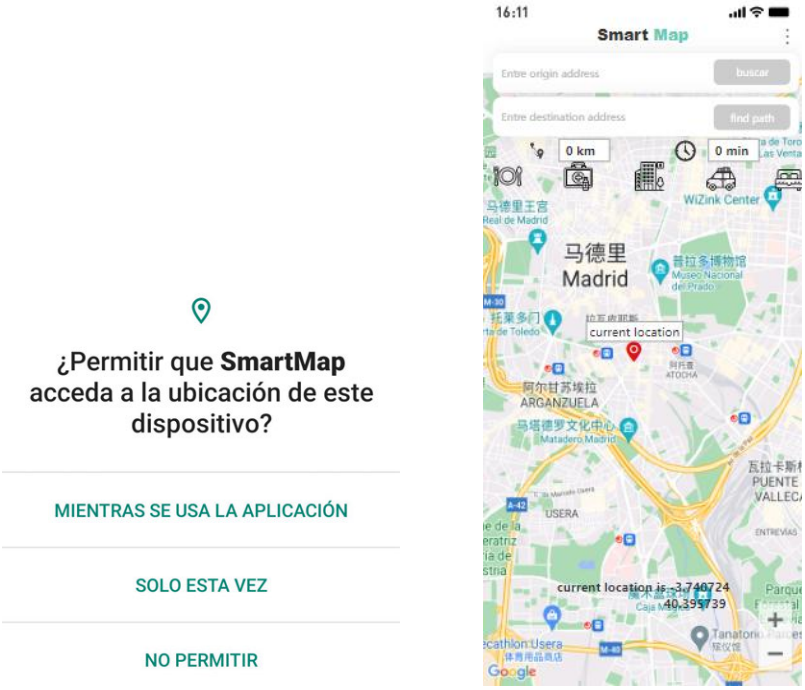

Figura 10 Wireframe

Inputs: En este caso, no es necesario introducir contenidos.

Gestures: Respecto a los gestos que el usuario realiza para interactuar con la aplicación, hacen clic el marcador rojo del mapa, y aparece un recordatorio de la ubicación actual del usuario ("current location").

Annotation:

Layout:En este caso se querrá tener el layout dinámico para que mantenga el marcador de Geolocalización del usuario en el centro del mapa.

Motion: El movimiento del dispositivo no implicará ningún cambio en la disposición de los elementos en pantalla debido a que la aplicación se mostrará bloqueada únicamente en la versión vertical (no existe una disposición en formato horizontal).

4.1.2.3. Tareas

TAREA 1: Añade mapas a la aplicación Android

En Google Cloud Platform, se crea un proyecto y se activa Maps SDK for Android API, que le proporcionará una clave API para acceder a la API de Google Maps. Entonces se copia tu API key e ingresarla en el archivo value\google_maps_api de tu aplicación.

google_maps_api.xml (debug)

```
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">API-KEY</string>
```

Se abre su proyecto en Android Studio y Se asegura de que se han añadido los permisos necesarios al archivo "AndroidManifest.xml".

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

En el archivo build.gradle del proyecto, se asegura de que se han añadido las dependencias de los servicios de Google Play. Se utiliza 'com.google.android.gms:play-services-maps:18.0.2' para insertar el mapa, y otra dependencia para los servicios de ubicación.

```
implementation 'com.google.android.gms:play-services-maps:18.0.2'
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

En su archivo de diseño, añade un elemento para mostrar el mapa

```
android:id="@+id/map"
android:name="com.google.android.gms.maps.SupportMapFragment"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

En su clase Activity, se implementa la interfaz OnMapReadyCallback y se obtiene la instancia del mapa en el método onCreate().

```
public class App extends AppCompatActivity implements
        OnMapReadyCallback,
        DirectionFinderListener{
private GoogleMap mMap;
.....
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_app);
//Obtén el SupportMapFragment y recibe una notificación cuando el mapa esté listo para ser utilizado.
    SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
```

```
    UiSettings uisettings = mMap.getUiSettings();
    uisettings.setZoomControlsEnabled(true); //Controlar la escala de zoom del mapa
}
```

TAREA 2: Permiso de solicitud de ubicación

Crea un método "getCurrentLocation()" para obtener la ubicación. En primer lugar, comprueba los permisos y, si no está autorizado, pide al usuario que concede permisos de localización a la aplicación.

```
private void getCurrentLocation(){
    if(ActivityCompat.checkSelfPermission
            (this,Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED
            && ActivityCompat.checkSelfPermission
            (this,Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},Request_code);
        return;
    }
}
```

Crea un método override onRequestPermissionsResult. Si la longitud de la matriz grantResults es mayor que 0 y el primer elemento (índice 0) es igual a PackageManager.PERMISSION_GRANTED, entonces el usuario ha concedido el permiso. En este caso, se llama al método getCurrentLocation().

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull
int[] grantResults) {
    switch (Request_code){
        case Request_code:
            if(grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){
                getCurrentLocation();
            }
    }
}
```

TAREA 3: Marca la ubicación en el mapa

Después de comprobar los permisos de localización locationpermisson, crea un objeto de solicitud de localización.

LocationServices es una clase de Google Play Services. Llamando a getFusedLocationProviderClient(this.getApplicationContext()) se puede obtener una instancia del objeto FusedLocationProviderClient. Esta instancia se puede utilizar para

solicitar información de ubicación del dispositivo, escuchar actualizaciones de ubicación y realizar otras operaciones relacionadas con la ubicación.

```
fusedLocationProviderClient=
LocationServices.getFusedLocationProviderClient(this.getApplicationContext());
```

LocationCallback() Devolución de llamada de ubicación para recibir notificaciones de la API de fusedLocationProviderClient.

```
private void getCurrentLocation(){
……
LocationCallback locationCallback = new LocationCallback(){……}
}
```

Verifica si el objeto de ubicación (Location) no es nulo. Cuando no es nulo,obtiene sus valores de latitud y longitud. Crea una variable para la latitud y la longitud, y añade un marcador aquí.

```
if(location != null){
    lat = location.getLatitude();
    lng = location.getLongitude();
    LatLng latLng = new LatLng(lat,lng);
    mMap.addMarker(new MarkerOptions().position(latLng).title("current location"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
```

TAREA 3: Muestra un mensaje que indica la ubicación actual en la parte inferior de la pantalla.

Crea un método override onLocationResult que se ejecutará cuando se disponga de información sobre la ubicación.

Escribe una condición para comprobar si el resultado es nulo. Si la solicitud de ubicación es nula, significa que no se puede obtener la información de ubicación actual. Se ha obtenido información de ubicación válida, muestra un mensaje Toast que indica la ubicación actual.

```
LocationCallback locationCallback = new LocationCallback(){
    @Override
    public void onLocationResult(LocationResult locationResult) {
        Toast.makeText(getApplicationContext(),"location result is=" +
locationResult,Toast.LENGTH_LONG).show();
        if(locationRequest == null){
            Toast.makeText(getApplicationContext(),"Current location is
null",Toast.LENGTH_LONG).show();
            return;
        }
        for(Location location:locationResult.getLocations()){
            if(location!=null){
                Toast.makeText(getApplicationContext(),"Current location is" +
location.getLongitude(),Toast.LENGTH_LONG).show();
```

```
            }
        }
    }
};
```

4.1.3. Confirmation

4.1.3.1. Criterios de Aceptación (Given - When - Then)

Feature: Validación de la correcta localización del usuario

Scenario: Una vez en la aplicación, la ubicación del usuario se marca en el mapa.

Given	El teléfono está conectado a la red
When	Se Acceden a esta aplicación
Then	Se marcan en el mapa y se muestran un mensaje Toast que indica la ubicación actual.

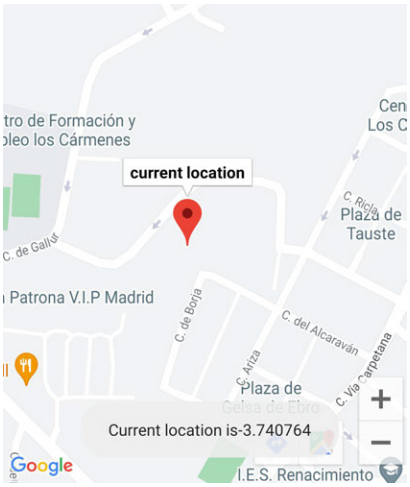

Figura 11 El resultado

4.2. Buscar sitios cercanos

4.2.1. Card

Como usuario,
quiero ser capaz de buscar sitios cercanos,
para poder mostrar puntos de interés en un radio de cinco kilómetros desde la ubicación del usuario.

- **Título**: “Buscar sitios cercanos”

- **Prioridad**: 7 （10 es la máxima prioridad）

- **Esfuerzo de trabajo estimado (días ideales)**: 2

4.2.2. Conversation

4.2.2.1. Texto de la conversación

Los usuarios pueden explorar las zonas cercanas. Una vez que el usuario haga clic en el icono de un restaurante, hospital, etc., se buscará un lugar en un radio de cinco kilómetros basado en la ubicación actual del usuario que coincida con los criterios. Todos los resultados se marcarán en el mapa y también se indicará el nombre del sitio.

4.2.2.2. Wireframe

Figura 12 Wireframe

Inputs: En este caso, no encontramos ningún input. Por ejemplo, el icono "hotel" será el que dará
inicio a buscar hoteles cercanos.
Gestures: El único gesto que se tendrá es el de presionar sobre el icono en la pantalla. Se ha utilizado un icono para el botón con el fin de que no sea necesario ningún tipo de anotación.

Annotation: El *layout* será *responsive* (se ajustará a todo tipo de dispositivos móviles). No se hará uso del teclado, razón por la cual el mismo se encontrará oculto.

Motion: Nuestra aplicación funcionará únicamente en la versión vertical, y en este caso, al girar el dispositivo, no se podrá visualizar de forma horizontal.

Endpoints: En el caso de la *Nearby Search* API, el *endpoint* es una URL específica que se utiliza para solicitar datos sobre ubicaciones cercanas. puede construir una URL con los parámetros necesarios y enviar una solicitud a ese *endpoint* para obtener información sobre sitios cercanos.

4.2.2.3. Tareas

TAREA 1: Añade mapas a la aplicación Android.

En la historia de usuario anterior, ya sabíamos cómo conseguirlo.

Sin embargo, debe ser consciente de que es necesario añadir una dependencia a build.gradle para leer archivos JSON. Y en Google Cloud Platform, se activa Places API.

```
implementation 'com.android.volley:volley:1.1.0'
```

TAREA 2: Inserta iconos en un diseño lineal.

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/hotel"
    android:padding="20dp"
    android:src="@drawable/ic_baseline_museum_24"/>
```

TAREA 3: Utiliza Nearby Search con la API de Google Places para Buscar sitios cercanos.

En primer lugar, defina el botón de imagen. Luego,Utilice el método findViewById para encontrar la vista.

```
ImageButton hotel;
hotel= findViewById(R.id.hotel);
```

Establece un escucha de eventos (OnClickListener) en la vista llamada restaurant que es una vista que pueda ser pulsada).

```
restaurant.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { }
});
```

En el método anterior, se compone una cadena sobre la URL. Una solicitud de búsqueda de sitios cercanos es una HTTP URL como la siguiente:

"https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters[3]"

Parámetros requeridos：

-location： El punto alrededor del cual recuperar la información del lugar. Debe especificarse como latitud, longitud.

-radius:Define la distancia (en metros) dentro de la cual devolver resultados de lugar. Puede sesgar los resultados a un círculo especificado pasando un parámetro de ubicación y un parámetro de radio.

-type:Restringe los resultados a los lugares que coincidan con el tipo especificado.

```
StringBuilder stringBuilder = new
StringBuilder("https://maps.googleapis.com/maps/api/place/nearbysearch/json?");
stringBuilder.append("location=" + lat+ "," + lng);
stringBuilder.append("&radius=5000");
stringBuilder.append("&type=hotel");
stringBuilder.append("&sensor=true");
stringBuilder.append("&key="+getResources().getString(R.string.google_maps_key));
```

Utilice la clase HttpURLConnection para establecer una conexión con una URL especificada y recuperar datos de la URL a través de esa conexión.

```
httpURLConnection =(HttpURLConnection) getURL.openConnection();
httpURLConnection.connect();
```

Obtiene el flujo de entrada de la conexión. El flujo de entrada asociado a la conexión se obtiene llamando al método getInputStream, que es el flujo de datos leído de la URL, y asignándolo a la variable inputStream.

```
inputStream = httpURLConnection.getInputStream();
```

Utilice el objeto BufferedReader para leer los datos línea por línea y añadirlos al objeto StringBuffer. Lea los datos de caracteres línea por línea llamando al método readLine y luego añada cada línea al objeto StringBuffer.

```
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
    StringBuffer sb = new StringBuffer();
    String line = "";
    while((line = bufferedReader.readLine()) != null ){
        sb.append(line);
    }
    URLData = sb.toString();
    bufferedReader.close();
}catch (Exception e){
    Log.d("Exception",e.toString());
}finally {
    inputStream.close();
    httpURLConnection.disconnect();
}
```

TAREA 3: Los resultados muestran puntos de interés cercanos en la aplicación.

Crea una clase que ayude a obtener datos sobre lugares cercanos. Crea tareas asíncronas para realizar solicitudes de red y operaciones de obtención de datos en un subproceso en

segundo plano con la ayuda de la clase DownloadURL. Recuperar datos de la URL y devolverlos a Google para ubicaciones cercanas.

```
@Override
protected String doInBackground(Object... objects) {
    try{
        googleMap = (GoogleMap) objects[0];
        URL = (String) objects[1];
        DownloadURL downloadURL = new DownloadURL();
        googleNearByPlacesData = downloadURL.retireveURL(URL);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return googleNearByPlacesData;
}
```

Crea otro método *onPostExecute*. Después de recuperar los datos sobre las ubicaciones cercanas, los datos se pasan al método *onPostExecute*, con el fin de analizar los datos obtenidos de la URL en formato JSON. Recorre la matriz *jsonArray* para obtener la longitud, latitud, nombre, etc. de cada elemento de la matriz. Todos los resultados están marcados en el mapa.

```
@Override
protected void onPostExecute(String s) {
    try{
        JSONObject jsonObject = new JSONObject(s);
        JSONArray jsonArray    = jsonObject.getJSONArray("results");

        for(int i=0; i<jsonArray.length();i++){
            JSONObject jsonObject1 = jsonArray.getJSONObject(i);
            JSONObject getLocation =
jsonObject1.getJSONObject("geometry").getJSONObject("location");

              String lat = getLocation.getString("lat");
              String lng = getLocation.getString("lng");
              JSONObject getName = jsonArray.getJSONObject(i);
              String name =    getName.getString("name");
            LatLng latLng = new
LatLng(Double.parseDouble(lat),Double.parseDouble(lng));
            MarkerOptions markerOptions = new MarkerOptions();
            markerOptions.title(name);
            markerOptions.position(latLng);
markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE));
            googleMap.addMarker(markerOptions);
          googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng,15));
        }
```

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

4.2.3. Confirmation

4.2.3.1. Criterios de Aceptación (Given - When - Then)

Feature: Verificar todos los hoteles en un radio de cinco kilómetros de la ubicación del usuario.

Scenario: Haciendo clic en el icono del hotel, se pueden marcar en el mapa todos los hoteles situados en un radio de cinco kilómetros de la ubicación del usuario.

Given	El teléfono está conectado a la red
When	Se hacen clic en el icono del hotel
Then	Se marcan todos los hoteles situados a menos de cinco kilómetros de la ubicación del usuario en y se pueden indicar sus nombres.

Utilice la API de búsqueda de lugares cercanos para solicitar datos sobre lugares cercanos. Los datos de respuesta de la API pueden incluir información como el nombre, la dirección y las coordenadas de latitud y longitud de los lugares cercanos.

https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=40.3957637,-3.7407371&radius=5000&type=hotel&sensor=true&key=YOU-KEY

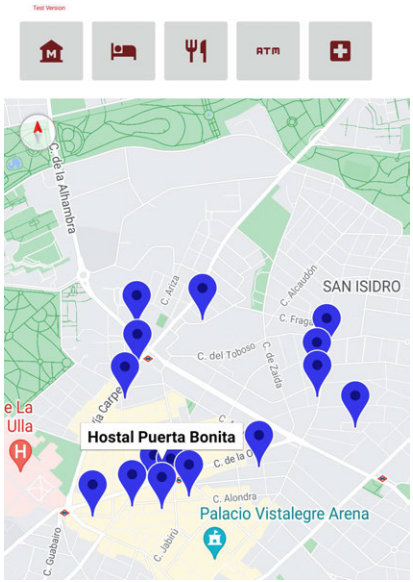

Figura 13 El resultado

4.3. Buscar un lugar

4.3.1. Card

Como usuario,
quiero ser capaz de buscar un lugar,
para poder marcar el resultado de la búsqueda en el mapa con un marcador rojo.

- **Título**: “Buscar un lugar”

- **Prioridad**: 7 （10 es la máxima prioridad）

- **Esfuerzo de trabajo estimado (días ideales)**: 3

4.3.2. Conversation

4.3.2.1. Texto de la conversación

El usuario introduce el lugar que desea visitar en el campo de texto y, a continuación, pulsa el botón de la derecha. Finalmente, el lugar se marca en el mapa. También se muestra el nombre del lugar.

4.3.2.2. Wireframe

Figura 14 Wireframe

Inputs: Se Introduce el lugar a la que desea ir en el campo de texto.
Gestures: Se tendrá que presionar sobre el campo de texto. Luego, el teclado se activará. A continuación, Deberá hacer clic en el botón de la derecha.
Motion: Si bien es *responsive* solo se diseña para orientación vertical.

Annotation:Aunque hay dos cuadros de texto, sólo puede introducir el lugar en el superior.

Endpoints: Para buscar un lugar, es necesario realizar una solicitud al punto de salida "Google Maps Geocoding API".

4.3.2.3. Tareas

TAREA 1: Añade mapas a la aplicación Android.

Utiliza Geocoding API para Buscar un lugar. Es necesario añadir la dependencia a build.gradle . Y en Google Cloud Platform, se activa Geocoding API.
TAREA 2: Escribe el nombre del lugar uno por uno.

Obtiene el texto introducido por el usuario en EditText y lo convierte en una cadena utilizando el método toString(). La información de localización se almacena en la variable location. Puede utilizar esta variable para posteriores operaciones de geocodificación o de otro tipo.

```
String location = etLocation.getText().toString();
```

TAREA 2: Hace clic en el botón de la derecha
La API[4] de Geocoding es un servicio que proporciona geocodificación inversa y geocodificación inversa de direcciones. La geocodificación es el proceso de convertir direcciones (por ejemplo, direcciones de calles) en coordenadas geográficas (por ejemplo, latitud y longitud) que puede utilizar para colocar marcadores en mapas o para posicionar mapas.

Una solicitud a la API de Geocoding tiene el siguiente formato:

"https://maps.googleapis.com/maps/api/geocode/outputFormat?parameters"

```
//construir URL
String URL = "https://maps.googleapis.com/maps/api/geocode/json?";
try {
 // Codificación de caracteres especiales como espacios en la entrada del usuario.
      location = URLEncoder.encode(location, "utf-8");
}
String address = "address=" + location;
String sensor = "sensor=false";
String key = "key="+getResources().getString(R.string.google_maps_key);
URL = URL + address + "&" + sensor + "&" + key;
```

Envía una solicitud HTTP a la API de geocodificación. Utilice la URL construida como destino de la solicitud y establezca los métodos y parámetros de solicitud adecuados. Obtenga los datos de geocodificación devueltos de la respuesta de la API, que se devuelve en formato JSON, y analice la respuesta de la API.

Los datos de respuesta de la API de geocodificación se descargan en segundo plano ejecutando el método doInBackground(). Una vez descargada, la información de localización se analiza creando un objeto ParserTask y pasándole los datos de respuesta. Los datos de respuesta de la API de geocodificación se analizan en segundo plano y se devuelve una lista

de la información de ubicación analizada. En el método onPostExecute(), se borran los marcadores existentes en el mapa y se recorre la lista de ubicaciones, creando marcadores para cada ubicación y añadiéndolos al mapa.

```
for (int i = 0; i < list.size(); i++) {
    MarkerOptions markerOptions = new MarkerOptions();
    // Obtener la latitud del lugar
    double lat = Double.parseDouble(hmPlace.get("lat"));
    // Obtener la longitud del lugar
    double lng = Double.parseDouble(hmPlace.get("lng"));
    // Obtener nombre
    String name = hmPlace.get("formatted_address");
    LatLng latLng = new LatLng(lat, lng);
 ......
    // Fijar la posición del marcador
    markerOptions.position(latLng);
    // Establecer el título del marcador
    markerOptions.title(name);
}
```

4.3.3. Confirmation

4.3.3.1. Criterios de Aceptación (Given - When - Then)

Feature: Comprueba que puedes marcar en el mapa el lugar al que quieres ir.

Scenario: Introduce el lugar en el campo de texto y, al pulsar el botón, el lugar se marca en el mapa.

Given	El teléfono está conectado a la red
When	Se hacen clic en el icono del hotel
Then	El lugar está marcado en el mapa y puede señalarse por el nombre.

https://maps.googleapis.com/maps/api/geocode/json?address=museo+nacional+del+prado+&sensor=false&key=AIzaSyDzVzQr0RBpISeU3jbf7fHPOhd-1VwENWc

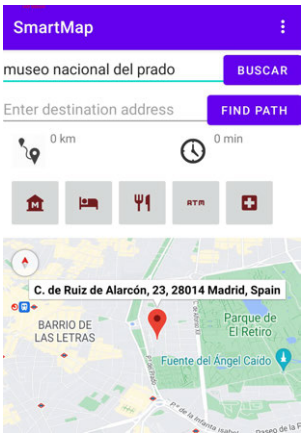

Figura 15 El resultado

4.4. Crear una ruta en el mapa

4.4.1. Card

Como usuario,
quiero ser capaz de crear una ruta en el mapa,
para poder marcar en línea en el mapa el trayecto entre los puntos de salida y llegada, y calcular la distancia entre las rutas y el tiempo empleado.

- **Título**: "Crear una ruta en el mapa"

- **Prioridad**:9 （10 es la máxima prioridad）

- **Esfuerzo de trabajo estimado (días ideales)**: 3

4.4.2. Conversation

4.4.2.1. Texto de la conversación

El usuario introduce el lugar de salida en el campo de texto, seguido del destino al que quiere ir en el campo de texto. Por último, pulsa el botón "Find Path". Las rutas de salida y llegada estarán marcadas en el mapa. Al mismo tiempo, también se indica la distancia de la ruta y el tiempo necesario para completarla.

4.4.2.2. Wireframe

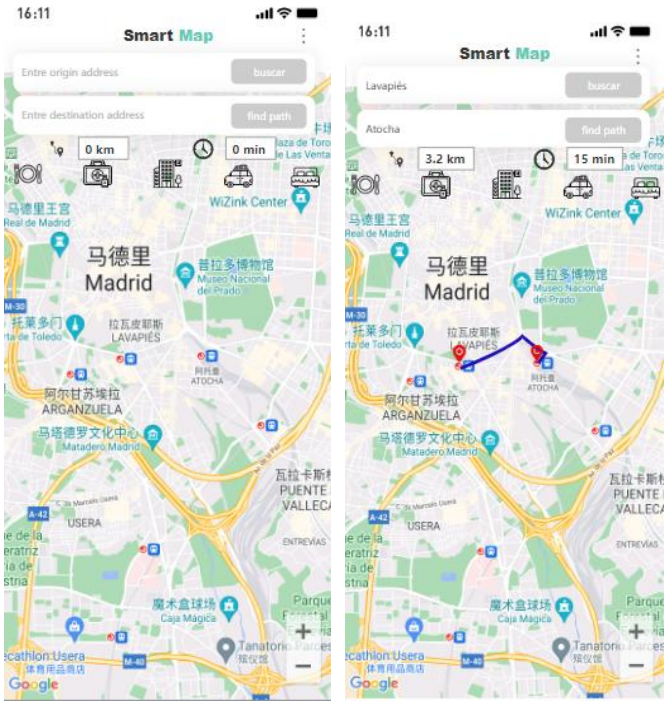

Figura 16 Wireframe

Inputs: Introduzca los puntos de salida y llegada en cada una de los dos campos de texto.
Gestures: Se tendrá que presionar sobre el campo de texto. Luego, el teclado se activará.
Motion: Si bien es responsive solo se diseña para orientación vertical.
Annotation: El campo de texto superior muestra el texto "Inicio". El inferior el texto "Fin", que tiene un botón “Find Path” a la derecha. Hay dos iconos, uno para la distancia y otro para el tiempo. Como está claro, no es necesario anotarlos.

Endpoints: Para crear una ruta en un mapa, es necesario realizar una solicitud al punto de salida "Google Maps Directions API".

4.4.2.3. Tareas

TAREA 1: Activa los servicios adecuados de la API de Google Maps, y añade las dependencias correctas y el mapa de la aplicación Android.

En la historia de usuario anterior, ya lo hemos hecho. Solo hace falta que la API Directions esté activada en Google Cloud Platform.

TAREA 2: Pulsa el campo de texto, y escribe letra a letra los nombres de salida y llegada. A continuación, pulsa el botón.
Cuando el usuario pulsa el botón, se llama al método sendRequest para recuperar el contenido del texto. Primero comprobará si el contenido de los dos campos de texto está vacío. Si está vacío, se recibirá un mensaje Toast.

```
private void sendRequest() {
    String origin = etOrigin.getText().toString();
    String destination = etDestination.getText().toString();
    if (origin.isEmpty()) {
```

```
            Toast.makeText(this, "Please enter origin address!", Toast.LENGTH_SHORT).show();
            return;
        }
        if (destination.isEmpty()) {
            Toast.makeText(this, "Please enter destination address!", Toast.LENGTH_SHORT).show();
            return;
        }
.....
}
```

TAREA 3: Si el campo de texto se introduce correctamente, la ruta entre los puntos de salida y llegada se muestra en el mapa. Y, los puntos de salida y llegadal están igualmente marcados.

Al igual que en la historia de usuario anterior, la implementación de la creación de una ruta también requiere que se envíe una solicitud a través de Http URL. La solicitud contendrá el punto de inicio, el punto final y la clave API.

Cómo compilar tus solicitudes a la API[5] de Directions:

"https://developers.google.com/maps/documentation/directions/get-directions?hl=es-419"

En el ejemplo anterior, outputFormat puede ser cualquiera de los siguientes valores:

-json (recomendado) indica el resultado en JavaScript Object Notation (JSON).

-xml indica el resultado como XML.

Required parameters:

-origin

-destination

En el método createURL, el método URLEncoder.encode() codifica las URL de origen y destino. Una vez que tenga la URL de la API, puede realizar una solicitud HTTP y obtener los datos de respuesta de la API.

```
private String createURL() throws UnsupportedEncodingException {
        String URLOrigin = URLEncoder.encode(origin, "utf-8");
        String URLDestination = URLEncoder.encode(destination, "utf-8");
        String URL = DIRECTION_URL_API + "origin=" + URLOrigin + "&destination=" + URLDestination +
"&key=" + GOOGLE_API_KEY;
.....
}
```

Una vez obtenidos los datos de la respuesta, el método onPostExecute() pasa los datos al método pareseJson() para leer la información necesaria. Los datos de respuesta "overview_polyline" son un campo de la respuesta de la API de Directions que contiene una visión general de alto nivel de la geometría de la ruta. Contiene un campo de cadena llamado "points". Este campo "points" contiene los puntos de coordenadas codificados que se utilizan para trazar la geometría general de toda la ruta. Los datos se analizan para crear una lista de objetos Route que contengan la información de la ruta y el resultado se pasa al oyente a través de un método de devolución de llamada.

```
List<Route> routes = new ArrayList<Route>();
```

```
JSONObject overview_polylineJson = jsonRoute.getJSONObject("overview_polyline");
route.points = decodePolyLine(overview_polylineJson.getString("points"));
.....
listener.onDirectionFinderSuccess(routes);
```

En este punto se ha obtenido una lista de objetos de Route que contienen información sobre la ruta. En el método onDirectionFinderSuccess, se recorre la lista de rutas y se realiza la operación para cada objeto Route.En primer lugar, se eliminan del mapa los marcadores y guiones anteriores. En segundo lugar, se crean marcadores para los puntos de salida y llegada. A continuación, se traza la ruta en el mapa.

```
@Override
public void onDirectionFinderSuccess(List<Modules.Route> routes) {
progressDialog.dismiss();//Borrar los marcadores y guiones anteriores del mapa
.....
      for (Route route : routes) {
.....
//crear marcadores para los puntos de salida y llegada
            originMarkers.add(mMap.addMarker(new MarkerOptions()
                        .icon(BitmapDescriptorFactory.fromResource(R.drawable.start_blue))
                        .title(route.startAddress)
                        .position(route.startLocation)));
            destinationMarkers.add(mMap.addMarker(new MarkerOptions()
                        .icon(BitmapDescriptorFactory.fromResource(R.drawable.end_green))
                        .title(route.endAddress)
                        .position(route.endLocation)));
//trazar la ruta
            PolylineOptions polylineOptions = new PolylineOptions().
                        geodesic(true).
                        color(Color.BLUE).
                        width(10);
            for (int i = 0; i < route.points.size(); i++)
                  polylineOptions.add(route.points.get(i));
      }
}
```

TAREA 4: Calcula el tiempo y la distancia necesarios para la ruta.

En la tarea anterior, de los datos de respuesta obtenidos se extrae información sobre la distancia y el tiempo. Por ejemplo, jsonLeg.getJSONObject("distancia") puede utilizarse para obtener un objeto JSON llamado "distancia", que contiene información relacionada con la distancia, como una representación textual de la distancia y una representación numérica.

La información de distancia y duración se almacena en las propiedades route.distance y route.duration creando objetos Distance y Duration y pasando los valores obtenidos de los objetos jsonDistance y jsonDuration a sus constructores.

```
JSONObject jsonDistance = jsonLeg.getJSONObject("distance");
JSONObject jsonDuration = jsonLeg.getJSONObject("duration");
```

```
route.distance = new Distance(jsonDistance.getString("text"), jsonDistance.getInt("value"));
route.duration = new Duration(jsonDuration.getString("text"), jsonDuration.getInt("value"));
```

Ahora puede utilizar route.distance y route.duration para obtener y mostrar información sobre la distancia y la duración de una ruta. La información sobre la duración y la distancia de la ruta obtenida se muestra en el control TextView correspondiente.

```
((TextView) findViewById(R.id.Duration)).setText(route.duration.text);
((TextView) findViewById(R.id.Distance)).setText(route.distance.text);
```

4.4.3. Confirmation

4.4.3.1. Criterios de Aceptación (Given - When - Then)

Feature: Comprobar si puedes crear líneas en el mapa

Scenario: Se Introducen los puntos inicial y final en el cuadro de texto y, a continuación, pulsa el botón.

Given	El teléfono está conectado a la red
When	Se Introducen los puntos inicial y final en el cuadro de texto y, a continuación, se pulsan el botón.
Then	Se marca la ruta entre el punto de salida y el de llegada en el mapa y se puede mostrar el tiempo y la distancia necesarios.

https://maps.googleapis.com/maps/api/directions/json?origin=puerta+del+sol+&destination=puerta+del+Alcal%C3%A1+%0A&key=YOU-KEY

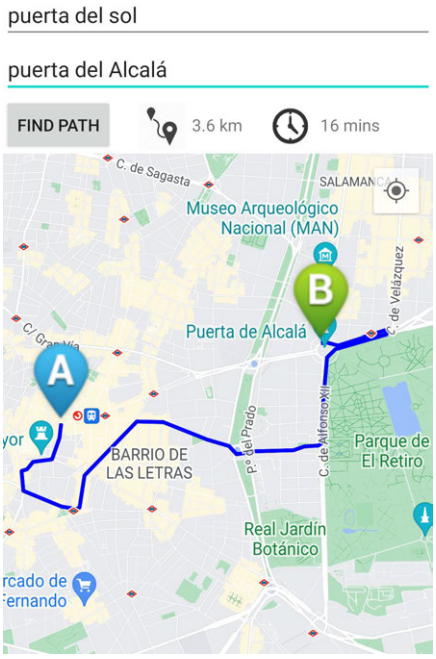

Figura 17 El resultado

4.5. Leer archivos KML

4.5.1. Card

Como usuario,
quiero ser capaz de leer archivos KML importados,
para poder mostrar la información del archivo KML en el mapa.

- **Título**: "leer archivos KML"

- **Prioridad**:6 （10 es la máxima prioridad）

- **Esfuerzo de trabajo estimado (días ideales)**: 2

4.5.2. Conversation

4.5.2.1. Texto de la conversación

El usuario hace clic en la barra de menús para seleccionar la información que desea ver y a continuación se muestra la información correspondiente en el mapa.

4.5.2.2. Wireframe

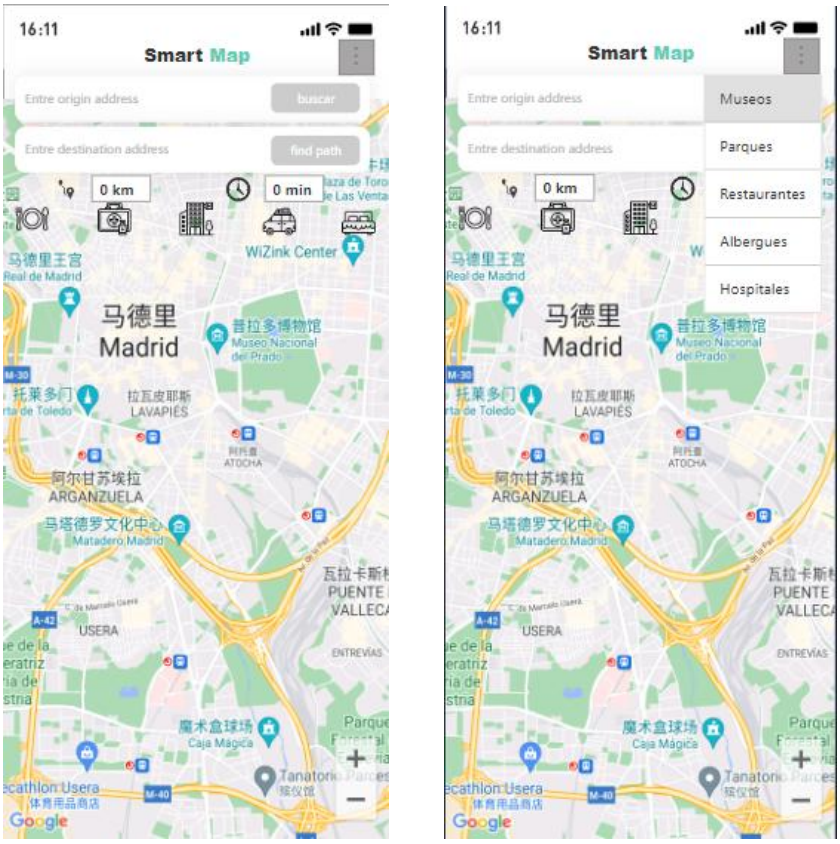

Figura 18 Wireframe

Inputs: En esta historia de usuario no se encuentra ningún input; ya que, la base es local.

Gestures: Deberá hacer clic en la barra de menú y seleccionar la información que desea ver.
Motion: Si bien es responsive solo se diseña para orientación vertical.
Annotation: El layout será responsive (se ajustará a todo tipo de dispositivos móviles). No se hará uso del teclado, razón por la cual el mismo se encontrará oculto.

4.5.2.3. Tareas

TAREA 1: Coloca el archivo KML en el directorio res del proyecto.

Fuente del archivo KML:
https://gestiona.comunidad.madrid/nomecalles/DescargaBDTCorte.icm

TAREA 2: Establece configuraciones relacionadas con Google Maps, como la habilitación de controles de mapas.

TAREA 3: Diseña el menú

Para crear el archivo de recursos del menú, crea un nuevo archivo XML en el directorio res/menu/ de su proyecto Android para definir la estructura y las propiedades de los elementos del menú. Maneja el evento de selección del elemento de menú y anula el método onOptionsItemSelected() para realizar la acción apropiada basada en el ID del elemento de menú.

TAREA 4: Muestra la información del archivo KML correspondiente en el mapa

Carga el archivo KML y lo muestra en el mapa llamando al método addKMLLayerToMap(). En el método, primero se crea un objeto KMLLayer y se le pasa el mMap, el ID de recurso del archivo KML y el contexto de aplicación getApplicationContext().A continuación, se llama al método addLayerToMap() para añadir el KMLLayer al mapa.

```
private KMLLayer addKMLLayerToMap(GoogleMap mMap) {
    KMLLayer layer = null;
.....
    try {
        layer = new KMLLayer(mMap, R.raw.museos, getApplicationContext());
        layer.addLayerToMap();
    }
    return layer;
}
```

Imprime la información del contenedor en el archivo KML, incluidos los ID de estilo y las propiedades de todos los objetos KMLPlacemark en el objeto KMLLayer.

El método printKMLContainers() acepta como argumento un objeto KMLLayer, que representa el archivo KML cargado. El método recorre internamente cada marcador para obtener todos los marcadores llamando al método getPlacemarks(). Dentro del bucle, el ID de estilo del marcador puede obtenerse utilizando el método getStyleId() y las propiedades del marcador pueden obtenerse utilizando el método getProperties().

```
private void printKMLContainers(KMLLayer layer, Iterable<KMLPlacemark> placemark_new) {
            for (KMLPlacemark placemark : placemark_new) {
              String styleId = placemark.getStyleId();
            KMLProperties properties = placemark.getProperties();
            ……}
}
```

4.5.3. Confirmation

4.5.3.1. Criterios de Aceptación (Given - When - Then)

Feature: Comprueba que todos los museos de Madrid pueden visualizarse en el mapa

Scenario: Cuando el usuario hace clic en el ítem del museo en el menú.

Given	El teléfono no está conectado a la red. Y el archivo KML correspondiente se ha importado en la aplicación.
When	Se hace clic en el ítem del museo
Then	Se marcan todos los museos de Madrid en el mapa.

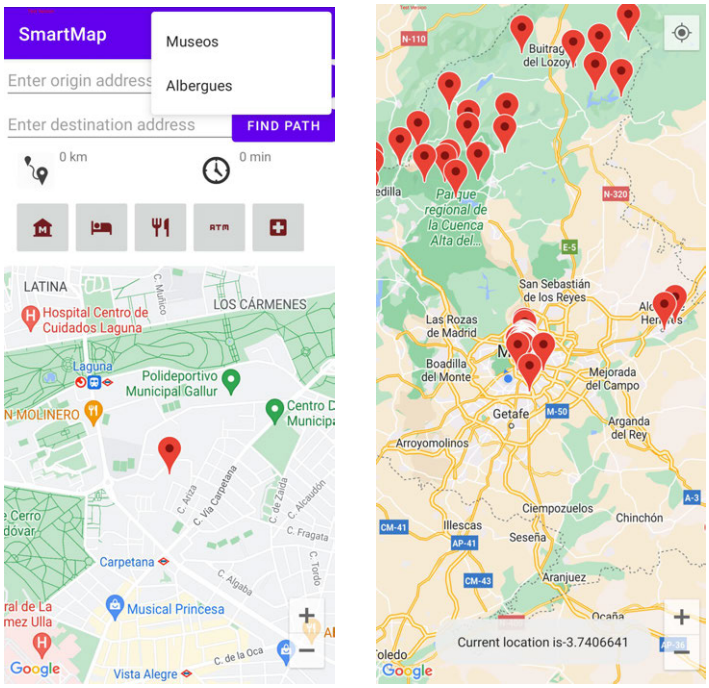

Figura 19 El resultado

5. Conclusiones y trabajos futuros

Tras finalizar el proyecto, ha llegado el momento de analizar qué se ha aprendido. Dividiré las conclusiones en tres grupos: análisis de la planificación, conclusiones técnicas, por último, oportunidades de mejora u orientaciones para futuros trabajos.

- **Planificación**: La planificación inicial se vio afectada por una serie de desviaciones, en primer lugar, cambió el alcance del proyecto ya que la idea original de encontrar puntos de interés cercanos leyendo y analizando archivos KML importados lo intentó y fracasó durante mucho tiempo. Al final se pensó en una solución y se resolvió el problema construyendo la URL correspondiente. Combinado con los conocimientos aprendidos en el curso de Geoinformática, el proyecto se completó finalmente.
- **Tecnología**: Construye la URL correspondiente y llama al punto final de servicio de la API de Google Maps para implementar varias funciones a través de peticiones URL HTTP. Este es un punto interesante. Google Maps API[6] proporciona una amplia gama de funciones y servicios que permiten implementar muchas características. He utilizado algunos de ellos en el proyecto y hay mucho más que explorar y aprender.
- **Oportunidades de mejora：** En la función de creación de rutas, considere la posibilidad de añadir más opciones de navegación (por ejemplo, navegación en transporte público, navegación a pie), aumentar la precisión o la velocidad de posicionamiento, etc. También puede explorarse la integración con otras aplicaciones y servicios relacionados para proporcionar funcionalidad y comodidad adicionales. Por ejemplo, integración con plataformas de medios sociales, conectividad con dispositivos inteligentes (p. ej., relojes inteligentes, hogares inteligentes), etc. para permitir una mayor interacción e interoperabilidad. Una cosa más, haz que la aplicación genere ingresos. Añadir anuncios a la aplicación permite a los usuarios obtener ingresos de la plataforma publicitaria cuando ven los anuncios o interactúan con ellos.

6. Bibliografía

[1] Recorrido por las 10 dinámicas de Agile Inception, Online, https://adrianalonso.es/project-management/recorrido-10-dinamicas-de-agile-inception/
[2] APLICACIÓN DE LA METODOLOGÍA MOBILE-D EN EL DESARROLLO DE UNA APP MÓVIL PARA GESTIONAR CITAS MÉDICAS DEL CENTRO JEL RIOBAMBA, Tesis - Ingeniería en Sistemas y Computación, http://dspace.unach.edu.ec/handle/51000/7073#:~:text=Mobile%2DD%20es%20una%20metodolog%C3%ADa,proyecto%2C%20permitiendo%20la%20reducci%C3%B3n%20de
[3] Búsqueda en los alrededores, Online, https://developers.google.com/maps/documentation/places/web-service/search-nearby?hl=es-419

[4] Geocoding API, Online, https://developers.google.com/maps/documentation/geocoding/start?hl=es-419
[5] Obtener instrucciones sobre cómo llegar a través de la API de Directions, Online, https://developers.google.com/maps/documentation/directions/get-directions?hl=es-419
[6] Las API de Google Maps Platform por plataforma, Online, https://developers.google.com/maps/apis-by-platform?hl=es-419