DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL ANDROID PARA LA NAVEGACIÓN Y APROVECHAMIENTO DE LA RED GEODÉSICA DEL IGAC Y CONVERSIÓN DE COORDENADAS DENTRO DEL SISTEMA MAGNA SIRGAS

RAFAEL FABIAN SANCHEZ OSORIO
20171732003
LAYDA YANETH GIL ÁNGEL
20171732002

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE MEDIO AMBIENTE Y RECURSOS NATURALES
INGENERÍA TOPOGRÁFICA
BOGOTÁ D.C
DICIEMBRE 2020

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL ANDROID PARA LA NAVEGACIÓN Y APROVECHAMIENTO DE LA RED GEODÉSICA DEL IGAC Y CONVERSIÓN DE COORDENADAS DENTRO DEL SISTEMA MAGNA SIRGAS

RAFAEL FABIAN SANCHEZ OSORIO
LAYDA YANETH GIL ÁNGEL

Trabajo de grado presentando como requisito para optar por el título de:
Ingeniero Topográfico

Director
MSc. Ing. EVELIO LUIS MADERA ARTEAGA

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE MEDIO AMBIENTE Y RECURSOS NATURALES
INGENIERÍA TOPOGRÁFICA
BOGOTÁ D.C
DICIEMBRE 2020

Índice general

1. INTRODUCCIÓN 9
2. DESCRIPCIÓN DEL PROYECTO 10
   2.1 Planteamiento/ Identificación del problema 10
   2.2 Objetivos 10
      2.2.1 Objetivo General 10
      2.2.2 Objetivos específicos 10
3. MARCO CONCEPTUAL 11
   3.1 Marco Teórico 11
      3.1.1 Android 11
      3.1.2 SDK 13
      3.1.3 Visual Studio Code 13
      3.1.4 Framework 14
      3.1.5 Desarrollo Nativo e Híbrido 14
      3.1.6 Almacenamiento de Datos 15
      3.1.7 SQLite 16
      3.1.8 Flutter 16
      3.1.9 Dart 17
      3.1.10 UI y UX 19
      3.1.11 UI (User Interface) 19
      3.1.12 UX (User Experience) 19
      3.1.13 Observaciones GNSS 20
      3.1.14 Tiempos de Rastreo 20
      3.1.15 Estaciones de Rastreo Continuo 21
      3.1.16 Tiempos de rastreo (estático diferencial) 21
      3.1.17 Red pasiva control horizontal y alturas niveladas 22
      3.1.18 GeoRED 24
      3.1.19 Coordenadas elipsoidales 25
      3.1.20 Coordenadas geocéntricas 27
      3.1.21 Transversal de Mercator 28
      3.1.22 Coordenadas Gauss Krüger 29

3.1.23 Coordenadas planas cartesianas 30
3.1.24 Conversión de Coordenadas 31
4. DESARROLLO DEL PROYECTO 39
4.1 Etapa I Requerimientos 41
4.2 Etapa II Análisis 45
4.2.1 Metodología de Desarrollo 45
4.2.2 Scrum 46
4.2.3 Framework de Desarrollo 47
4.2.4 Entorno de Desarrollo (IDE) 47
4.3 ETAPA III 48
4.3.1 Diseño de UI 48
4.3.2 Diseño Base De Datos Relacional 49
4.3.3 Modelo De Datos 50
4.3.4 Librerías Utilizadas 55
4. ETAPA IV 57
4.2 Diagrama de Clases 57
4.1.1 framework.dart 59
4.1.2 main.dart 60
4.1.3 acercaDe.dart 60
4.1.4 menuPrincipal.dart 61
4.1.5 database.dart 61
4.1.6 tiemposRastreo.dart 64
4.1.7 redPasivaIGAC.dart 65
4.1.8 conversionPuntoIndividual.dart 66
4.1.9 conversionArchivo.dart 68
4.1.10 Clases de uso genérico 70
4.2 Codificación 74
4.2.1 Implementación módulo de Navegación Red Pasiva IGAC 75
4.2.2 Implementación módulo para el cálculo de Tiempos de Rastreo 82
4.2.3 Conversión de Coordenadas 87
5. CONCLUSIONES 102
5.1 Verificación, contraste y evaluación de los objetivos 102
5.2 Síntesis del modelo Propuesto 103
6. BIBLIOGRAFÍA 104

Índice de Figuras

Ilustración 1 Uso y justificación de async – await 18
Ilustración 2 Método Estático Diferencial 22
Ilustración 3 Red Pasiva Magna Sirgas 24
Ilustración 4 Coordenadas Geográficas 26
Ilustración 5 Coordenadas Geocéntricas 28
Ilustración 6 Proyección Transversal de Mercator 29
Ilustración 7 Proyección cartográfica Gauss Krüger 30
Ilustración 8 Obtención de coordenadas planas cartesianas 31
Ilustración 9 Coordenadas rectangulares geocéntricas y elipsoidales 32
Ilustración 10 Coordenadas elipsoidales y Planas de Gauss 33
Ilustración 11 Coordenadas elipsoidales y planas cartesianas 36
Ilustración 12 Plan de trabajo desarrollo App 40
Ilustración 13 Metodología scrum 46
Ilustración 14 Entorno de Desarrollo Visual Studio Code 48
Ilustración 15 Navegación Red Pasiva del IGAC 49
Ilustración 16 Modelo de Datos Relacional Utilizado 50
Ilustración 17 Diagrama de Clases Aplicación 57
Ilustración 18 Modelo Clase FramerWork.dart 59
Ilustración 19 Modelo de Clase main.dart 60
Ilustración 20 Modelo de Clase acercaDe.dart 60

Ilustración 21 Modelo de Clase menuPrincipal.dart 61
Ilustración 22 Modelo de Clase database.dart 62
Ilustración 23 Modelo de Datos Clase Tiempos de Rastreo 64
Ilustración 24 Modelo de clase redPasivaIGAC.dart 65
Ilustración 25 Modelo de Datos conversionPuntoIndividual.dart 67
Ilustración 26 Modelo de ClaseconversionArchivo.dart 69
Ilustración 27 Modelo de clase AltNiveladas.dart 71
Ilustración 28 Modelo de Clase coordenadasElipsoidales.dart 71
Ilustración 29 Modelo de clase CoordenadasGeocentricas.dart 72
Ilustración 30 modelo de Clase coordenadasON.dart 72
Ilustración 31 modelo de Clase CPlanasGenerico.dart 73
Ilustración 32 modelo de Clase origenesCartesianos.dart 73
Ilustración 33 modelo de clase origenesGauss.dart 74
Ilustración 34 modelo de clase RedPIGACPuntos.dart 74
Ilustración 35 Muestra de Datos red Pasiva del IGAC 76
Ilustración 36 Información Punto de Referencia Red Pasiva IGAC 78
Ilustración 37 Mapa Base Navegación 79
Ilustración 38 clúster de Puntos Red Pasiva IGAC 80
Ilustración 39 Ubicación Usuario Navegación Red Pasiva 81
Ilustración 40 Navegación Red Pasiva del IGAC 82
Ilustración 41 Selección de Red y Norma de Referencia Cálculo de Tiempos de Rastreo ... 83
Ilustración 42 Establecimiento de Ubicación para cálculo de Rastreo 84
Ilustración 43 Cálculo de Tiempos de Rastreo 86

Ilustración 44 Compartir resultado cálculo tiempo de Rastreo 87

Ilustración 45 Conversión de Coordenadas 97

Ilustración 46 Interfaz conversión de Coordenadas Elipsoidales y Geocéntricas Punto Individual 98

Ilustración 47 Selección de Origen de Plano Cartesiano 99

Ilustración 48 Selección de Origen Proyección Gauss Krüger 100

Ilustración 49 Configuración de Archivo Importado 101

Índice de tablas

Tabla 1 Historia de Usuario - Navegación red pasiva y altura nivelada 42

Tabla 2 Historia de usuario - Conversión de coordenadas 43

Tabla 3 Historia de usuario - Cálculo de tiempo de rastreo red activa IGAC y GeoRed 44

Tabla 4 Modelo de Datos Entidad Origen Plano Cartesiano 51

Tabla 5 Modelo de Datos Entidad Origen Gauss 52

Tabla 6 Modelo de Datos Entidad Departamento 52

Tabla 7 Modelo de Datos Entidad Municipios 53

Tabla 8 Modelo de Datos Entidad Altura Niveladas 53

Tabla 9 Modelo de Datos Entidad Estaciones GeoRed 54

Tabla 10 Modelo de datos Entidad Estaciones Magna Eco 54

Tabla 11 Modelo de Datos Red Pasiva IGAC 55

1. INTRODUCCIÓN

El desarrollo tecnológico ha traído consigo la masificación y el acceso a tecnología que permite acceder a servicios y sensores mediante el uso de hardware móvil. Estas facilidades han permitido que hoy en día sea normal contar con teléfonos inteligentes, los cuales normalmente tienen sensores como: receptores GNSS, giroscopios, acelerómetros, etc. Este tipo de sensores permiten el desarrollo de software que, en conjunto con elementos del hardware básico de cualquier dispositivo inteligente, posibilitan el desarrollo de aplicaciones para usuarios específicos. En este sentido, el desarrollo de aplicaciones para el aprovechamiento de información de las labores propias de la topografía se convierte en una posibilidad para facilitar y optimizar trabajos específicos. La navegación de los puntos de la red pasiva del IGAC y el cálculo de tiempo de rastreo es un trabajo base que se realiza para todos los proyectos de ingeniería que deban cumplir con especificaciones técnicas establecidas por el ente encargado de producir la cartografía básica del país.

En el caso de Colombia, es el Instituto geográfico Agustín Codazzi. Dentro de sus funciones se encuentra el establecimiento y densificación de la red geodésica actual, y suministro de información de referencia dentro del sistema de coordenadas usado en Colombia, función que es desarrollada mediante la materialización de vértices geodésicos en el territorio nacional y la implementación y puesta en marcha de una red de rastreo continua. Al tener integrado el módulo de recepción de información GNSS dentro de los dispositivos móviles, se abre la posibilidad de mejorar el aprovechamiento de la información geográfica del IGAC.

Se plantea el desarrollo de una aplicación para los dispositivos que usen el sistema operativo Android que; facilite la navegación de los puntos pertenecientes a la red Pasiva, determine el tiempo de rastreo para observaciones GNSS realizadas en metodología estático diferencial y realice la conversión de coordenadas dentro del datum MAGNA a las diferentes proyecciones y orígenes usados en Colombia de forma oficial.

2. DESCRIPCIÓN DEL PROYECTO

2.1 Planteamiento/ Identificación del problema

Los trabajos de geomática que involucran un posicionamiento GNSS y la navegación de los puntos de la red pasiva implican identificar la ubicación geográfica de cada punto materializado para obtener sus coordenadas MAGNA SIRGAS y su respectiva cota geométrica. En el caso de la red activa, implica conocer la distancia de la estación más cercana al punto del levantamiento topográfico. Un problema recurrente es la falta de señal en zonas rurales para consultar la información de los datos abiertos del IGAC. Así mismo, la consulta del tiempo de rastreo según diferentes normas de las empresas institucionales puede ser compleja si no hubo planeación anticipada de los trabajos en campo. Es importante tener alternativas locales para estas consultas. Por esta razón, se plantea el desarrollo de una aplicación bajo el sistema Android que permita tener todos los datos anteriormente mencionados localmente en dispositivos inteligentes para no depender de cobertura telefónica en apartados rurales.

2.2 Objetivos

2.2.1 Objetivo General

Diseñar e implementar una aplicación móvil que permita la navegación y cálculo de tiempos de rastreo usando la red de referencia geodésica del IGAC y la conversión de coordenadas dentro del datum MAGNA SIRGAS.

2.2.2 Objetivos específicos

- Implementar un módulo para la navegación de puntos pertenecientes a la red Pasiva del IGAC.
- Diseñar un módulo dentro de la aplicación para determinar los tiempos de rastreo mediante la observación GNSS en estático diferencial usando como puntos de referencia las estaciones de rastreo continuo del IGAC.
- Desarrollar un módulo que permita la conversión de coordenadas dentro del datum MAGNA SIRGAS entre las proyecciones planas Cartesianas, Planas Gauss Krüger, Elipsoidales, Geocéntricas y el sistema plano CTM12.

3. MARCO CONCEPTUAL

3.1 Marco Teórico

3.1.1 Android

Es un sistema operativo de código abierto para teléfonos móviles, fue desarrollado por Android Inc. en el año 2003 por Andy Rubin, Rich Miner, Nick Sears y Chris White con la idea principal de realizar software para teléfonos móviles construida sobre la versión 2.6 del kernel de Linux, fue comprada por la compañía Google en agosto de 2007 y actualmente por el Open Handset Alliance. (Escola Tècnica Superior d'Enginyeria Informàtica, 2012)

Su existencia se hizo pública el 5 de noviembre de 2007 por la fundación Open Handset Alliance, asociación de 47 empresas de hardware, software y telecomunicaciones dedicadas a la composición y estandarización de dispositivos móviles. Google ha sido el encargo de publicar gran cantidad de código fuente de Android bajo la licencia de software Apache, con la característica principal de ser software libre y de código abierto, lo cual permite que esté disponible para cualquier desarrollador. (Universidad Carlos III de Madrid, s.f.)

El sistema operativo de Google ha evolucionado, por lo cual se han presentado diferentes versiones en el tiempo. Existe una particularidad en la asignación de nombre a las versiones y es que tienen nombre de postre continuando el orden alfabético. La primera versión de Android fue conocida como “Apple Pie” para el inicio de pruebas, la segunda versión fue llamada “Banana Bread” para corregir errores de la versión 1.0 y no fueron usadas comercialmente. La versión 1.5 fue llamada: “Cupcake”, fue la primera versión en usar un nombre de postre. La versión “Donut” llegó el soporte para redes CDMA, lo cual permitió que Android llegara a Estados Unidos y Asia. Una de las mejoras más significativas fue la posibilidad de correr el sistema operativo en múltiples resoluciones de pantalla, lo cual permitió que se pudieran crear pantallas con resolución QVGA, HVGA, WVGA, FWVGA, QHD y 720p. En el año 2009 fue ofrecido en exclusiva un teléfono de la marca Motorola que permitió impulsar las mejoras para la versión 2.0 “Eclair”, un aspecto importante es que Google Maps Navigation fue publicado, permitiendo integrar un sistema de navegación de

automóviles con guía de voz e información del tráfico de una manera gratuita. La versión "Froyo" (2010) fue basada en el kernel de Linux 2.6.32 añadiendo funcionalidades como Wi-Fi hotspot y tethering por USB y la desactivación del tráfico de datos a través de la red del operador. "Gingerbread" (2010) integra funcionalidades de cortar, copiar y pegar a lo largo del sistema, mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos y administrador de descargas. "Honeycomb" (2011) nos muestra la posibilidad de navegar por pestañas, auto rellenar formularios y navegar de manera privada. "Ice Cream Sandwich" (2011) unifica el uso en cualquier dispositivo, tanto en teléfonos, tablets, televisores y netbooks. Implementa las capturas de pantalla, con solo pulsando el botón de bajar volumen y el botón de encendido e implementa una nueva fuente llamada "Roboto". "Jelly Bean" (2012) posee Android Beam mejorado con posibilidad de transmitir vídeo por NFC, Google Chrome se convierte en el navegador por defecto de Android y se implementa Gestual Mode para las personas discapacitadas visualmente. (Basterra, 2016)

La versión 4.4 "KitKat" (2013), llegaron las transparencias: en el panel de notificaciones, por lo cual se mezcla con el fondo de pantalla, menú para cambiar el fondo de pantalla y widgets en la pantalla de inicio. "Lolipop" (2014), permitió que se visualizaran las notificaciones en la pantalla de bloqueo, además de implementar seguridad con la versión antirrobo y restablecimiento de fábrica con la cuenta de Google. "Marshmallow" (2015), un punto muy importante de esta versión es que llega el soporte nativo para el lector de huella además de la posibilidad de abrir la cámara con un doble toque en el botón de encendido, "Nougat" (2016), permite que aplicaciones de terceros añadan botones, así mismo llegan los emojis con distintos tonos de piel y la posibilidad de elegir otro idioma en el sistema operativo. "Oreo" (2017), llega la posibilidad de silenciar notificaciones y el autocompletado de formularios. "Pie" (2018), permite el uso del brillo y la batería inteligente y la navegación por gestos, lo cual hizo que se redujera la barra de accesos a un solo botón. (Ramirez, 2019)

La versión más reciente: "Android 10" (2019), permite dar respuestas inteligentes a notificaciones o mensajes. Así mismo, cuando se reciba un correo electrónico con un enlace o un vídeo de YouTube se podrá ver directamente desde la aplicación en el dispositivo

móvil, se perfecciona el modo oscuro en un mayor número de aplicaciones y se agrega la posibilidad de otorgar o quitar acceso de ubicación a aplicaciones del dispositivo. (NTS, 2019)

La versión de Android 11 (2020) se centra en las mejoras de conversaciones a través del uso de un apartado exclusivo en la barra de notificaciones, así mismo se desarrolló la posibilidad de manejar chats a través de burbujas. También, desarrolló un apartado dedicado al internet de las cosas y control de dispositivos inteligentes disponibles en el menú de encendido. (Android Developers, 2020)

3.1.2 SDK

El kit de desarrollo de Software, llamado SDK por sus siglas en inglés (Software Development Kit) es un paquete de herramientas de datos que permite realizar desarrollos en diferentes lenguajes de programación. Está compuesto por un API a través del cual los proyectos de software se pueden enlazar con un código fuente. La arquitectura de la interfaz de programación suele estar acompañado de documentación con instrucciones de uso, compiladores, bibliotecas, entornos de desarrollo, debbugers, protocolos y proyectos de prueba. (IONOS ES, 2019)

Android Studio es el entorno de desarrollo oficial, integrado y publicado por Android para el desarrollo de Apps. Está disponible para Windows, macOS y Linux. Fue constituido para el desarrollo y depuración para Android, está compuesto por un emulador, herramientas de compilación, complemento Grade para Android y herramientas de plataforma SDK. (Android, 2019)

3.1.3 Visual Studio Code

Es un ligero y poderoso editor de código fuente que está disponible para Windows, macOS y Linux. Tiene un soporte incorporado para Javascript, TypeScript y Node.js además de poseer gran cantidad de extensiones para otros lenguajes de programación como C++, Java, Python, PHP y tiempos de ejecución como .NET y Unity. Existen algunos complementos de

uso común como GIT con su respectivo soporte incorporado para el código fuente, sin embargo, requiere que se instale por separado. La biblioteca de lenguajes de programación hace que existan diferentes opciones para elegir al momento de elegir un idioma. La riqueza de estos se puede aprovechar en el uso de Visual Studio Code con soporte para el resaltado de sintaxis, terminaciones inteligentes, linting, código de navegación y depuración. (Microsoft, 2020)

3.1.4 Framework

Es un marco de trabajo, como su traducción literal lo índica “Marco de referencia”, se utiliza para estructurar y normalizar la información. Se desarrolla como es esquema o paso a paso que se debe seguir para el correcto desarrollo e implementación de una aplicación. Existen diferentes tipos de frameworks según el lenguaje de programación, el tipo de aplicación a desarrollar y especificaciones como la base de datos y el sistema operativo. (Sanchez, 2006)

El objetivo del uso de un Framework es acelerar el proceso de programación, reusar código existente y promover el uso adecuado de buenas prácticas de desarrollo y el uso de patrones. Esto puede permitir usar nuevamente código en el futuro e identifica y corrige bugs o errores de programación. (Gutiérrez. , 2011)

3.1.5 Desarrollo Nativo e Híbrido

El desarrollo de aplicaciones nativas se refiere al desarrollo de una aplicación para ejecutarse específicamente en una plataforma específica y se tiene en cuenta el dispositivo, el sistema operativo y su versión. Las aplicaciones nativas son desarrolladas en un entorno de desarrollo integrado (IDE) proporcionando todas las herramientas para su construcción y depuración. La principal ventaja de dichas aplicaciones es que interactúa con las herramientas que posee el dispositivo como la cámara, GPS o el acelerómetro. Se podría considerar una desventaja que es necesario desarrollar una aplicación para cada plataforma disponible, lo cual puede generar mayores costos de desarrollo y actualización. (Nahuel Delía, 2017)

Las aplicaciones nativas se desarrollan en Objective C o Swift en IOS y Java o Kotlin en Android. Disponer de Swift y Kotlin permite realizar una programación funcional y robusta que proporciona que sean lenguajes tipados con null – safety. (Batanero & Sesma, 2017)

Las aplicaciones híbridas utilizan tecnologías como HTML, CSS Y JavaScript a través de Frameworks como Ionic, PhoneGap, etc. Es importante tener en cuenta que se desarrollan una sola vez; lo que permite que se pueda compilar para varios sistemas operativos, es lo que se conoce como "desarrollo multiplataforma". (NextU, 2018)

El desarrollo multiplataforma pretende optimizar la relación costo/beneficio con la característica de compartir la codificación entre diferentes plataformas de desarrollo. Entre las ventajas más importantes, se pueden recalcar: menor tiempo y costo de desarrollo; utilidades similares a las de desarrollo nativo con acceso al hardware del dispositivo y entornos de desarrollo como Visual Studio o en su lugar, la utilización de tecnologías HTML5, Javascript y CSS. Las aplicaciones multiplataforma pueden clasificarse en aplicaciones web móviles, híbridas, interpretadas y generadas por compilación cruzada. (Nahuel Delía, 2017)

3.1.6 Almacenamiento de Datos

Los datos tienen un papel fundamental dentro del funcionamiento de cualquier software, el procesamiento y su representación dentro de cualquier aplicación. Se han desarrollado diferentes métodos de almacenamiento de información en medio digital; bases de datos relacionales y bases de datos no relacionales. Las bases de datos relacionales normalmente son usadas para el almacenamiento de grandes cantidades de datos y su uso e implementación normalmente requieren del uso hardware robusto. Dentro de este tipo de bases de datos se encuentra; SQLite utilizada para el almacenamiento de forma local usando el modelo relacional, teniendo dentro de sus principales características, el poco uso de recursos de hardware para su funcionamiento.

3.1.7 SQLite

Es una base de datos de código abierto que funciona bajo el modelo relacional, se caracteriza por ser la base de datos usada por excelencia para el almacenamiento de datos de forma local dentro de dispositivos que no cuentan con características robustas de hardware como es el caso de celulares. El almacenamiento y consulta de información se realiza bajo el lenguaje de consulta estructurado (SQL). Resaltando dentro de sus bondades librerías para su uso en los lenguajes de programación utilizado con mayor frecuencia en actualidad, como es caso de Dart. (SG Buzz Mexico, 2020)

3.1.8 Flutter

Es el kit de herramientas de UI de Google de código abierto para la realización de aplicaciones que se pueden compilar, para móviles, web y escritorio a partir de una misma base de código. Los widgets de Flutter, incorporan las diferencias que se pueden presentar entre plataformas como scrolling, navegación, íconos y fuentes para garantizar rendimiento nativo en Android y en iOS. (Google, 2020)

El framework de Flutter ofrece diferentes características a los desarrolladores, entre las que se puede encontrar; uso de lenguaje de programación “Dart”, Interfaces de usuario preconstruidas con Material Design y Cupertino, amplia biblioteca de widgets, interfaz de usuario multiplataforma y alto rendimiento nativo. (Tutorials Point, 2020)

El primer equipo de Flutter, evaluó más de una docena de idiomas de programación, eligiendo “Dart” debido a que fue el lenguaje que más se ajustaba a la forma de construcción de las interfaces de usuario. Las características más importantes que se destacan de Dart son:

1. Puede ser compilado JIT (Just in Time), para ciclos de desarrollo rápidos y flujo de trabajo incluyendo el Stateful Hot Reload (Recarga en caliente con manejo de estado)

2. Facilita la creación de animaciones que se ejecutan a 60 fps con las herramientas de object allocation y garbage collection.
3. Es un lenguaje que permite usar características que son familiares para los usuarios que tienen conocimiento en lenguajes dinámicos y estáticos.

A lo largo de la historia, los lenguajes de programación han sido subdivididos en dos grandes grupos; lenguajes dinámicos y estáticos. Los compiladores estáticos se ejecutaban durante la creación de un programa "AOT" por sus siglas en inglés (Ahead of time) o Antes del tiempo de ejecución) en código de máquina nativo debido a que dichos lenguajes necesitaban saber el tipo de dato; en los lenguajes dinámicos el tipo no se fija de antemano. Por lo cual, los lenguajes dinámicos suelen ser ejecutados por JIT. Dart, puede tener lo más significativo de los dos lenguajes, ya que puede ser compilado en AOT y e JIT. Esto puede permitir que la compilación, el tiempo de ejecución y la puesta en marcha sean rápidos. (Leler, 2018)

3.1.9 Dart

Dart es un lenguaje de programación de código abierto, basado en clases creado por Google. Está especializado a la creación de entornos de usuario, puede realizar cambios de forma interactiva para ver los resultados de compilación al instante y compila código de máquina ARMX64 para dispositivos móviles, computadoras de escritorio y backend. Está basado en una programación orientada a objetos basada en clases, de herencia única. (Dart Dev, 2020)

Existen herramientas que complementan el uso de Dart como Chromium, el cual es el directorio donde está almacenada una versión especial del navegador web de Chrome llamada Dartium. Esta herramienta permite la compilación de aplicaciones nativas de Dart en navegadores web, esto se realiza a través de una máquina virtual llamada Dart VM. La segunda herramienta es llamada Dart-SDK es el kit de desarrollo de software de Dart e incluye diferentes librerías y herramientas de línea de comandos como el compilador Dart to JavaScript. (Dart Doc, 2019)

Dart multiplataforma se encuentra gracias a su framework, Flutter. Lo que significa que permite programar en distintas plataformas como Android, IOS, macOS, Linux, Mac y asistentes virtuales. (Platzi, 2019)

Es importante tener en cuenta que a comparación de otros frameworks como puede ser Ionic, en Flutter el código se compila a código nativo, por lo cual alcanza un rendimiento más avanzado. Existe otro punto de vista importante y es que, por ejemplo, a diferencia de React Native, Flutter tiene sus propios componentes nativos, los cuales los encontramos en el esquema de programación como widgets, lo cual permite que la aplicación guarde las características del fronted en cualquier dispositivo, independientemente del sistema operativo o la versión de este. (Universidad de Barcelona, 2020)

Para entender la particularidad de asincronismo de dart, es importante recordar que existen lenguajes de programación que tienen “async” y “await” en su sintaxis, lo cual permitía que si se usaba en una función el tipo de retorno iba a cambiar y se iba a generar una pausa:



Ilustración 1 Uso y justificación de async – await

Fuente: (Universidad de Barcelona, 2020)

Lo cual nos explica que se está utilizando un objeto ProcessedData pero antes de que el createData() sea finalizado, tendrá que esperar dos futures. Lo cual significa que, el código se empezará a ejecutar, procederá a detenerse esperando un evento en disco, después

continuará, y por último se detendrá nuevamente y esperará un evento en red. Es allí cuando se finalice el procedimiento cuando se podrá obtener un valor. Es importante mencionar que, esperando el evento del futuro, es cuando el código permite el almacenamiento en una “caja vacía” mientras que recibe el disco en red o la variable, lo cual hará que exista el retorno y guarde en esa caja vacía el valor que se estuvo esperando. (Brogdon, 2019)

3.1.10 UI y UX

Los términos UI (User interface) y UX (User experience) hacen referencia al desarrollo que tiene una aplicación dentro de la parte visual (Lo que es visible al usuario), el desarrollo de estos dos componentes dentro de las aplicaciones ha tomado relevancia en el desarrollo de software durante la última década, siendo en la actualidad tan relevante como el propio backend del software, ya que finalmente estos dos elementos son los que interactúan con el usuario.

3.1.11 UI (User Interface)

UI hace referencia al desarrollo de la interfaz del usuario, los colores utilizados, iconos, fuentes y demás elementos que caracterizan visualmente a una aplicación, es fundamental en el desarrollo del software ya que la UI permite al usuario interactuar con el software, se busca principalmente que el uso del software sea lo más intuitivo posible, dentro de los principales componentes de la UI se tienes los componentes de navegación, contenedores de información, inputs, contenedores, etc. (InterWare, 2020)

3.1.12 UX (User Experience)

La UX se enfoca en la experiencia que genera el software al ser utilizado por el usuario final, esta experiencia cubre aspectos desde la rapidez con la que cambia de pantallas, la forma como se muestra la información, el retorno de alguna acción como por ejemplo el efecto de hundir un botón al ser tocado, etc. Cabe resaltar que UI forma parte de la UX y que no se puede desarrollar una buena UX sin tener una buena UI.

3.1.13 Observaciones GNSS

Las observaciones o métodos de posicionamiento GNSS permiten la obtención de coordenadas a través de diferentes metodologías según las características y requerimientos de la información que se desea obtener, dentro de los métodos de observación GNSS existen dos grandes grupos: estáticos y cinemáticos.

Su funcionamiento está basado en el almacenamiento de información suministrada por constelaciones de satélites que orbitan alrededor de la tierra, enviando de forma continua información de la posición (X,Y,Z) junto con la hora en la que se envía la señal. Esta información es analizada y procesada para determinar las coordenadas de un punto observado a través de la obtención de distancias entre el receptor GNSS y los satélites que suministran la información. Esta distancia es calculada a partir del tiempo que tarda la señal en llegar desde el satélite hasta el receptor, finalmente al obtener la distancia para un momento dado entre varios satélites y el receptor se logra obtener las coordenadas del punto observado. (Universidad Politecnica de Valencia, 2020)

3.1.14 Tiempos de Rastreo

Dentro del proceso de obtener coordenadas de un punto de forma precisa mediante observación GNSS en metodología estático diferencial se hace necesario calcular los tiempos de observación que se deben respetar para obtener determinada precisión, este tiempo de rastreo es directamente proporcional a la distancia que exista entre la base de referencia GNSS y el punto en observación, así como las condiciones en las que se encuentre el punto observado que pueden llegar a dificultar la observación como es el caso de la vegetación, construcciones y/o demás elementos que generen ruido en la observación. Este tiempo es determinada por lo general mediante el seguimiento de normas y procedimiento establecido por entidades oficiales que se encargan de establecer los parámetros para la generación de información geoespacial, en el caso concreto de Colombia el ente encargado de establecer estos modelos matemáticos para establecer los tiempos de rastreo es el IGAC, aunque dependiendo de la entidad a la cual se le entrega la información se pueden tener modelos diferentes modelos matemáticos para establecer el tiempo de rastreo.

3.1.15 Estaciones de Rastreo Continuo

Son un conjunto de estaciones GNSS usadas como referencia para la obtención de coordenadas mediante la observación GNSS, estas estaciones son usadas de forma pública y en Colombia se tienen dos grandes redes de este tipo de estaciones (Red MAGNA-Eco & GEORED), son usadas para levantamientos diferenciales y la información es de uso libre y es suministradas por las entidades estatales para su uso público en general. (Instituto Geográfico Agustín Codazzi IGAC, 2020)

Son utilizadas principalmente en la observación GNSS en metodología estático diferencial, para la obtención de puntos de referencia topográficos desde los cuales se realiza generalmente los levantamientos topográficos georreferenciados.

3.1.16 Tiempos de rastreo (estático diferencial)

Este método es utilizado para obtener las coordenadas ΔX, ΔY, ΔZ de una manera precisa. Esta metodología es usada comúnmente para la medición de puntos base, con equipos de doble frecuencia L1/L2. Se caracteriza por la ocupación simultanea de dos o más puntos por un tiempo específico según la distancia del punto de la Red Activa del IGAC más cercano. El tiempo de rastreo se rige por la regla: 15 minutos iniciales más la suma de 5 minutos por kilómetro que se separe de la estación IGAC más cercana. Este método requiere como mínimo 4 satélites ya que se caracteriza por la recepción de dichos satélites de manera fija, para poder garantizar precisiones en longitudes extensas respecto a la base. Se rige bajo el concepto DGPS (Differential GPS), que se basa en el concepto de estaciones terrestres de posición conocida, esta ubicación se compara con la obtenida por el GPS, se calculan las diferencias y se aplican las respectivas correcciones para que tengan el mismo valor como resultado. (Instituto Geográfico Agustín Codazzi IGAC, 2011)

Es importante mencionar que, al cabo del tiempo de posicionamiento los datos deben ser analizados en una computadora para realizar la corrección diferencial que permita conocer las coordenadas de los puntos posicionados mediante un postproceso que se constituye por diferentes insumos como lo son: las coordenadas semanales SIRGAS, las efemérides

precisas y los archivos rinex de la estación más cercana al punto de rastreo. (Huerta, Mangiaterra, & Noguera, 2005)

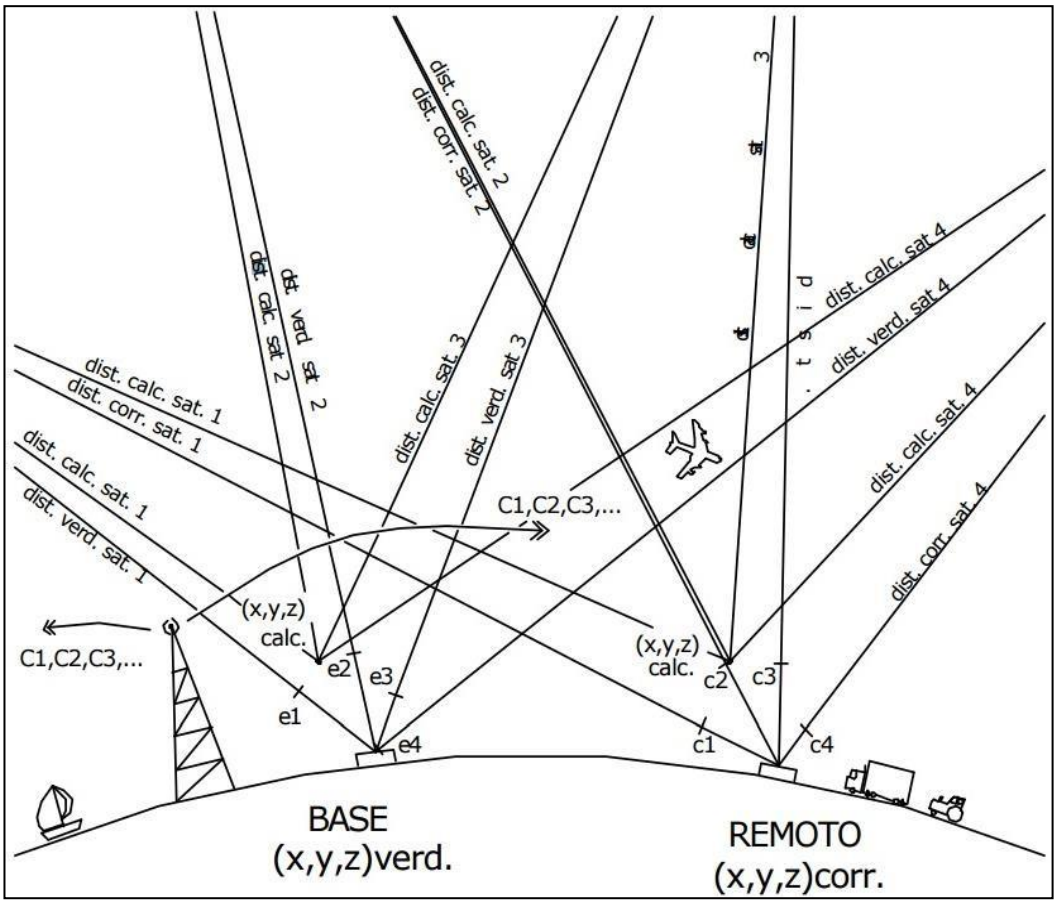


Ilustración 2 Método Estático Diferencial

Fuente: (Huerta, Mangiaterra, & Noguera, 2005)

3.1.17 Red pasiva control horizontal y alturas niveladas

La red pasiva del IGAC está constituida por 62 mojones ocupados con GPS en las campañas de 1994, 1995 y 1997. Sus coordenadas se definieron sobre el ITRF94 época 1995.4. Su precisión interna está en el orden de (+-2 mm … +- 7 mm), su exactitud horizontal en +-2 cm y la vertical en +-6 cm. En el año 2011 se realizaron mediciones sobre 42 de los 62 mojones para cambiar su época a 2012. En el año 2018 mediante la resolución 715 de 2018 del Instituto Geográfico Agustín Codazzi se actualiza el Marco Geocéntrico Nacional de

Referencia Magna Sirgas de ITRF94 época 1995.4 al ITRF2014 época 2018.0. (Instituto Geográfico Agustin Codazzi IGAC, 2020)

Las redes de nivelación en Colombia han sido establecidas por el Instituto Geográfico Agustín Codazzi IGAC a lo largo de las carreteras nacionales y siguiendo estándares técnicos del Servicio Geodésico Inter Americano. Se establecieron tres niveles de precisión:

Red de nivelación de primer orden: Contiene los circuitos básicos de nivelación con un diámetro promedio de 100 km. Estos han sido medidos con métodos geodésicos de alta precisión (Nivelación geométrica) y sus puntos se materializaron con incrustaciones de bronce en lugares geológicamente estables. La distancia entre los puntos varía 1.2 Km en áreas montañosas hasta 2.5 Km en zonas planas.

Red de nivelación de segundo orden: Éstas deben densificar los circuitos de primer orden, las mediciones de ida y vuelta deben estar alrededor de ±8mm<s [km]. A esta clase pertenecen circuitos que han sido nivelados con métodos de alta precisión en un solo sentido.

Red de nivelación de tercer orden: Éstas densifican las redes de primer y segundo orden para aplicaciones de precisiones menores. Sus errores de cierre no deben ser mayores de ±12mm<s [km]. Actualmente, la totalidad de las líneas de nivelación contiene más de 20 000 puntos a lo largo de 26 000 km. (Instituto Geográfico Agustín Codazzi IGAC, 2018)

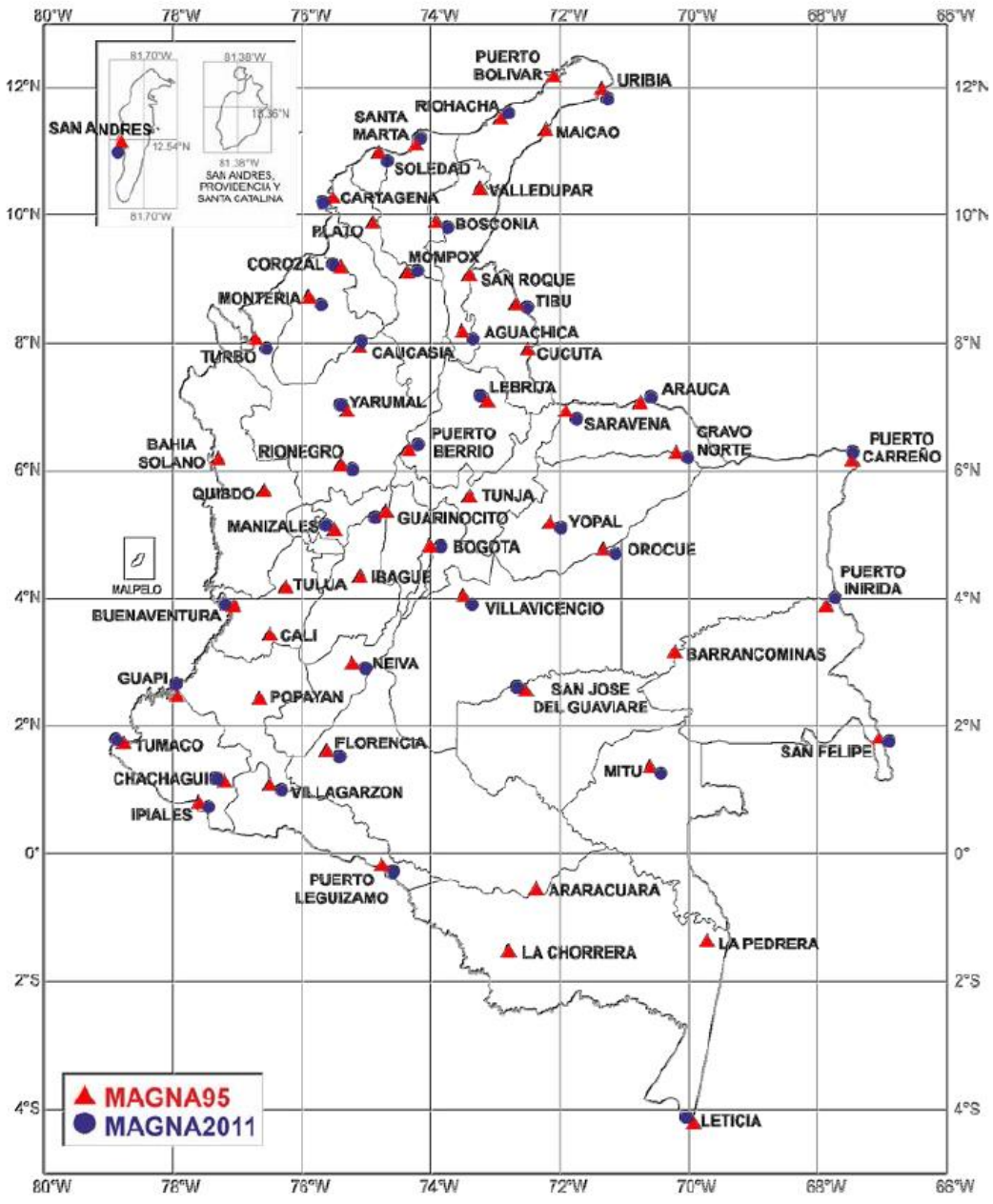


Ilustración 3 Red Pasiva Magna Sirgas

Fuente: (Instituto Geográfico Agustin Codazzi IGAC, 2020)

3.1.18 GeoRED

El proyecto GeoRED es un proyecto de investigación, innovación y desarrollo que es financiado por el Estado Colombiano con el propósito de estudiar la deformación de la corteza terrestre en el territorio colombiano mediante tecnologías GNSS. (Servicio Geológico Colombiano, 2020)

GeoRED corresponde a la denominación adaptada para el proyecto “Implementación de la Red Nacional de Estaciones Geodésicas Satelitales GPS con propósitos dinámicos” es un proyecto de investigación y desarrollo basado en instrumentación, bajo el marco de la Dirección Técnica de Geoamenazas, como resultado de consideraciones técnicas, económica y política con el propósito de aumentar el conocimiento de geodinámica que a su vez cumple un papel importante en la gestión del riesgo por fenómenos naturales. (Servicio Geológico Colombiano , 2020)

El objetivo general del proyecto es: **“*Mejorar la capacidad técnica, científica y operativa en Colombia para el análisis, interpretación y toma de decisiones de fenómenos asociados al estado de deformación tectónica regional y volcánica local en el territorio colombiano, empleando tecnología satelital GPS*”.** (Servicio Geológico Colombiano SGC)

Para Colombia es de gran importancia aumentar el conocimiento y la capacidad de captura y análisis de información relacionados con la geodinámica del territorio para modelar y realizar cuestionamientos en la toma de decisiones frente a amenazas sísmicas y volcánicas que se puedan presentar en el país. Esto puede reducir los riesgos de las poblaciones que se encuentran asociadas a estos tipos de desastres permitiendo optimizar la formulación de planes locales, departamentales, regionales y nacionales en zonas de amenaza baja, media y alta.

3.1.19 Coordenadas elipsoidales

Un elipsoide es una elipse que se encuentra en revolución, es utilizada de forma amplia en la geodesia ya que es la forma geométrica que más se asemeja a la verdadera forma de la tierra. A lo largo de la historia, han existido diferentes modelos elipsoidales utilizados en diferentes países, siendo inicialmente de mayor uso los elipsoides locales; los cuales por lo general no son geocéntricos, es decir, las coordenadas geocéntricas 0,0,0 del elipsoide no coinciden con el centro de masas de la tierra y adicionalmente puede que la este rotado y que el semieje mayor de este no coincida con el ecuador. En el caso de Colombia al igual que la mayoría de los países del mundo, debido a la necesidad cada día más notoria de tener

cartografía que pudiera ser utilizada de forma global, adopto inicialmente el elipsoide de referencia propuesto por Hayford 1910 como un sistema de referencia internacional, adoptando este elipsoide se creó el datum BOGOTA, bajo el cual se ha desarrollado la mayor parte de la cartografía del país. Posteriormente, se procedió a utilizar el elipsoide GRS80 sobre el cual se estableció el datum MAGNA SIRGAS, siendo este último el datum de referencia utilizado en la actualidad, ya que este datum es geocéntrico y permite la conversión de coordenadas entre sistemas de referencia de otros países logrando que la cartografía pueda ser proyectada y usada en otros territorios. (deTopografia, 2019)

Las posiciones X, Y, Z se expresan en coordenadas curvilíneas latitud (ϕ) y longitud (ʎ) cuyos valores se toman del elipsoide de referencia. El origen será entonces el centro geométrico del elipsoide. El eje X coincide con la intersección del plano ecuatorial y el meridiano de referencia del elipsoide, el eje Y forma un sistema coordenado de mano derecha y por último el eje Z coincide con el eje menor del elipsoide. (Instituto Geográfico Agustín Codazzi IGAC, 2004)

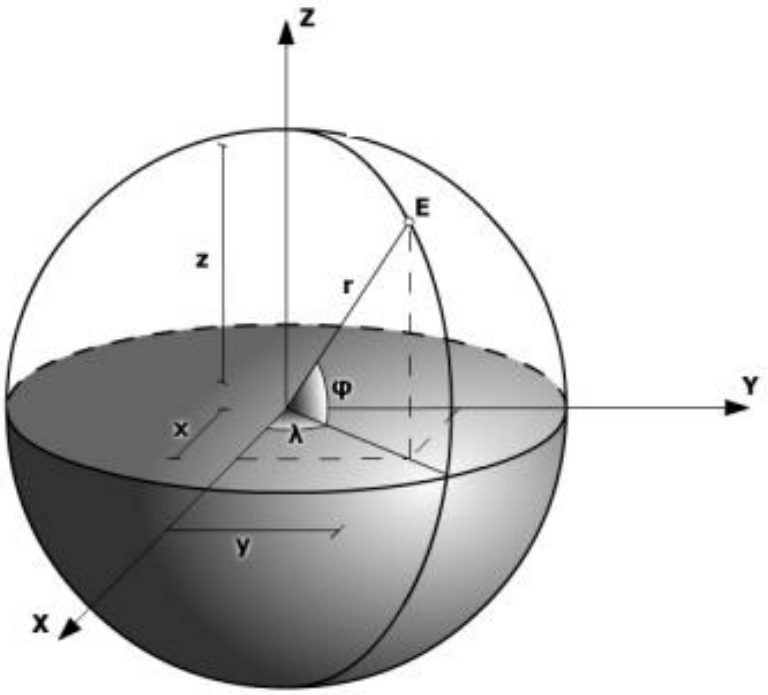


Ilustración 4 Coordenadas Geográficas

Fuente: (Furones, 2010)

Se llama latitud geográfica (ϕ) al ángulo que forma la vertical del punto E con el plano del Ecuador terrestre. Puede variar entre 0° a 90° en el hemisferio boreal (Latitud Norte) y de 0° a -90° en el hemisferio austral (Latitud Sur). Se llama Longitud geográfica (ʎ) al ángulo que forma el meridiano del punto con el meridiano de origen, este valor varía de 0° a 180° en el hemisferio oriental y de 0° a -180° en el hemisferio occidental. (Furones, 2010)

3.1.20 Coordenadas geocéntricas

El concepto del sistema se basa en la utilización de ejes de coordenadas fijos sobre el geocentro de la tierra, de manera que siempre sean las mismas. Tiene las siguientes características: El origen como geocentro de la tierra, el eje X se sitúa en dirección al plano meridiano Greenwich y contenido del plano del ecuador. El eje Y contenido en el plano del Ecuador perpendicular al eje de tal forma que se forme una tripleta dextrógira. El eje Z que coincida con el eje de rotación terrestre, el plano meridiano como un punto en la tierra que contiene el eje de rotación y por último el plano ecuador que es perpendicular al eje de rotación. (Furones, 2010)

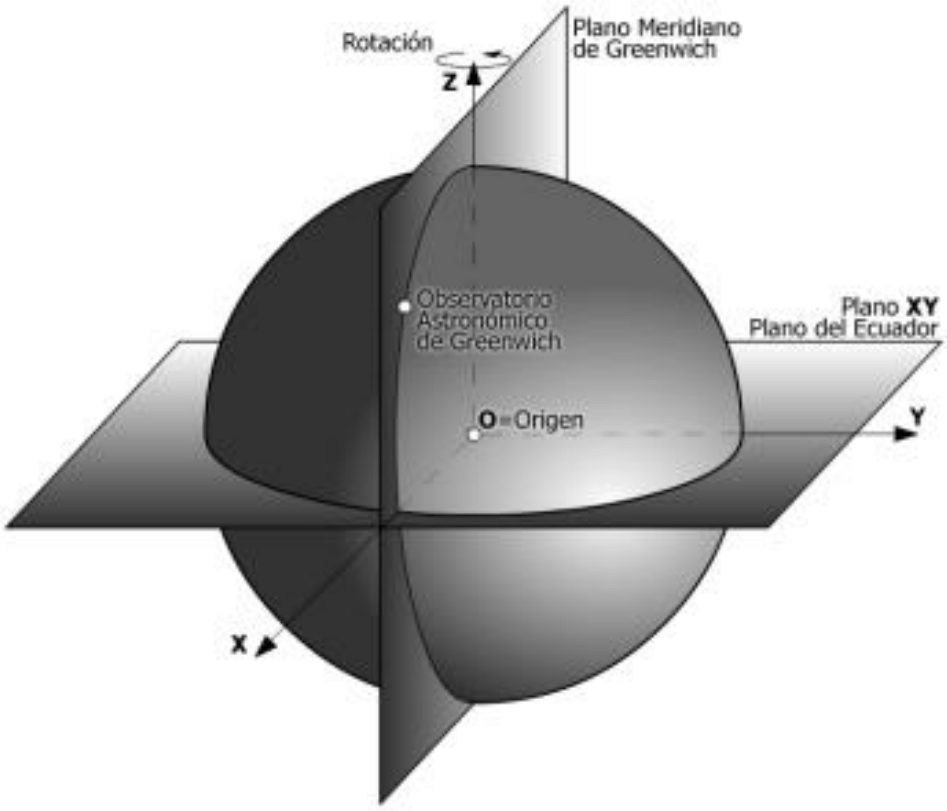


Ilustración 5 Coordenadas Geocéntricas

Fuente: (Furones, 2010)

3.1.21 Transversal de Mercator

La proyección Transversal de Mercator es una proyección de tipo cilíndrico, también es conocida como proyección Gauss-Kruger, estando diferenciada porque mientras la Transversal de Mercator tiene el eje del cilindro en el Ecuador, la Gauss-Kruger su eje puede estar ubicado en un párelo diferente de esta manera disminuyendo la distorsión de los elementos lejanos al paralelo de origen del sistema de coordenadas proyectado, fue desarrollada por Gerardus Mercator en 1569, con el objetivo de mostrar líneas de rumbo usadas principalmente en la navegación. (ESRI, 2019)

El sistema de proyección corresponde con el de Gauss Krügger, con la diferencia de que se controla con husos de 6° y tiene un factor de escala equivalente a m = 0.9996. (Instituto Geográfico Agustín Codazzi IGAC, 2004)



Ilustración 6 Proyección Transversal de Mercator

Fuente: (Naukas, 2019)

3.1.22 Coordenadas Gauss Krüger

La proyección cartográfica oficial de Colombia es el sistema Gauss-Krüger, es una representación conforme del elipsoide sobre un plano, es decir que el ángulo que es formado por dos líneas sobre la superficie se mantiene al ser proyectado sobre un plano. Los meridianos y paralelos se intersecan perpendicularmente, pero no son líneas rectas, por el contrario, se consideran curvas complejas. La proyección se controla mediante husos que están separados 1.5° a cada lado del meridiano central. Su origen se encuentra ubicado en el Observatorio Astronómico de Bogotá asignando los valores de 1000000 m N y 1000000 m E. Los orígenes complementarios están controlados a 3° y a 6° de longitud al este y al oeste de dicho punto. Las coordenadas planas de Gauss son obtenidas a partir de las coordenadas elipsoidales. (Instituto Geográfico Agustín Codazzi IGAC, 2004)

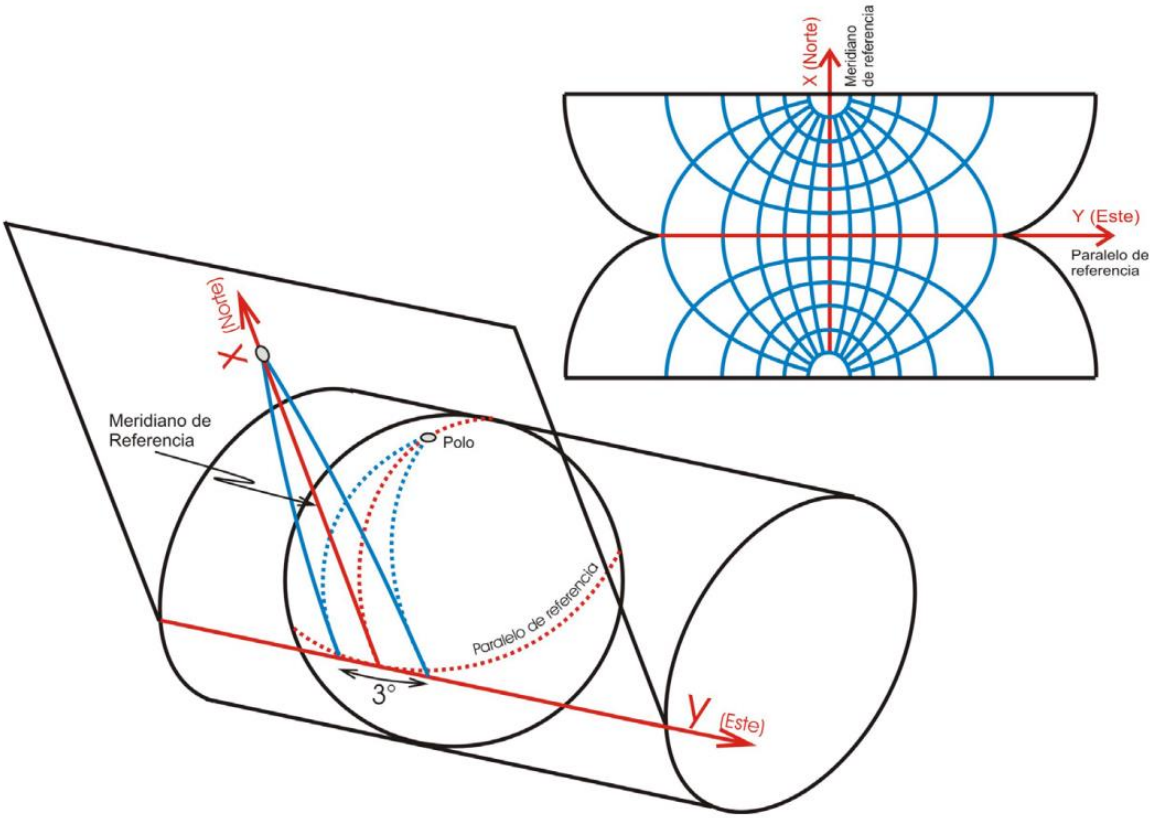


Ilustración 7 Proyección cartográfica Gauss Krüger

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2004)

3.1.23 Coordenadas planas cartesianas

Es una representación conforme del elipsoide sobre un plano paralelo, es ubicado con una altura Ho, a uno tangente del elipsoide en un punto origen de latitud y longitud. La proyección del meridiano que pasa por este punto es representada por la coordenada norte. (Instituto Geográfico Agustín Codazzi IGAC, 2004)

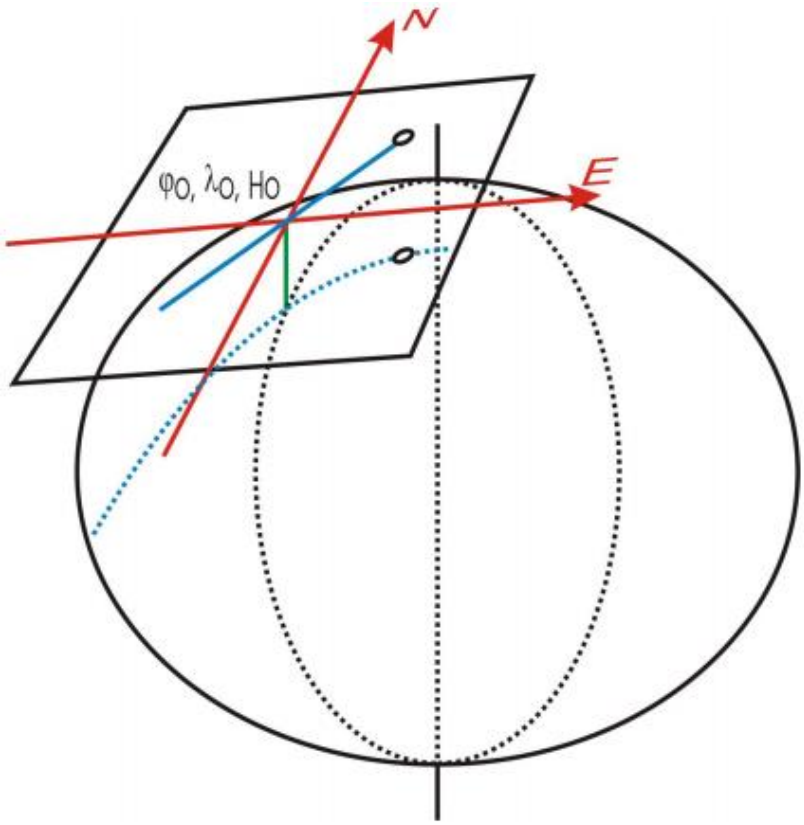


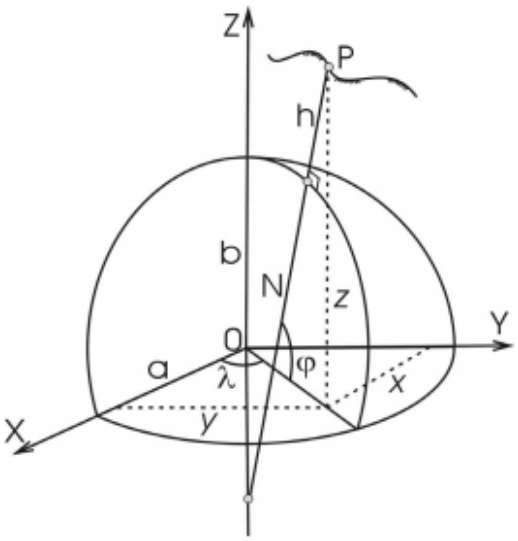
Ilustración 8 Obtención de coordenadas planas cartesianas

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2004)

3.1.24 Conversión de Coordenadas

La conversión de coordenadas tiene como base las coordenadas elipsoidales. Es decir, las conversiones se realizan desde el sistema de origen a elipsoidal y posteriormente al sistema de coordenadas de destino. Las posiciones son expresadas en términos de Latitud (ϕ) y longitud (ʎ) debido a que requieren la introducción de un elipsoide de referencia. La ubicación del elipsoide asociado a un sistema de coordenadas [XYZ] se conoce como Datum Geodésico; este a su vez puede ser geocéntrico o local. Los datum locales tienen sus coordenadas (ϕ, ʎ) definidas independientemente de la altura H, la cual es definida sobre el nivel medio del mar. Por otro lado, los datum geocéntricos son tridimensionales, lo cual permite definir las coordenadas de un punto respecto a la misma superficie de referencia, denominada; altura geodésica o elipsoidal. (Instituto Geográfico Agustín Codazzi IGAC, 2004)

Las conversiones entre coordenadas cartesianas geocéntricas [XYZ] y coordenadas elipsoidales [ϕ, ʎ, h] corresponden a las siguientes relaciones matemáticas:



a	semieje mayor del elipsoide de referencia
b	semieje menor del elipsoide de referencia
e^2	primera excentricidad del elipsoide de referencia
e'^2	segunda excentricidad del elipsoide de referencia
N	Radio de curvatura principal o gran normal:

$$N = \frac{a}{\sqrt{1 - e^2 \sin^2 \varphi}}$$

Ilustración 9 Coordenadas rectangulares geocéntricas y elipsoidales

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2004)

Donde:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (N+h)\cos\varphi\cos\Lambda \\ (N+h)\cos\varphi\sin\Lambda \\ ((1-e^2)N+h)\sin\varphi \end{bmatrix}$$

$$\tan\varphi = \frac{Z + e'^2 b \sin^3\vartheta}{\sqrt{X^2+Y^2} - e^2 a\cos^3\vartheta}$$

$$\tan\vartheta = \frac{Z\,a}{\sqrt{X^2+Y^2}\,b}$$

$$\tan\Lambda = \frac{Y}{X}$$

$$h = \frac{\sqrt{X^2+Y^2}}{\cos\varphi} - N$$

La conversión entre coordenadas elipsoidales [ϕ, ʎ, h] y planas de Gauss [N, E] tienen la siguiente relación matemática:

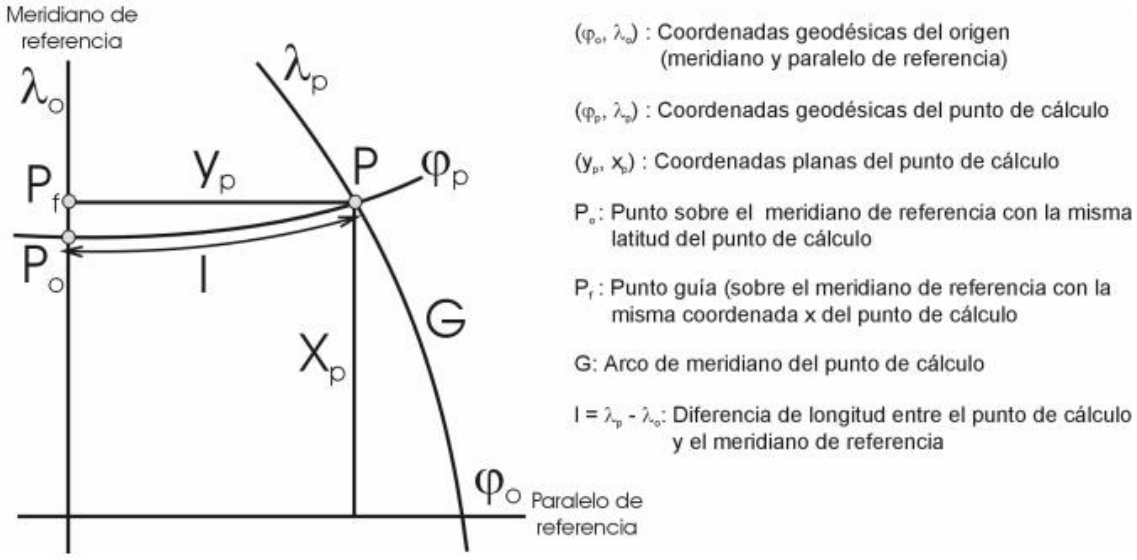


Ilustración 10 Coordenadas elipsoidales y Planas de Gauss

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2004)

Coordenada Norte:

$$N = \big(G(\varphi_P) - (\varphi_0)\big) + \frac{t}{2} N\, l^2\, cos^2\varphi_P + \frac{t}{24}\, N\, cos^4\varphi_P (5 - t^2 + 9\eta^2 + 9\eta^4)\; l^4$$
$$+ \frac{t}{720}\, N\, cos^6\varphi_P\; (61 - 58t^2 + t^4 + 270\eta^2 - 330t^2\eta^2)\, l^6$$
$$+ \frac{t}{40320}\, N\, cos^8\varphi_P (1385 - 3111t^2 + 543t^4 - t^6)\, l^8 + \cdots$$
$$+ 1000\,000{,}0$$

Coordenada Este:

$$E = N\, l \cos\varphi_P + \frac{1}{6}\, N\, cos^3\varphi_P(1 - t^2 + \eta^2)\, l^3 + \frac{1}{120}\, N\, cos^5\varphi_P(5 - 18t^2 + t^4 + 14\eta^2 - 58t^2\eta^2)\, l^5 + \frac{1}{5040} N\, cos^7\varphi_P(61 - 479t^2 + 179t^4 - t^6)\, l^7 + \cdots + 1000\,000{,}0$$

Siendo:

$$l = \Lambda_p - \Lambda_0 \;\; ; \;\; t = \tan\varphi_p \;\; ; \;\; \eta^2 = e'^2 cos^2\varphi_P \;\; ; \;\; N = \frac{a}{\sqrt{1 - e^2 sin^2\varphi_P}}$$

Arco de meridiano del punto de cálculo G (φ_p):

$$G\left(\varphi_p\right) = \alpha\,[\varphi + \beta \sin 2\varphi + \gamma \sin 4\varphi + \delta \sin 6\varphi + \varepsilon \sin 8\varphi + \cdots]$$

$$\alpha = \frac{a+b}{2}\left(1 + \frac{1}{4}n^2 + \frac{1}{64}n^4 + \cdots\right)$$

$$\beta = -\frac{3}{2}n + \frac{9}{16}n^3 - \frac{3}{32}n^5 + \cdots$$

$$\gamma = \frac{15}{16}n^2 - \frac{15}{32}n^4 + \cdots$$

$$\delta = -\frac{35}{48}n^3 + \frac{105}{256}n^5 - \cdots$$

$$\varepsilon = \frac{315}{512}n^4 + \cdots$$

$$n = \frac{a-b}{a+b}$$

Donde a, b, e^2, e'^2 son las constantes del elipsoide de referencia y φ se expresa en radianes.

El cálculo de coordenadas elipsoidales [ϕ, ʎ] a partir de coordenadas planas Gauss Kruger [N, E] se adelanta a través de:

Latitud:

$$\varphi = \varphi_f + \frac{t_f}{2N_f^2}\left(-1-\eta_f^2\right)\Delta E^2 + \frac{t_f}{24N_f^4}\left(5+3t_f^2+6\eta_f^2-6t_f^2\eta_f^2-3\eta_f^4-9t_f^2\eta_f^4\right)\Delta E^4 + \frac{t_f}{720N_f^6}\left(-61-90t_f^2-45t_f^4-107\eta_f^2+162t_f^2\eta_f^2+45t_f^4\eta_f^2\right)\Delta E^6 + \frac{t_f}{40320N_f^8}\left(1385+3633t_f^2+4096t_f^4+1575t_f^6\right)\Delta E^8 + \cdots$$

Longitud:

$$\lambda = \lambda_0 + \frac{1}{N_f\cos\varphi_f}\Delta E + \frac{1}{6N_f^3\cos\varphi_f}\left(-1-2t_f^2-\eta_f^2\right)\Delta E^3 + \frac{1}{120N_f^5\cos\varphi_f}\left(5+28t_f^2+24t_f^4+6\eta_f^2+8t_f^2\eta_f^2\right)\Delta E^5 + \frac{1}{5040N_f^7\cos\varphi_f}\left(-61-662t_f^2-1320t_f^4-720t_f^6\right)\Delta E^7 + \cdots$$

Latitud punto guía φ_f:

$$\varphi_f = \frac{\Delta N}{\bar{\alpha}} + \bar{\beta}\sin\frac{2\Delta N}{\bar{\alpha}} + \bar{\gamma}\sin\frac{4\Delta N}{\bar{\alpha}} + \bar{\delta}\sin\frac{6\Delta N}{\bar{\alpha}} + \bar{\varepsilon}\sin\frac{8\Delta N}{\bar{\alpha}} + \cdots$$

$$\bar{\alpha} = \frac{a+b}{2}\left(1+\frac{1}{4}n^2+\frac{1}{64}n^4+\cdots\right)$$

$$\bar{\beta} = \frac{3}{2}n - \frac{27}{32}n^3 + \frac{269}{512}n^5 + \cdots$$

$$\bar{\gamma} = \frac{21}{16}n^2 - \frac{55}{32}n^4 + \cdots$$

$$\bar{\delta} = \frac{151}{96}n^3 - \frac{417}{128}n^5 - \cdots$$

$$\bar{\varepsilon} = \frac{1097}{512} n^4 + \cdots$$

$$n = \frac{a-b}{a+b}$$

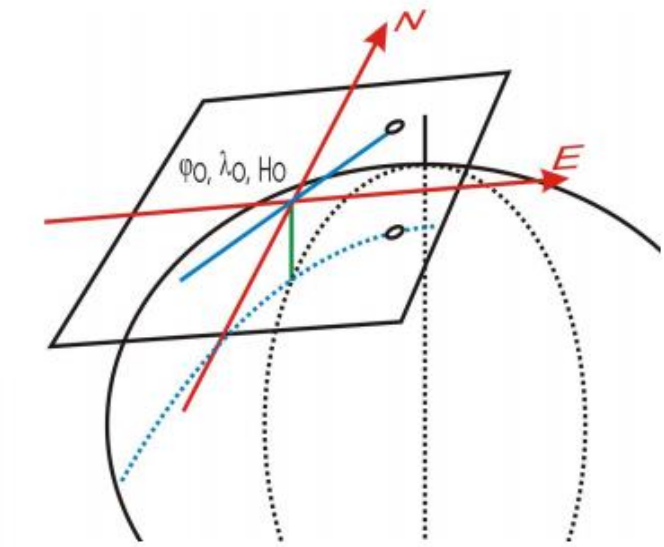
Siendo:

$$\Delta N = N - N_0 \quad ; \quad \Delta E = E - E_0 \quad ; \quad t_f = \tan \varphi_f \quad ; \quad \eta_f^2 = e'^2 cos^2 \varphi_f \quad ;$$

$$N = \frac{a}{\sqrt{1 - e^2 sin^2 \varphi_f}}$$

N_0 corresponderá con la coordenada norte para ϕ = 0 (ecuador), siendo para el datum Magna Sirgas $N_0 = 491\,767,5344$

La conversión de coordenadas elipsoidales [ϕ, ʎ, h] y planas cartesianas [N, E] tiene la siguiente relación matemática:



N, E: coordenadas cartesianas planas del punto de cálculo

φ_o, λ_o : coordenadas geográficas del origen del sistema de coordenadas cartesianas

Ilustración 11 Coordenadas elipsoidales y planas cartesianas

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2004)

$$N = M_{\varphi_0}\left[\Delta\varphi_{rad} + \frac{\tan\varphi_0\ \Delta\Lambda_{rad}^2\ N_\varphi^2 cos^2\varphi}{2M_{\varphi_0}N_{\varphi_0}}\right]X\left[1 + \frac{P_p}{M_{\varphi m}}\right] + N_0$$

$$E = \Delta\Lambda_{rad}N_\varphi\cos\varphi\left[1 + \frac{P_p}{N_{\varphi 0}}\right] + E_0$$

Igualmente, la conversión de coordenadas cartesianas a geodésicas tiene la relación:

$$\varphi = \varphi_0 + \Delta\varphi$$

$$\Lambda = \Lambda_0 + \Delta\Lambda$$

$$\Delta\varphi_{rad} = \frac{\Delta N}{\left[1 + \frac{P_P}{a(1-e^2)}\right]M_{\varphi 0}} - \left[\frac{\tan\varphi_0}{2\ M_{\varphi_0}N_{\varphi_0}}\ X\left(\frac{\Delta E}{1 + \frac{P_P}{a}}\right)^2\right]$$

$$\Delta\Lambda_{rad} = \frac{\Delta\ Este}{N_\varphi\cos\varphi\left(1 + \frac{P_P}{a}\right)}$$

Siendo:

N, E: Coordenadas cartesianas planas del punto de cálculo

N_0, E_0: Origen del sistema de coordenadas cartesianas

$$\Delta N = N - N_0$$

$$\Delta E = E - E_0$$

φ, Λ: Coordenadas geográficas del punto de cálculo

φ_0, Λ_0: Coordenadas geográficas del origen del sistema de coordenadas cartesianas

$$\Delta\varphi = \varphi - \varphi_0$$

$$\Delta\lambda = \lambda - \lambda_0$$

$$\varphi_m = \frac{(\varphi - \varphi_0)}{2}$$

P_P: Altura del plano de proyección

a: Semieje mayor del elipsoide de referencia

e^2: Primera excentricidad del elipsoide de referencia

$$N = \frac{a}{\sqrt{1 - e^2 sin^2\varphi}}$$

$$M = \frac{a\,(1 - e^2)}{(1 - e^2 sin^2\varphi)^{3/2}}$$

4. DESARROLLO DEL PROYECTO

La metodología de trabajo fue desglosada mediante el enfoque sistemático BPN *Business Process Management* o "*gestión de procesos de negocios*" combinando metodologías y tecnologías con el fin de sustentar el ciclo de mejora continua según cuatro etapas fundamentales:

1. Modelamiento: basado en un ciclo gráfico que permita visualizar y argumentar las etapas.
2. Automatización: Se exponen los participantes en cada etapa y su rol en cada fase del flujo de trabajo.
3. Tiempos y cantidades: Teniendo en cuenta los indicadores de rendimiento KPI's *Key Performance Indicators* para identificar puntos críticos de mejora.
4. Optimización: implementar las mejoras derivadas del análisis de los indicadores de rendimiento para depurar y optimizar el proceso permitiendo volver al punto 1.

La gestión y el mejoramiento de los procesos hace parte de pilares fundamentales de los principios de la calidad total. Por lo cual, implica un enfoque hacia el cliente externo desplegando sus necesidades y expectativas siendo mejoradas según la gestión y el mejoramiento de procesos. (Scholarium, 2018)

Dentro de la metodología BPN se encuentra el modelado de procesos BPNM *Business Process Model and Notation.* Es un estándar mundial de notación gráfica estandarizada que permite el modelado de procesos o negocio a través de un flujo de trabajo. Su principal objetivo es proveer una notación fácilmente legible por los analistas y administradores del negocio para crear una comunicación eficiente de entre el diseño de procesos y su implementación. (Sistema gestor de trabajo Flokzu, 2018)

El BPNM Está caracterizado por tres formas primarias, llamadas nodos de flujo:

1. Rectángulo redondeado: Especifica una actividad o un paso de acción en el proceso.
2. Rombo: Denota una ramificación condicional.
3. Círculo: Inicio o fin de un proceso.

Por otro lado, la flecha sólida es un flujo de secuencia que se pone al inicio y al final de cada forma.

Gráfico BPNM:

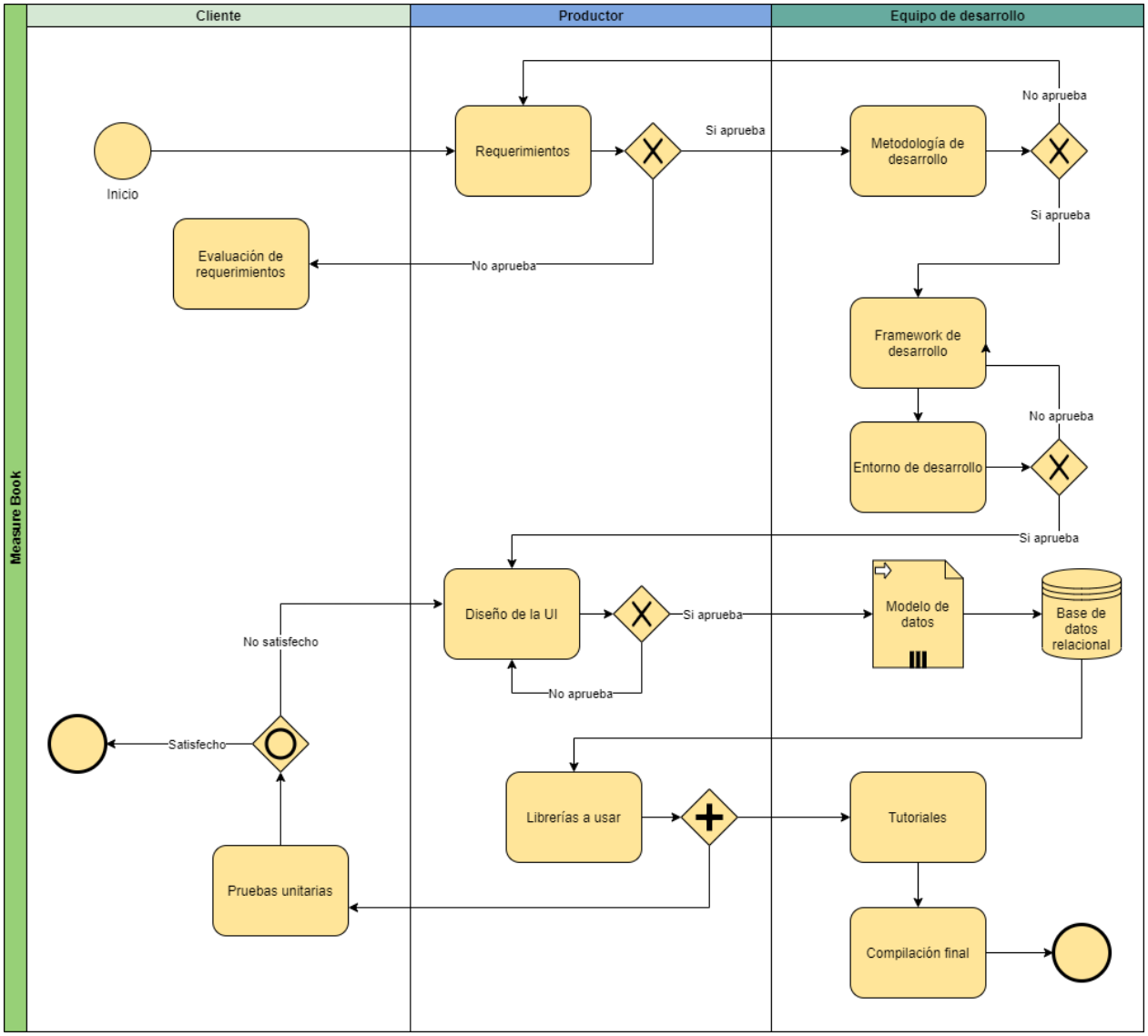


Ilustración 12 Plan de trabajo desarrollo App

Fuente: Elaboración propia

4.1 Etapa I Requerimientos

En la primera etapa de la metodología de trabajo, se procedió a establecer claramente las funcionalidades y herramientas que debería tener la aplicación. Estas funcionalidades fueron determinadas a partir de la experiencia profesional en el sector y el análisis de la posible solución que podría brindar la aplicación asumiendo el rol de cliente.

- La aplicación permitirá la navegación de puntos de la red pasiva
  - La navegación de los puntos se podrá realizar sin acceso a internet
  - Se realizará un clúster dinámico de los puntos con el fin de facilitar la visualización de los mismos.
  - Los puntos mostraran toda la información suministrada por el IGAC para cada uno de estos (Velocidad, fecha de Instalación, coordenadas Elipsoidales, Geocéntricas).
- La aplicación permitirá determinar los tiempos de rastreo de observaciones GNSS en estático diferencial usando como puntos de referencia las estaciones de rastreo continuo del IGAC
  - Se permitirá calcular los tiempos de rastreo según lo establecido para observaciones GNSS en metodología estático diferencial según las normas del IGAC, EAB e IDU.
- La aplicación permitirá la conversión de coordenadas dentro del Datum MAGNA SIRGAS
  - La conversión de coordenadas se podrá realizar entre los diferentes de proyección usados en Colombia
  - Se deberá permitirá la conversión para todos los planos cartesianos existentes dentro del software MAGNA PRO
  - Se podrá permitirá convertir archivos de coordenadas en formato CSV

Historias de Usuarios:

Tabla 1 Historia de Usuario - Navegación red pasiva y altura nivelada

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Usuario: Prueba de campo 1 – Jefferson Betancourt</td></tr>
<tr><td colspan="3">Nombre historia: Navegación red pasiva IGAC – Altura nivelada</td></tr>
<tr><td colspan="2">Prioridad en negocio: Alta</td><td>Riesgo en desarrollo: Alto</td></tr>
<tr><td colspan="2">Puntos estimados: 1</td><td>Iteración asignada: 1</td></tr>
<tr><td colspan="3">Programador responsable: Layda Yaneth Gil Ángel</td></tr>
<tr><td colspan="3">Descripción: Se pretende realizar la navegación de puntos de la red pasiva del IGAC en el municipio de Puerto Boyacá en el departamento de Boyacá. Encontrando como más cercano el punto 15572014 con fecha de instalación del año 2015. Coordenadas geográficas Latitud 6.110357008 N Longitud -74.55462121 W. Se procede a su vez a buscar el punto para obtener la altura nivelada 140.401 msnm.</td></tr>
<tr><td colspan="3">Validación:

<table>
<tr><th>Escenario</th><th>Descripción</th></tr>
<tr><td>Escenario 1</td><td>Se identifica en la interfaz la opción de “Red pasiva IGAC” para acceder a la actividad creada para la navegación de los puntos, teniendo en cuenta el mapa base y la capa de puntos que constituye la actividad. Se procede a usar el botón de ubicación</td></tr>
</table>
</td></tr>
</table>

	geográfica y se activa un zoom in a la zona de estudio. Se encuentra el punto más cercano, se da clic y se obtienen los datos del punto.
Escenario 2	Se identifica en la interfaz la opción “Alturas niveladas”, se procede a ejecutar la actividad y activar la ubicación geográfica. Se encuentran los puntos más cercanos y se inspecciona la información del punto.

Tabla 2 Historia de usuario - Conversión de coordenadas

Historia de Usuario	
Número: 2	**Usuario**: Prueba de oficina – Edisson Camilo Sánchez
Nombre historia: Conversión de coordenadas sistema MAGNA SIRGAS	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: Alto
Puntos estimados: 1	**Iteración asignada**: 1
Programador responsable: Rafael Fabian Sánchez Osorio	
Descripción: Se pretende realizar la conversión de coordenadas del punto CD-774 con coordenadas geográficas Latitud 4.595682923 N Longitud -74.16604336 W altura elipsoidal 2582.329 a coordenadas geocéntricas XYZ.	
Validación:	

Escenario	Descripción
Escenario 1	Se procede a examinar la opción de conversión de coordenadas obteniendo los valores 1735422.098 X -6119007.522 Y 507837.412 Z. Se verificó a su vez con el software magna pro 4.2 y con las coordenadas geocéntricas que se encuentran registradas del punto, obteniendo una coincidencia perfecta en la conversión.

Tabla 3 Historia de usuario - Cálculo de tiempo de rastreo red activa IGAC y GeoRed

Historia de Usuario	
Número: 3	**Usuario**: Prueba de oficina – Juliana Sanchez
Nombre historia: Cálculo de tiempo de rastreo red activa IGAC Y GeoRed	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: Alto
Puntos estimados: 1	**Iteración asignada**: 1
Programador responsable: Layda Yaneth Gil Ángel – Rafael Fabian Sánchez Osorio	
Descripción: Se pretende calcular el tiempo de rastreo con la red MAGNA ECO y la red GeoRed para realizar una comparación del tiempo sugerido respecto a la norma IDU para la elaboración de estudios topográficos.	

Validación:

Escenario	Descripción
Escenario 1	Se establece como ubicación geográfica el punto con coordenadas 4° 35’ 49.5251” N -74° 3’ 56.3346” W obteniendo como resultado las estaciones BOGA y AEGU del IGAC con tiempo máximo de rastreo 2 horas y 23 minutos según la norma del IDU.
Escenario 2	Se establece como ubicación geográfica el punto con coordenadas 4° 35’ 49.5251” N -74° 3’ 56.3346” W obteniendo como resultado las estaciones BOGT y BP01 de GeoRed con tiempo máximo de rastreo: 50 minutos según la norma del IDU.

4.2 Etapa II Análisis

Definidos los requerimientos de la aplicación, se procedió a realizar el desarrollo de la segunda etapa en la cual se analizó y estableció la metodología de desarrollo, el modelo de datos a utilizar y el diseño de la base de datos relacional.

4.2.1 Metodología de Desarrollo

Con el fin de agilizar el desarrollo del proyecto, se procedió a realizar una consulta de las metodologías agiles de desarrollo de software más usadas en la actualidad, con el fin de establecer a cuál de estas nos podíamos acoger para el desarrollo del proyecto, teniendo en cuenta los recursos con lo que se contaban.

Dentro de las opciones de desarrollo encontradas se optó por usar la establecido dentro del framework SCRUM. Principalmente porque permitía ir desarrollando requerimientos y evaluarlos para establecer si estos cumplían con los que se plantearon inicialmente.

4.2.2 Scrum

La metodología de desarrollo será “Scrum” en la cual se intentan desarrollar los proyectos de forma incremental e iterativa, en un ciclo. Se enfoca en que cada ciclo especifique las mejoras necesarias para que se redirija de la forma en la que lo necesite cada cliente. Esto potencializa la finalidad de la aplicación y simplifica la arquitectura planteada y los requerimientos de cada elemento que constituye el proyecto. Se basa en la capacidad multidisciplinaria y autonomía para desarrollar cada etapa del proyecto y agregar valor en cada una de las etapas del ciclo, estos ciclos se denominan “spring”. Los roles que conforman el equipo de scrum son:

Productor: Líder que vela por los intereses del cliente para satisfacer las necesidades del proyecto.

Equipo de desarrollo: Grupo interdisciplinario que debe reunir las habilidades necesarias para planificar, desarrollar y plantear las soluciones estratégicas que requiera el proyecto.

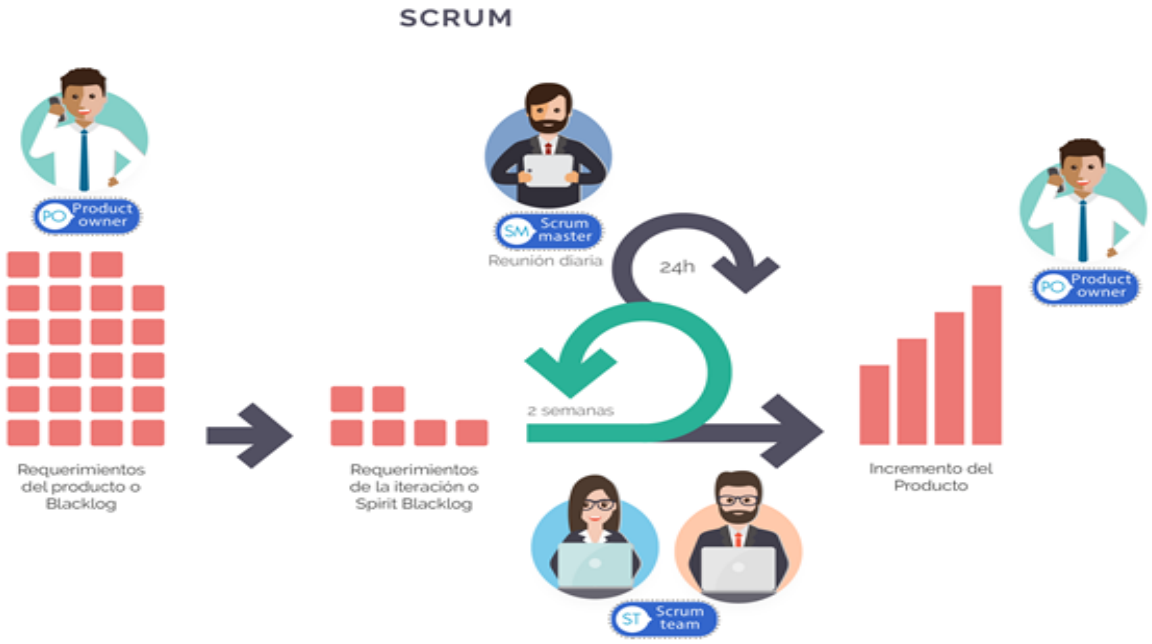


Ilustración 13 Metodología scrum

Fuente: (Blog CONECTART)

4.2.3 Framework de Desarrollo

Definida la metodología de desarrollo se procedió a escoger el marco de trabajo con el cual se desarrollaría la aplicación. Esta debería cumplir con la totalidad de las siguientes características:

- Librerías para la creación de visores cartográficos que permitan la navegación con el GPS interno del celular para la búsqueda de puntos de referencia.
- Librerías con el almacenamiento de datos en base de datos SQLite.
- Librerías que permitan cargar y leer archivos en formato CSV para la conversión de coordenadas.
- Creación de aplicaciones nativas para Android.

Una vez validados los requerimientos que debería tener el marco de desarrollo para lograr llevar a cabo el desarrollo de la APP, se procedió a revisar la documentación del Framework y los diferentes entornos de desarrollo dentro de los cuales era compatible.

Después de consultar y realizar diferentes pruebas el equipo de trabajo optó por realizar el desarrollo de la aplicación dentro del Framework Flutter. Esto debido a que es de código abierto y permite el desarrollo de aplicaciones nativas multiplataforma (Android, iOS, Web, Windows, MAC y WEB). Es importante tener en cuenta que, fue de gran importancia para la elección del framework que permitiera migrar la aplicación desarrollada naturalmente para Android a otras plataformas. Adicionalmente, estaba constituido por widgets prefabricados que facilitaban la construcción de la interfaz y permitía compilar a código de procesador, lo cual permitía tener un desarrollo nativo en comparación con las demás opciones multiplataforma.

4.2.4 Entorno de Desarrollo (IDE)

Definido el marco de trabajo se procedió a establecer el entorno de desarrollo. Debía ser compatible con el marco de trabajo seleccionado, en este caso se tenían dos entornos de desarrollo para realizar la aplicación: la primera de estas es Android Studio y la Segunda Visual Studio Code.

Analizados los entornos de desarrollo se optó por realizar el desarrollo de la aplicación dentro del entorno; Visual Studio Code, dadas las facilidades para instalar complementos y el buen funcionamiento de estos, en comparación con Android Studio.

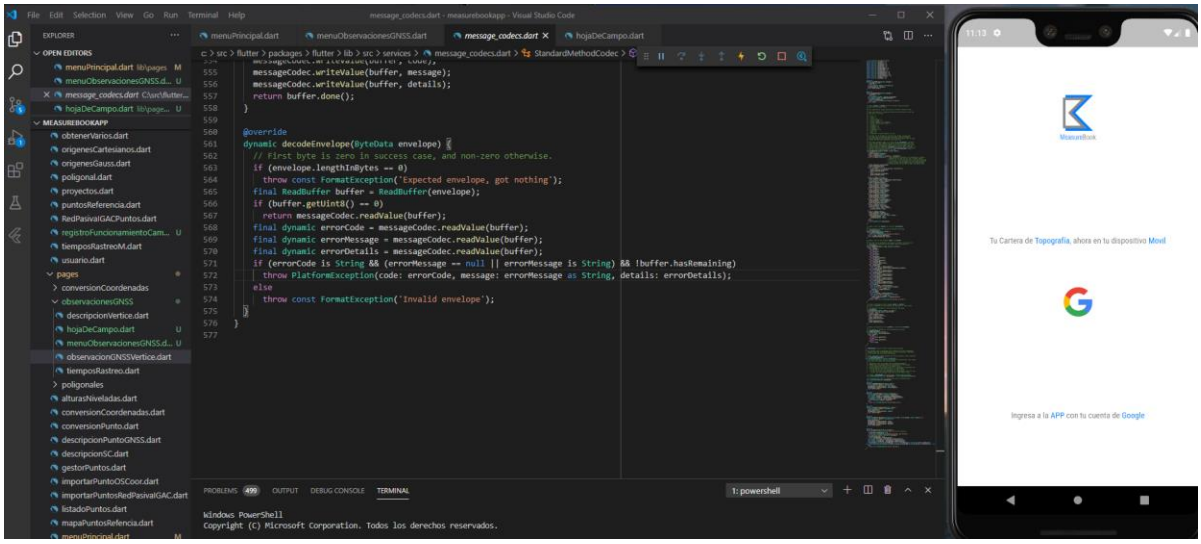


Ilustración 14 Entorno de Desarrollo Visual Studio Code

Fuente: Elaboración Propia

4.3 ETAPA III

4.3.1 Diseño de UI

Para el desarrollo de la UI se utilizó el software Adobe XD, el cual cuenta con diferentes herramientas que facilitan el diseño de la interfaz. Adicionalmente, permite la instalación de complementos prediseñados que facilitan el diseño de la interfaz.

El diseño de la UI fue realizado teniendo en cuenta los requerimientos iniciales, y con base en el diseño de la interfaz se procedió a establecer los diferentes componentes funcionales, desarrollando de esta forma un maquetado general de la aplicación.

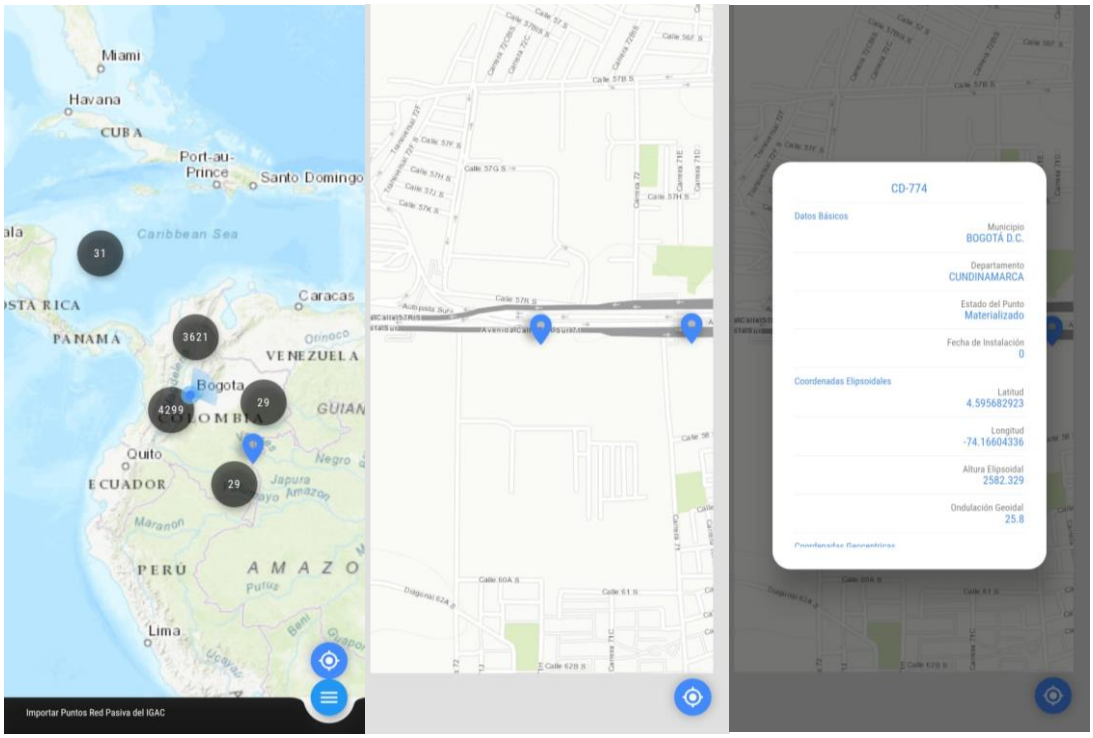


Ilustración 15 Navegación Red Pasiva del IGAC

Fuente: Elaboración Propia

En la ilustración anterior, se aprecia el diseño de la UI para la actividad de navegación de puntos pertenecientes a la red pasiva del IGAC. Permite visualizar con mayor facilidad la cantidad de puntos disponibles por zona, estas agrupaciones se desintegran cuando el usuario realiza se acerca a su zona de interés. Adicionalmente, la interfaz cuenta con un botón que permite obtener la ubicación geográfica del usuario, mostrando los puntos más cercanos. Es importante mencionar que, cada uno de los puntos despliega una tabla de atributos con las coordenadas elipsoidales, geocéntricas, sus velocidades y datos generales como; año de materialización y nomenclatura del punto.

4.3.2 Diseño Base De Datos Relacional

El diseño de modelo de datos relacional fue realizado buscando tener la menor redundancia de dato posible y facilitando la gestión de la información dentro del funcionamiento de la aplicación.

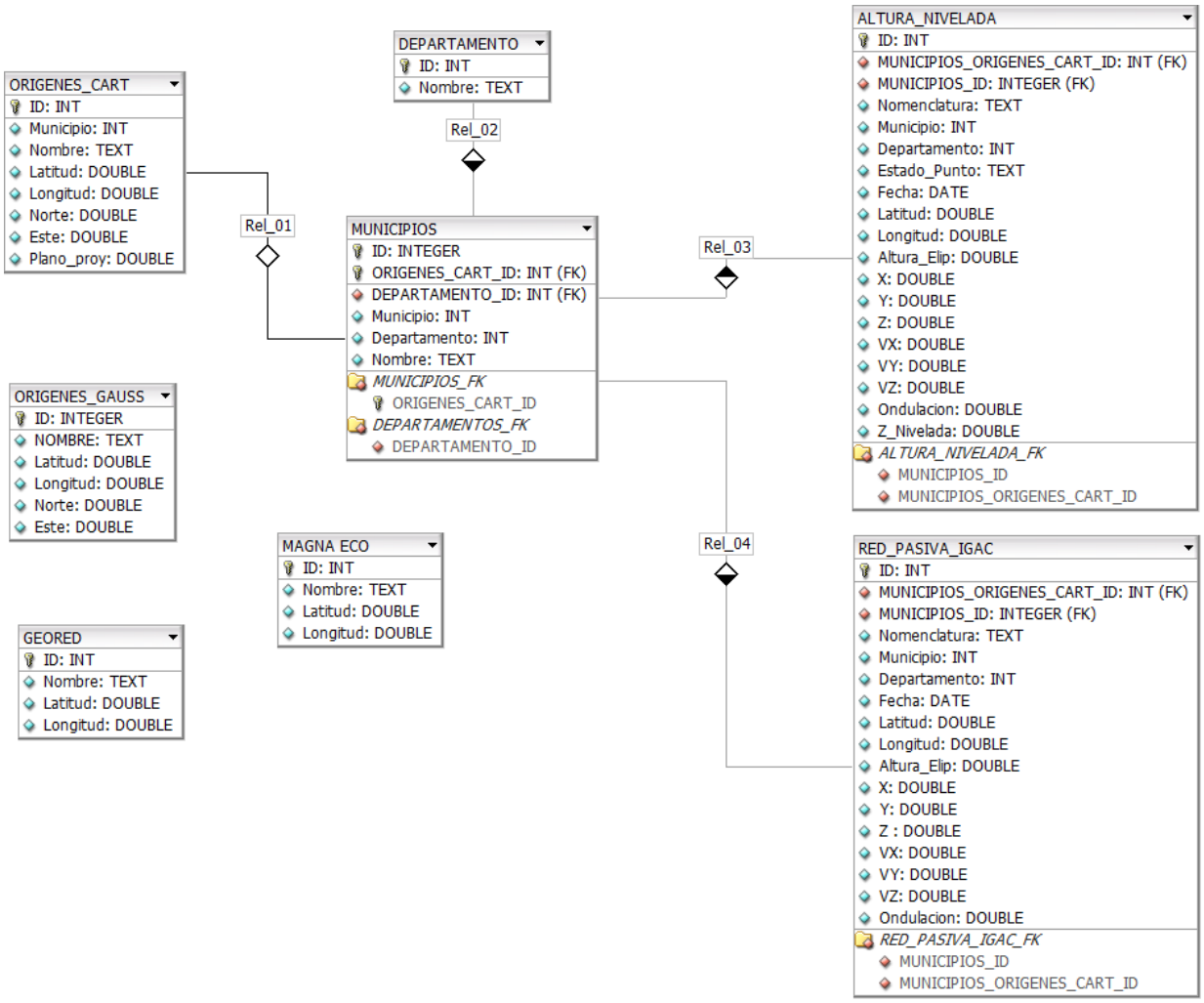


Ilustración 16 Modelo de Datos Relacional Utilizado

Fuente: Elaboración Propia

4.3.3 Modelo De Datos

El desarrollo del modelo de datos a utilizar para el almacenamiento de la información según las utilidades de la misma, fueron realizados a partir del maquetado general según el diseño de UX. El almacenamiento de datos es realizado de forma local dentro del dispositivo y no se almacena en ningún otro sitio.

Dentro del modelo de datos para el desarrollo de la aplicación se destacan:

Planos cartesianos: Esta entidad almacena la información de los diferentes sistemas de coordenadas en la proyección plana cartesiana. Esta información es mostrada al usuario durante la creación del proyecto y es filtrada por departamentos y municipios. Se realizó de esta manera debido a que, su área de cobertura suele ser usada solamente a nivel municipal.

Tabla 4 Modelo de Datos Entidad Origen Plano Cartesiano

Descripción Tema	***Origen plano cartesiano:*** *Entidad que almacena la información de los diferentes sistemas de coordenadas en la proyección plana cartesiana.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
PK_ORIGENES_CART	Integer	20	Identificador del origen cartesiano
FK_MUNICIPIOS	String	250	Identificador de Municipio
NOMBRE	String	250	Nombre del Plano Cartesiano
LATITUD	Double	100	Latitud de Origen del plano Cartesiano
LONGITUD	Double	100	Longitud de Origen del plano Cartesiano
NORTE	Double	100	Coordenada asignada al falso Norte de la proyección
ESTE	Double	100	Coordenadas asignadas al falso Este de la Proyección
PLANO_PROY	Double	100	Altura del Plano de proyección del Plano Cartesiano

Origen Gauss: Esta entidad almacena los datos correspondientes a los seis orígenes Gauss usados en Colombia.

Tabla 5 Modelo de Datos Entidad Origen Gauss

Descripción Tema	***Entidad Origen Gauss:*** *Esta entidad almacena los datos correspondientes a los seis orígenes Gauss usados en Colombia*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
PK_ORIGENES_GAUSS	Integer	20	Identificador del origen Gauss
NOMBRE	String	250	Nombre de la Proyección Gauss
LATITUD	Double	100	Latitud de Origen Proyección Gauss
LONGITUD	Double	100	Longitud de Origen Proyección Gauss
NORTE	Double	100	Coordenada asignada al falso Norte de la proyección
ESTE	Double	100	Coordenadas asignadas al falso Este de la Proyección

Departamentos: Esta entidad almacena los nombres de los diferentes departamentos del país, y es usada principalmente como filtro para realizar la búsqueda de planos cartesianos.

Tabla 6 Modelo de Datos Entidad Departamento

Descripción Tema	***Departamentos:*** *Esta entidad almacena los nombres de los diferentes departamentos del país, y es usada principalmente como filtro para realizar la búsqueda de planos cartesianos.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
PK_DEPARTAMENTO	Integer	20	Identificador del departamento.
NOMBRE	String	250	Nombre del departamento.

Municipios: Esta entidad almacena el nombre de los municipios y el departamento del país al que pertenecen, con el fin de facilitar el filtrado y la búsqueda de planos cartesianos.

Tabla 7 Modelo de Datos Entidad Municipios

Descripción Tema	***Municipios:*** *Esta entidad almacena el nombre de los municipios y el departamento del país al que pertenecen, con el fin de facilitar el filtrado y la búsqueda de planos cartesianos.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
PK_MUNICIPIOS	Integer	20	Identificador del municipio
FK_DEPARTAMENTOS	Integer	20	Identificador del Departamento al cual pertenece el municipio
Nombre	String	250	Nombre del Municipio

Altura Niveladas: Almacena datos correspondientes a los puntos de la Red Pasiva del IGAC que han sido nivelados geométricamente. Son usados en los levantamientos topográficos, generalmente para la obtención de altura del nivel medio del mar.

Tabla 8 Modelo de Datos Entidad Altura Niveladas

Descripción Tema	***Alturas niveladas:*** *Almacena los datos correspondientes a los puntos de la Red Pasiva del IGAC que han sido nivelados geométricamente.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
NOMENCLATURA	String	250	Nombre del punto
MUNICIPIO	String	250	Nombre de Municipio
DEPARTAMENTO	String	250	Nombre de Departamento
ESTADO_PUN	String	250	Estado Físico del Punto
FECHA	Date	15	Año de Materialización del punto
LATITUD	Double	100	Latitud del Punto
LONGITUD	Double	100	Longitud del Punto
ALTURA_ELI	Double	100	Altura Elipsoidal del Punto
X	Double	100	Coordenada X Geocéntrica del Punto
Y	Double	100	Coordenada Y Geocéntrica del Punto
Z	Double	100	Coordenada Z Geocéntrica del Punto
Vx	Double	100	Velocidad en X del Punto
Vy	Double	100	Velocidad en Y del Punto
Vz	Double	100	Velocidad en Z del Punto
Ondulación	Double	100	Ondulación Geoidal del Punto
Z_Nivelada	Double	100	Altura del punto nivelado

GEORED: Esta entidad almacena las coordenadas y nombres de las estaciones pertenecientes a la red de rastreo continua del Servicio Geológico Colombiano.

Tabla 9 Modelo de Datos Entidad Estaciones GeoRed

Descripción Tema	***Estaciones GeoRed:*** *Esta entidad almacena las coordenadas y nombres de las estaciones pertenecientes a la red de rastreo continua del Servicio Geológico Colombiano.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
NOMBRE	String	250	Nombre del punto
LATITUD	Double	100	Latitud del Punto
LONGITUD	Double	100	Longitud del Punto

Magna ECO: Esta entidad almacena la información de nombre y coordenadas de las estaciones pertenecientes a la red Activa del IGAC MAGNA ECO.

Tabla 10 Modelo de datos Entidad Estaciones Magna Eco

Descripción Tema	***Estaciones Magna Eco:*** *Esta entidad almacena la información de nombre y coordenadas de las estaciones pertenecientes a la red Activa del IGAC MAGNA ECO.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
ID	Integer	20	Identificador de la Estación de rastreo continua
NOMBRE	String	250	Nombre del punto
LATITUD	Double	100	Latitud del Punto
LONGITUD	Double	100	Longitud del Punto

Red Pasiva IGAC: Esta entidad almacena los datos correspondientes a los vértices geodésicos que conforman la red pasiva del IGAC.

Tabla 11 Modelo de Datos Red Pasiva IGAC

Descripción Tema	***Red pasiva IGAC:*** *Esta entidad almacena los datos correspondientes a los vértices geodésicos que conforman la red pasiva del IGAC.*		
CAMPO	**TIPO DE DATO**	**TAMAÑO**	**DESCRIPCIÓN**
FID	integer	20	Identificador del Punto
NOMENCLATU	String	250	Nomenclatura del Punto
MUNICIPIO	String	250	Nombre del Municipio
DEPARTAMENTO	String	250	Nombre del Departamento
ESTADO_PUN	String	250	Estado del Punto
FECHA	Date	15	Año de Instalación
LATITUD	Double	100	Latitud del Punto
LONGITUD	Double	100	Longitud del Punto
ALTURA_ELI	Double	100	Altura Elipsoidal
X	Double	100	Coordenadas X geocéntrica del Punto
Y	Double	100	Coordenada Y geocéntrica del Punto
Z	Double	100	Coordenadas Z geocéntrica del Punto
Vx	Double	100	Velocidad en X del punto
Vy	Double	100	Velocidad en Y del Punto
Vz	Double	100	Velocidad en Z del Punto
Ondulación	Double	100	Ondulación Geoidal del Punto

4.3.4 Librerías Utilizadas

El desarrollo de la aplicación está fundamentado en la utilización de librerías disponibles dentro de la web oficial de Dart y Flutter, con las cuales se realizó la construcción de algunos elementos de la interfaz y el desarrollo de algunas funcionalidades de la aplicación.

Durante el desarrollo de la aplicación se utilizaron las siguientes librerías de código abierto y uso libre:

- SQFLITE 1.2.0: Librería utilizada para realizar las operaciones básicas dentro de la base de datos utilizada dentro de la aplicación, mediante la cual se realizan las operaciones básicas en la base de datos (CRUD).
- LATLONG 0.6.1: Librería utilizada para gestionar las coordenadas navegadas obtenidas de la antena GNSS del dispositivo móvil.
- FLUTTER_MAP_MARKER_CLUSTER ^0.2.7: Librería utilizada para realizar el agrupamiento de puntos dentro de los MapView que se encuentran dentro de la aplicación.
- FLUTTER_MAP ^0.8.2: Librería utilizada para generar los mapas de navegación que se tienen dentro de la aplicación.
- USER_LOCATION ^0.1.2: Librería utilizada para la obtención de posición del usuario en tiempo real.
- MAP_CONTROLLER ^0.7.1: Librería implementada para facilitar el uso de zoom y navegación dentro de los MapView.
- SHARE ^0.6.4+1: Librearía utilizada para compartir los datos de coordenadas convertidas.
- FILE_PICKER ^1.9.0+1: Librería utilizada para importar archivos CSV para convertir coordenadas.
- CSV ^4.0.3: Librería utilizada para leer y escribir archivos en formato CSV.
- ESYS_FLUTTER_SHARE ^1.0.2: Librería utilizada para compartir archivos.

4. ETAPA IV

4.2 Diagrama de Clases

En la siguiente ilustración, se aprecia el diagrama de las clases de la aplicación:

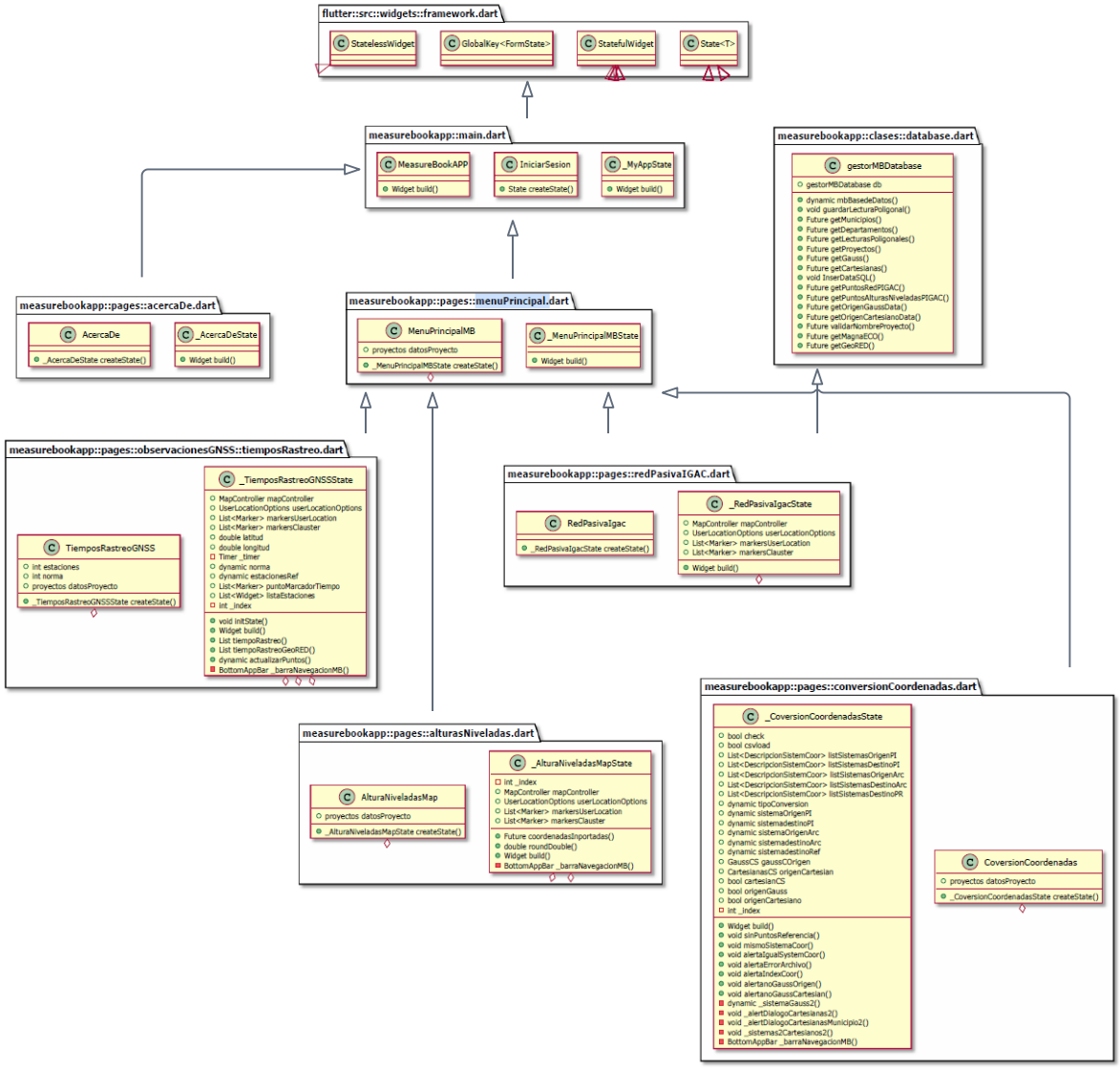


Ilustración 17 Diagrama de Clases Aplicación

Fuente: Elaboración Propia

La clase principal de la aplicación es el framework como tal, posteriormente se tiene la clase main.dart encargada de implementar la pantalla inicial, en donde la navegación permite realizar el acceso a las clases acercaDe.dart (Donde se muestran el nombre de los

integrantes del proyecto) y menuPrincipal.dart la cual permite acceder a los cuatro sub menus de la aplicación (alturasNiveladas.dart, TiemposRastreoGNSS.dart, ConversiónCoordenadas.dart y RedPasivaIgac.dart). Finalmente, se tiene la clase database.dart la cual se encarga de realizar la comunicación con la base de datos.

Dentro de las clases se tienen tres grupos (Interfaz, Interfaz – Lógica y Lógica), en las clases de tipo interfaz como su nombre lo indica, son clases que tiene como única funcionalidad mostrar al usuario información y permitir la navegación dentro de la aplicación teniendo dentro de este primer grupo las clases main.dart, acercaDe.dart y menuPrincipal.dart. Por otro lado, las clases del tipo Interfaz-Lógica además de mostrar al usuario la información que le permite hacer uso de la aplicación tienen métodos lógicos que dan funcionalidad a la aplicación, dentro de este se grupo se tienen las clases tiemposDeRastreo.dart, alturasNiveladas.dart, redPasivaIGAC.dart y conversionCoordenadas.dart. Finalmente, las clases del tipo lógica son aquellas en las que su funcionamiento no tiene ningún componente dentro de la interfaz, basando su uso netamente como backend. De las clases principales se tiene la clase database.dart encargada de gestionar la conexión con la base de datos.

Las clases principales de tipo lógico contienen relaciones con las clases secundarias, un ejemplo claro de esto son las relaciones que tiene la clase database.dart con los diferentes modelos de datos que le permiten parsear los datos de tipo texto plano(como resultado de una consulta) a objetos (Orígenes de Coordenadas Cartesianas).

4.1.1 framework.dart

En la siguiente ilustración se aprecian las clases principales usadas del framework para el desarrollo de la App:

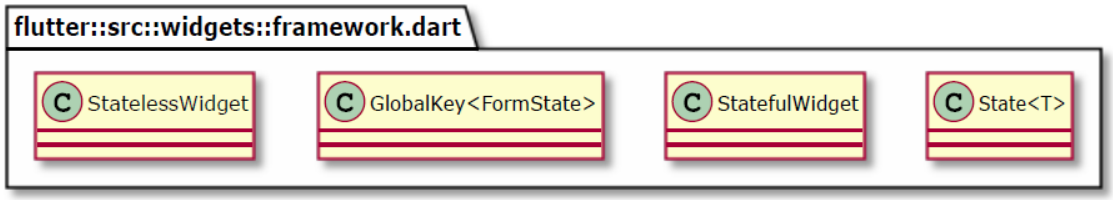


Ilustración 18 Modelo Clase FramerWork.dart

Fuente: Elaboración Propia

StatelessWidget: Esta clase permite la creación de interfaces de tipo estático, es decir, aquellas interfaces de usuario que no cambian, es usada para la creación de interfaces de información o navegación como es el caso de las clases acercaDe.dart, main.dart, menuPrincipal.dart.

GlobalKey: Esta clase es implementada para identificar el estado de cada uno de los elementos que componen la aplicación, como por ejemplo Swipers, TextDialog, Switch, etc.

StatefulWidget: Esta clase permite la creación de interfaces de tipo dinámico, que cambian en tiempo real con las acciones del usuario, el ejemplo más relevante dentro de la aplicación de la utilización de esta clase, son las clases que usan mapas como es el caso de la actividad creada para la navegación de puntos.

State: Esta clase se encarga de gestionar los estados de los diferentes componentes de la aplicación, como por ejemplo el cambio de coordenadas de origen y destino en la conversión de coordenadas, o la actualización de un mapa de navegación cuando el usuario solicita que se active la ubicación del mismo.

4.1.2 main.dart

En la siguiente ilustración se aprecia las clases y métodos de la clase principal main.dart

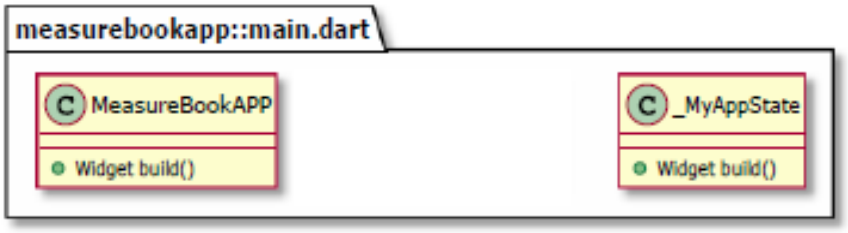


Ilustración 19 Modelo de Clase main.dart

Fuente: Elaboración Propia

MeasurebookApp: Es la clase encargada de construir la pantalla de inicio y es de uso netamente de UI.

_MyAppState: Es una clase privada encargada de gestionar los estados generales de la aplicación onStop(), onCreate(), onStart(), onResume(), onPause(), onDestroy().

4.1.3 acercaDe.dart

En la siguiente ilustración se aprecia la clase encargada de construir la pantalla de AcercaDe, donde se muestra la información básica de la aplicación.

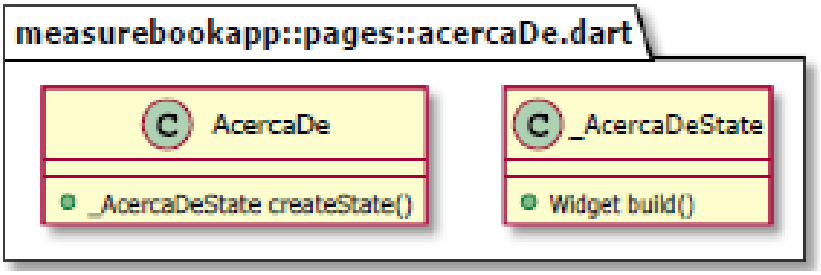


Ilustración 20 Modelo de Clase acercaDe.dart

Fuente: Elaboración Propia

_AcercaDeState createState(): Este método está encargado de la creación del estado en el que se encuentra la actividad, por ejemplo en estado onStop() que se encarga de mantener

la información cargada en la memoria RAM del dispositivo, cuando el usuario cambia de aplicación o bloquea el celular, para luego cuando se reanude la actividad no se tenga que volver a renderizar la interfaz y realizar la carga de datos.

__AcercaDeState Widget build(): Encarga de gestionar los diferentes recursos de la aplicación cuando está cambia de estado.

4.1.4 menuPrincipal.dart

En la siguiente ilustración se aprecia la estructura de la clase MenuPrincipal.

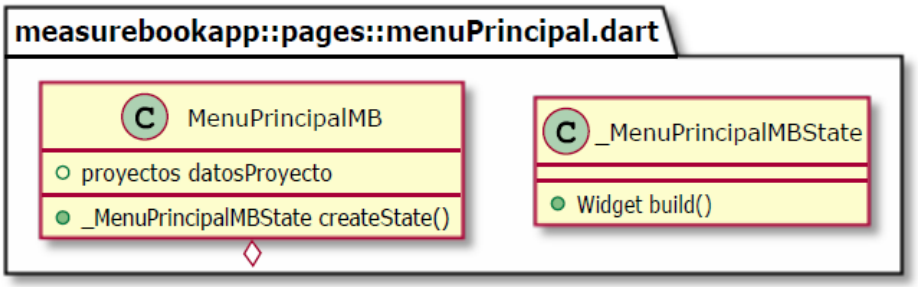


Ilustración 21 Modelo de Clase menuPrincipal.dart

Fuente: Elaboración Propia

_MenuPrincipalMBState CreateState(): Método encargado de gestionar el estado de la pantalla del menú principal.

_menuPrincipalMbState Widget build(): Método encargado de realizar el renderizado de los elementos de la interfaz según el estado en el que se encuentre la actividad.

4.1.5 database.dart

En la siguiente ilustración se aprecian las variables y composición de la clase encargada de gestionar la base de datos.

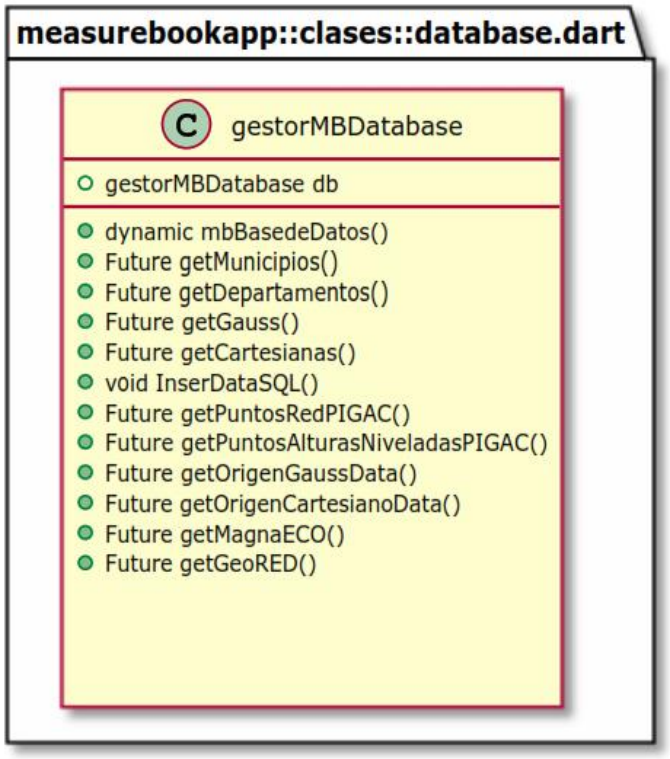


Ilustración 22 Modelo de Clase database.dart

Fuente: Elaboración Propia

mbBasedeDatos(): Método encargado de obtener el path de la base de datos y habilitar la lectura de la misma, es un método genérico que permite acceder a la data de la base de datos.

Future getMunicipios(): Es un método asincrónico encargado de obtener el listado de los municipios de un departamento determinado, retorna una lista de objeto que es usada para que el usuario elija el municipio en el cual desea consultar la existencia de orígenes de planos cartesianos.

Future getDepartamentos(): Es un método asincrónico encargado de obtener y parsear en un listado de objetos los departamentos del país, usados para la selección de sistemas de coordenadas planos cartesianos.

Future getGauss(): Este método asincrónico es el encargado de obtener los datos correspondientes a los orígenes Gauss Kruger del país, retornando un listado de objetos del tipo OrigenGauss.

Future getCartesianas(): Este método asincrónico es el encargado de obtener los datos correspondientes a los orígenes cartesianos disponibles en el municipio seleccionado por el usuario. Tiene como parámetros de entrada el municipio, a partir del cual se hace la consulta en la base de datos y se obtiene los datos del origen cartesiano, retornando como resultado un listado de objetos tipo OrigenCartesiano.

Void InsertSQL(): Método asincrónico encargado de realizar consultas, inserciones o borrado de información en la base de datos, recibiendo como un dato de entrada una cadena de texto con la consulta SQL correspondiente.

Future getPuntosRedPasivaIGAC(): Método asincrónico encargado de obtener el listado de los puntos de referencia almacenados en la base de datos, retorna un listado de objetos de tipo PuntosRedPasivaIgac.

Future getPuntosAlturasNiveladasIGAC(): Método asincrónico encargado de obtener el listado de los puntos con altura nivelada de la red Pasiva del IGAC, retornar un listado de objetos de Tipo PuntosAlturaNivelada.

Future getOrigenGaussData: Método asincrónico encargado de obtener los datos de falso norte, falso este, latitud de origen, longitud de Origen y factor de escala. Utilizando como dato de entrada el id del sistema de coordenadas asignado como llave principal dentro de la base de datos.

Future getOrigenesCartesianoData(): Método asincrónico encargado de obtener los datos de falso norte, falso este, latitud de origen, longitud de Origen y Altura del plano de proyección. Utilizando como dato de entrada el id del sistema de coordenadas almacenado como llave principal dentro de la base de datos.

Future getMagnaECO(): Método asincrónico encargado de obtener el listado de los puntos de referencia perteneciente a la red de rastreo continuo del IGAC, retornando un listado de objetos del tipo EstacionMagnaECO.

Future getGeoRED(): Método asincrónico encargado de obtener el listado de los puntos de referencia perteneciente a la red de estaciones continuas del Servicio Geológico Colombiano, retornando como resultado un listado de objetos del tipo EstacionesGEORED.

4.1.6 tiemposRastreo.dart

En la siguiente ilustración se aprecia el modelo de la clase tiempo de rastreo.

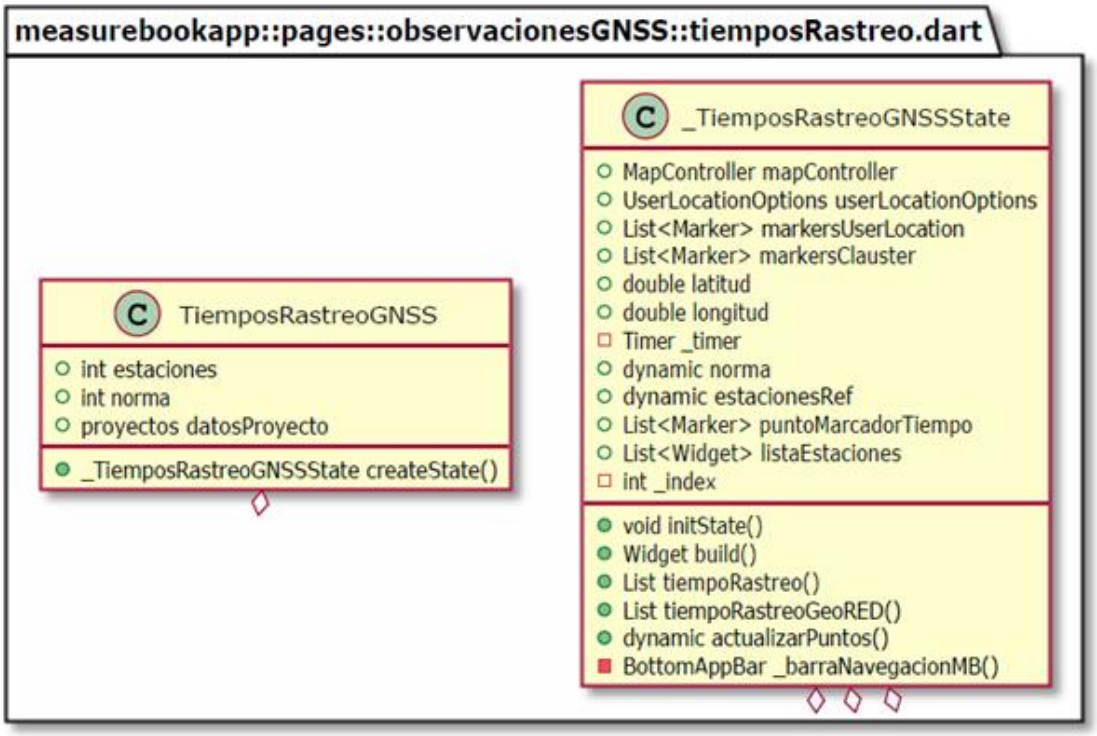


Ilustración 23 Modelo de Datos Clase Tiempos de Rastreo

Fuente: Elaboración Propia

_TiemposRastreoGNSSState createState(): Método encargado de gestionar el estado en el que se encuentra la actividad

Void initState(): Método encargado de generar el renderizado de los elementos visibles para el usuario en el estado onStart() de la actividad.

WidgetBuild(): Método encargado de construir los elementos de la actividad según el estado en el que se encuentre la misma, es el encargado de construir los cambios en los widgets, que realiza el usuario al interactuar con la aplicación.

List tiempoRastreo(): Método encargado de realizar la consulta a la base de datos de las estaciones de rastreo de la red seleccionada por el usuario, y posteriormente calcular la distancia mediante el problema geodésico directo entre las estaciones y la ubicación establecida por el usuario, para posteriormente organizar una lista de las estaciones más cercanas y cambiar el estado de la actividad generando la visualización de los datos de tiempo de rastreo.

Dynamic ActualizarPuntos(): Método encargado de actualizar los datos que se le muestran al usuario de los tiempos de rastreo cuando este cambia la ubicación del punto de cálculo, este método se encarga de ejecutar nuevamente el método tiempoRastreo() y actualizar los datos en la UI.

4.1.7 redPasivaIGAC.dart

En la siguiente la ilustración se aprecia el modelo de la clase Red Pasiva IGAC.

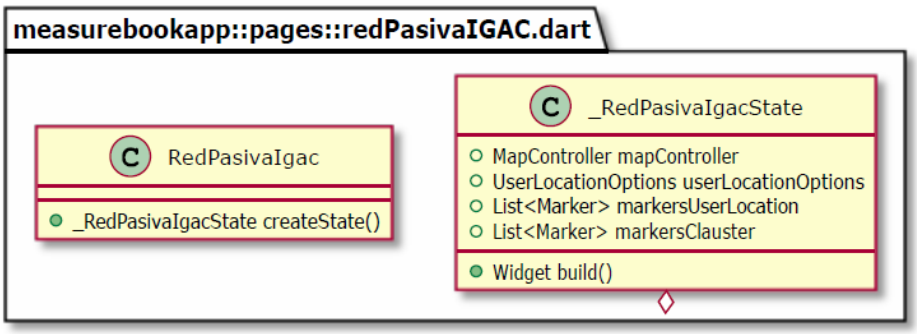


Ilustración 24 Modelo de clase redPasivaIGAC.dart

Fuente: Elaboración Propia

_RedPasivaIgacState createState(): Método encargado de gestionar el estado en el que se encuentra la actividad.

WidgetBuild(): Método encargado de construir los elementos de la actividad según el estado en el que se encuentre la misma, es el encargado de construir los cambios en la visualización en el mapa y de las interacciones del usuario con los puntos de referencia.

4.1.8 conversionPuntoIndividual.dart

En la siguiente ilustración se aprecia el modelo de clases usado para la conversión de un punto individual.

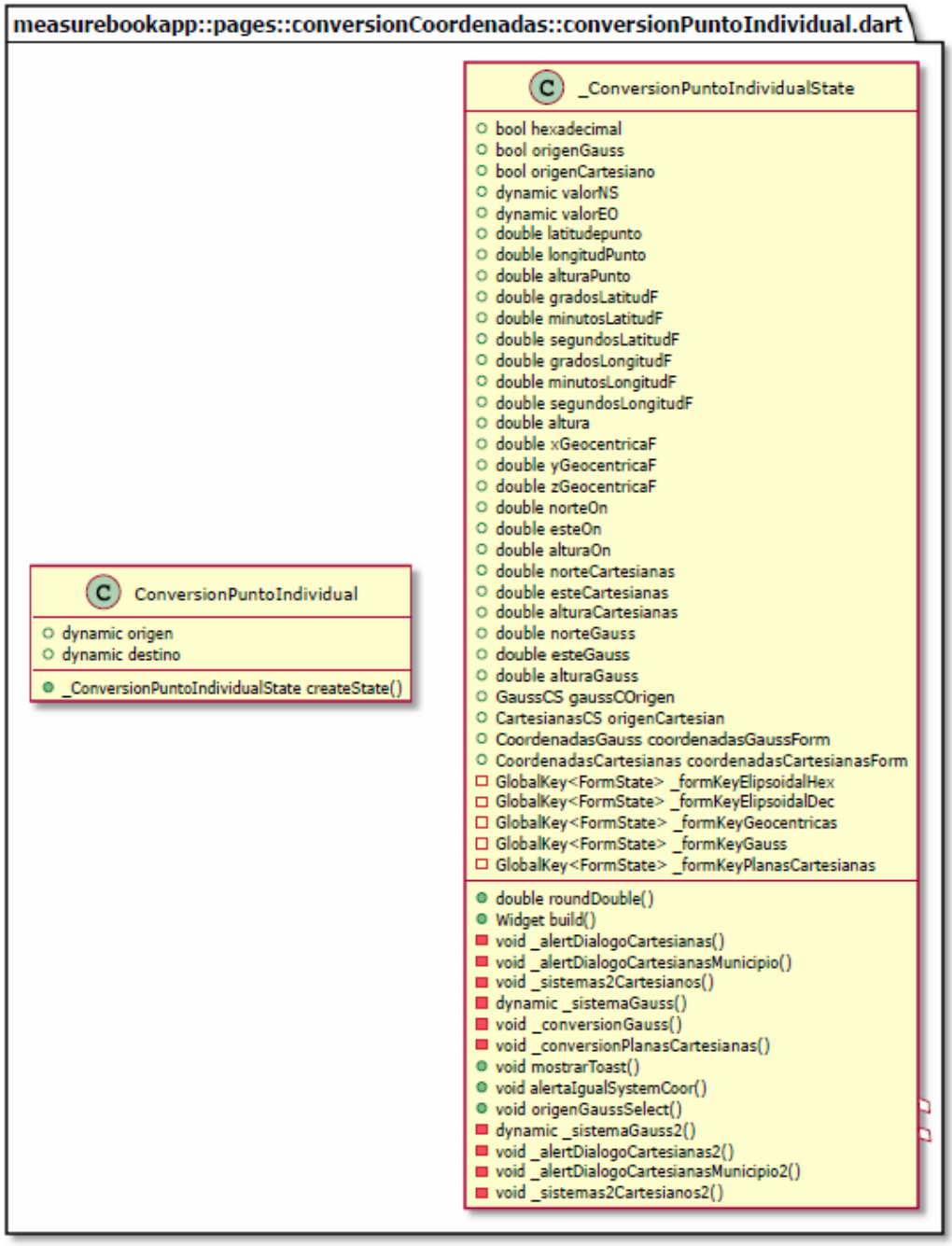


Ilustración 25 Modelo de Datos conversionPuntoIndividual.dart

Fuente: Elaboración Propia

roundDouble(): Función encargada de redondear a tres cifras significativas las coordenadas planas, geocéntricas y a 8 las elipsoidales.

WidgetBuild(): Método encargado de construir los elementos de la actividad según el estado en el que se encuentre la misma, es el encargado de construir los cambios en la visualización de sistemas de coordenadas de origen y destino.

Void _alertDialogCartesianas(): Método encargado de mostrar al usuario el listado de planos cartesianos disponibles para el municipio seleccionado.

Void _alertDialogCartesianaMunicipio(): Método encargado de mostrar al usuario el listado de departamento y municipios, para facilitar la selección del origen del plano cartesiano que desee utilizar para realizar la conversión de las coordenadas.

Void _conversionGauss(): Método encargado de mostrar al usuario los diferentes orígenes Gauss disponibles de forma oficial para Colombia y seleccionar el que desea utilizar para realizar la conversión de las coordenadas.

Void _mostrarToast(): Método encargado de mostrar alertas al usuario sobre errores que esté realice al ingresar coordenadas o convertirlas.

Void _alertaIgualSystemCoor(): Método encargado de mostrar una alerta al usuario cuando seleccionar el mismo sistema de coordenadas de partida y destino, para realizar la conversión de las coordenadas.

4.1.9 conversionArchivo.dart

En la siguiente ilustración se aprecia el esquema de la clase usada para la conversión de coordenadas mediante la importación de un archivo CSV.

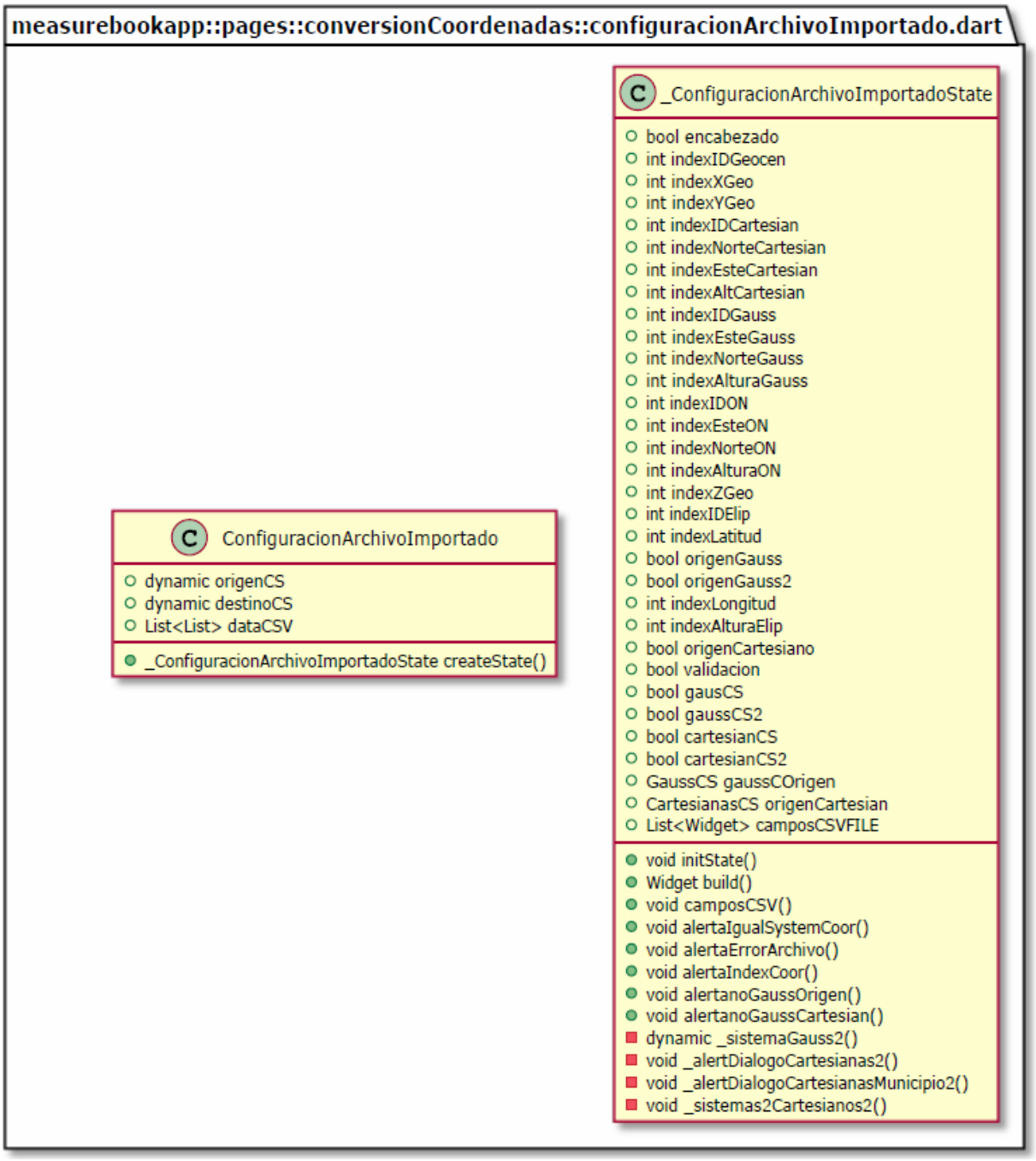


Ilustración 26 Modelo de ClaseconversionArchivo.dart

Fuente: Elaboración Propia

Void initState(): Método encargado de renderizar y realizar la carga de los elemento de la UI, para la visualización del usuario de las opciones para realizar la carga del archivo CSV para la conversión de las coordenadas.

WidgetBuild(): Método encargado de construir los elementos de la actividad según el estado en el que se encuentre la misma, es el encargado de construir los cambios en la visualización de sistemas de coordenadas de origen y destino.

Void camposCSV(): Método encargado de leer el encabezado del archivo CSV cargado para que el usuario pueda relacionar las columnas del CSV con los componentes de las coordenadas que importa.

Void _alertaIgualSystemCoor(): Método encargado de mostrar una alerta al usuario cuando selecciona el mismo sistema de coordenadas de partida y destino.

Void alertaErrorArchivo(): Método encargado de mostrar un alerta al usuario cuando el CSV no se encuentre estructurado de la forma correcta (Símbolo decimal punto y separador de columnas coma).

Void alertaIndexCoor(): Método encargado de mostrar una alerta al usuario cuando tenga dos columnas asignadas a la misma componente de la coordenada, es decir cuando el usuario asigna por ejemplo la misma columna a los componentes norte y este de una coordenadas plana.

Void AlertaGaussOrigen(): Método encargado de mostrar al usuario los orígenes Gauss disponibles para realizar la conversión de coordenadas.

Void AlertaGaussCartesiano(): Método encargado de mostrar los orígenes gauss y orígenes cartesianos, cuando se realiza la conversión entre estos sistemas.

4.1.10 Clases de uso genérico

AltNiveladas.dart: Clase encargada de estructurar los datos de los puntos que contienen altura niveladas en objetos, resultantes de la consulta en la base de datos.

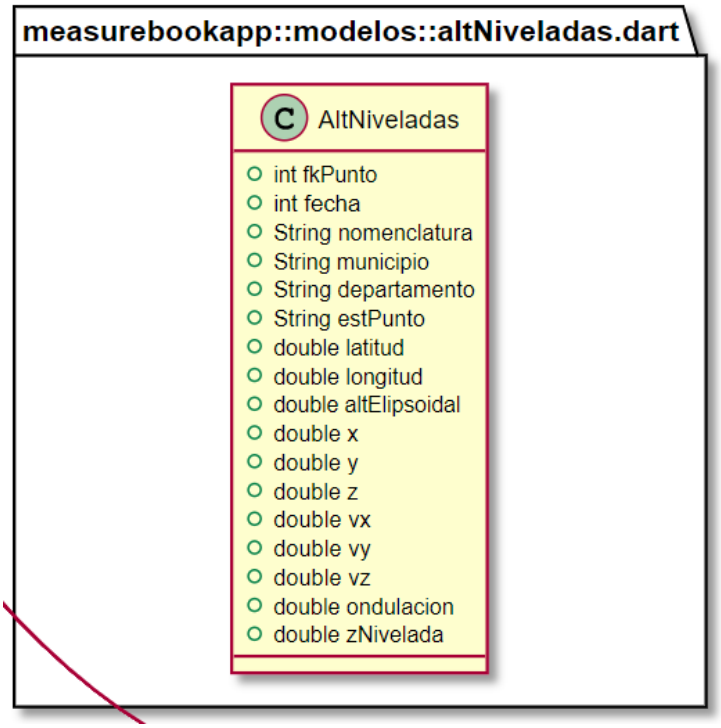


Ilustración 27 Modelo de clase AltNiveladas.dart

Fuente: Elaboración Propia

CoordeandasElipsoidales.dart: Clase encargada de estructurar los datos de los puntos que se encuentran en el sistema de coordenadas Elipsoidal, es usado en la conversión de coordenadas y en la navegación de puntos.

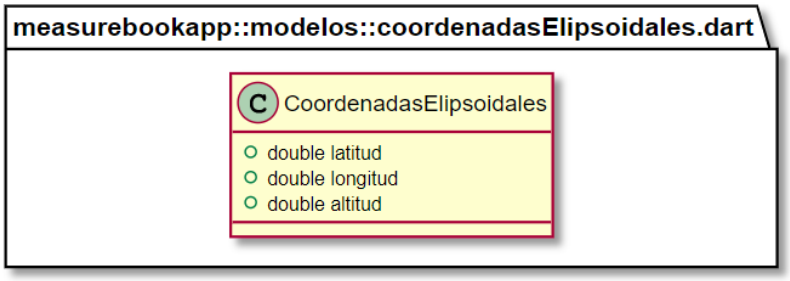


Ilustración 28 Modelo de Clase coordenadasElipsoidales.dart

Fuente: Elaboración Propia

CoordenadasGeocentricas: Clase encargada de estructurar los datos de los puntos que se encuentran en el sistema de coordenadas Geocéntrico, es usado en la conversión de coordenadas.

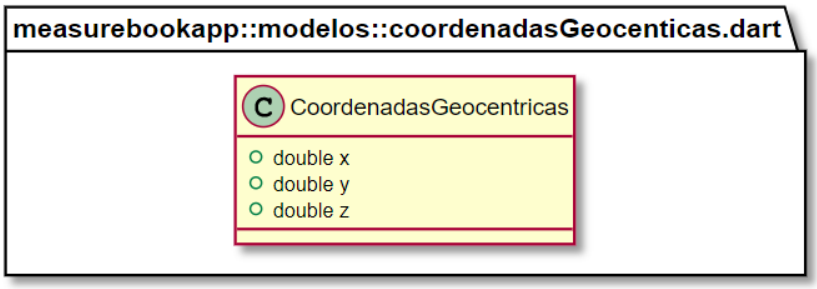


Ilustración 29 Modelo de clase CoordenadasGeocentricas.dart

Fuente: Elaboración Propia

CoordenadasON: Clase encargada de estructurar los datos de los puntos que se encuentran en el sistema de coordenadas Magna Origen Nacional, es usado en la conversión de coordenadas.

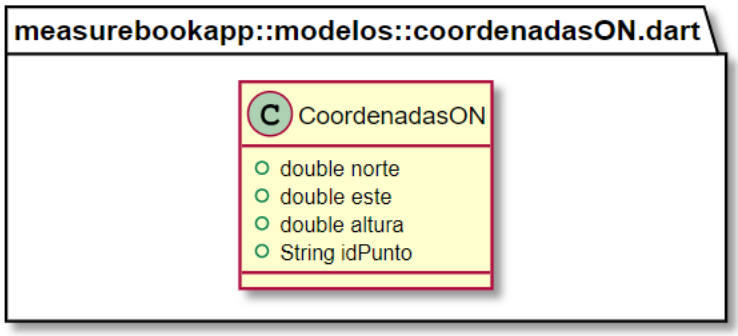


Ilustración 30 modelo de Clase coordenadasON.dart

Fuente: Elaboración Propia

CPlanasGenerico.dart: Clase encargada de estructurar de todos los sistemas de coordenadas planos proyectados usados en la aplicación, es usado en la conversión de coordenadas de tipo plano como es el caso de las coordenadas Gauss Krüger, Planas Cartesianas y Transversal de Mercator.

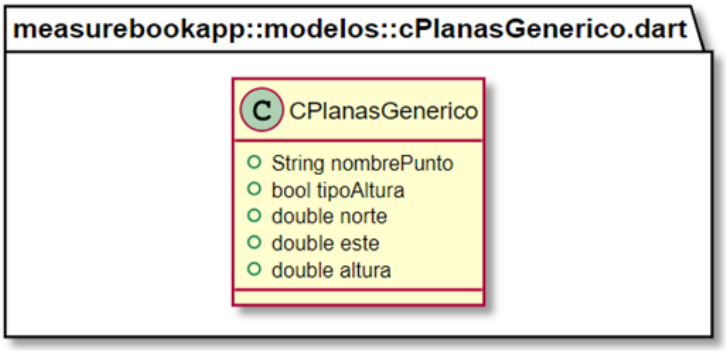


Ilustración 31 modelo de Clase CPlanasGenerico.dart

Fuente: Elaboración Propia

Estaciones.dart: Clase encarga de estructurar los datos obtenidos de la consulta en la base de datos de las estaciones de rastreo continuo de las redes MAGNA ECO y GEORED.

OrigenesCartesianos.dart: Clase encargada de estructurar los datos obtenidos de la consulta en la base de datos de los orígenes de planos cartesianos.

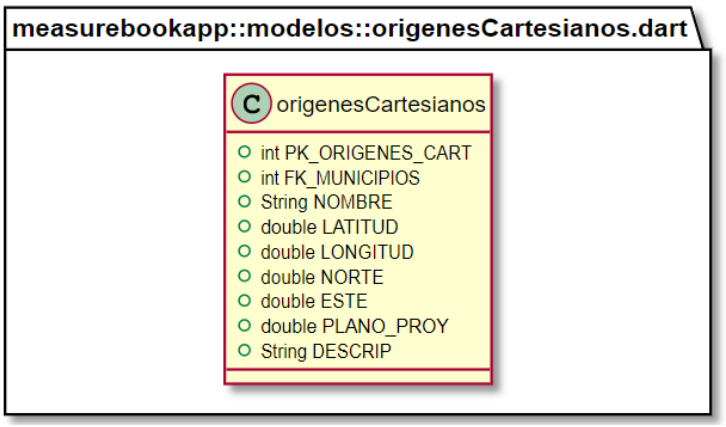


Ilustración 32 modelo de Clase origenesCartesianos.dart

Fuente: Elaboración Propia

OrigenesGauss.dart: Clase encargada de estructura los datos obtenidos de la consulta en la base de datos de los orígenes Gauss Krüger disponibles para Colombia.

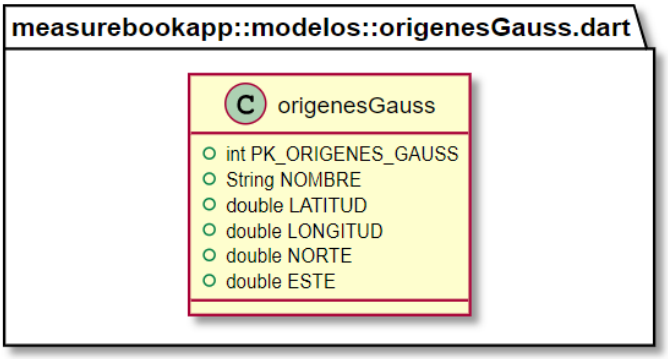


Ilustración 33 modelo de clase origenesGauss.dart

Fuente: Elaboración Propia

RedPasivaIGACPuntos.dart: Clase encargada de estructurar los datos obtenidos de la consulta en la base de datos de los puntos pertenecientes a la red Pasiva del IGAC.

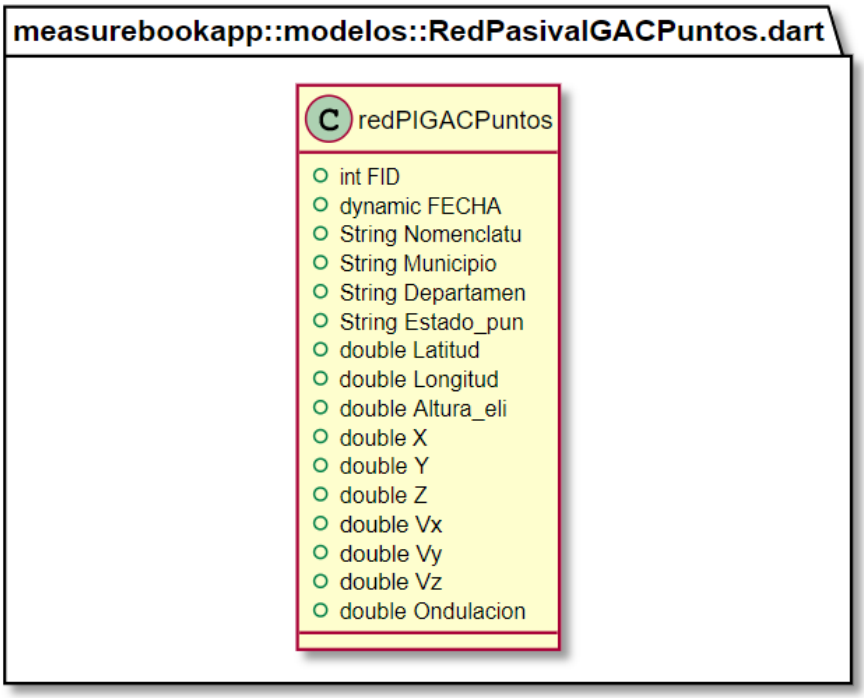


Ilustración 34 modelo de clase RedPIGACPuntos.dart

Fuente: Elaboración Propia

4.2 Codificación

En esta etapa del proyecto se describen las actividades desarrolladas, con el fin de generar las funcionales establecidas inicialmente en los requerimientos. Para lo cual, se utilizó el

framework de desarrollo multiplataforma flutter y el editor de código Visual Studio Code, con los cuales y junto con la instalación de las diferentes dependencias se realizó el desarrollo de la aplicación.

El desarrollo de la aplicación se inició con la interfaz de la aplicación, para lo cual se utilizaron varios widgets prefabricados que se encuentran dentro del framework de trabajo y facilitan de manera significativa el desarrollo. Las opciones de desarrollo tienen dos grandes estilos: Material Desing y Cupertino. Según los criterios de diseño y preferencias de los programadores, se eligieron los iconos y estilos contenidos dentro de Material Desing debido a que son los estilos correspondientes para aplicaciones Android.

Esta librería se encuentra dentro de las librerías nativas del Framework de desarrollo y su utilización se fundamenta en la utilización de widgets prefabricados de uso genérico dentro de las aplicaciones Android, estos widgets son personalizables y adicionalmente el framework permite el desarrollo de nuevos elementos.

El desarrollo fue realizado mediante la implementación de cada una de las funciones que componen la aplicación, para este fin se realizó la programación de cada una de las funcionalidades, iniciando con la navegación de la red de puntos pertenecientes a la red pasiva del IGAC.

4.2.1 Implementación módulo de Navegación Red Pasiva IGAC

La implementación de la navegación de los puntos de la red Pasiva del IGAC se inició con la implementación de la persistencia de los datos dentro de la aplicación. Para lo cual, se utilizó una base de datos en SQLite y la librería Sqflite, con las cuales se procedió a cargar la información de los puntos de la red pasiva. Los datos fueron obtenidos de la página de datos abiertos del IGAC.

FID	Nomenclatu	Municipio	Departamen	Estado_pun	FECHA	Latitud	Longitud	Altura_eli	X	Y	Z	Vx	Vy
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
0	B5-TL-1	ESPINAL	TOLIMA	Materializado	2009	4.165050473	-74.91806242	359.211	1655331.494	-6142629.924	460176.1421	0.0011	0.001
1	(EA)-MACARE...	BOGOTÁ D.C.	CUNDINAMAR...	Materializado	0	4.614969955	-74.0628738	2725.381	1746429.572	-6115845.096	509975.6901	0.0007	0.001
2	8372001	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.81673493	-75.16494449	8.331	1604216.765	-6056709.725	1189102.325	0.0145	0.004
3	8372002	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.82570576	-75.15684351	16.611	1605027.459	-6056310.626	1190078.53	0.0145	0.004
4	8372003	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.82967944	-75.14495881	4.708	1606259.501	-6055886.458	1190508.013	0.0144	0.004
5	8372004	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.85174439	-75.12702984	2.946	1608036.158	-6054938.196	1192904.809	0.0143	0.004
6	8372005	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.86951258	-75.08991161	4.525	1611863.537	-6053538.826	1194835.309	0.0142	0.004
7	8372006	JUAN DE ACO...	ATLÁNTICO	Materializado	2014	10.87792987	-75.07728082	7.125	1613153.421	-6053016.095	1195750.151	0.0141	0.004
8	13657016	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.923429605	-75.10381635	188.795	1615295.587	-6072354.794	1091939.509	0.0149	0.005
9	13657017	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.944515089	-75.09040469	183.465	1616612.073	-6071582.647	1094235.9	0.0149	0.005
10	13657018	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.947488122	-75.08909648	174.374	1616733.783	-6071482.195	1094558.236	0.0149	0.005
11	13657019	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.952242181	-75.08736501	166.569	1616891.901	-6071338.106	1095074.827	0.0149	0.005
12	13657020	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.961144824	-75.10372473	185.024	1615119.183	-6071652.558	1096047.913	0.0149	0.005
13	13657021	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.957217874	-75.11221178	204.3	1614243.968	-6071982.681	1095623.428	0.0149	0.005
14	13657022	SAN JUAN NE...	BOLÍVAR	Materializado	2010	9.940039783	-75.12676019	244.487	1612796.554	-6072747.899	1093758.851	0.0149	0.005
15	20228014	CURUMANÍ	CESAR	Materializado	2013	9.188888887	-73.65207054	27.781	1772372.121	-6042272.206	1011795.549	0.0108	0.004
16	44001033	RIOHACHA	LA GUAJIRA	Materializado	2013	11.26497307	-73.15023448	8.155	1813402.844	-5987479.836	1237766.382	0.0112	0.004
17	44001035	RIOHACHA	LA GUAJIRA	Materializado	2013	11.2714731	-73.12402928	11.245	1816101.223	-5986518.273	1238472.139	0.0112	0.004
18	44001036	RIOHACHA	LA GUAJIRA	Materializado	2013	11.28131085	-73.11999504	14.238	1816461.798	-5986189.568	1239539.939	0.0112	0.004
19	44001037	RIOHACHA	LA GUAJIRA	Materializado	2011	11.29216544	-73.11596429	19.216	1816816.086	-5985841.567	1240718.395	0.0112	0.004
20	44001038	RIOHACHA	LA GUAJIRA	Materializado	2013	11.30000207	-73.11279103	10.187	1817095.705	-5985569.984	1241566.699	0.0112	0.004
21	44001039	RIOHACHA	LA GUAJIRA	Materializado	2013	11.30870582	-73.10924566	7.523	1817410.486	-5985274.45	1242510.282	0.0112	0.004
22	44001040	RIOHACHA	LA GUAJIRA	Materializado	2013	11.32731285	-73.10189551	10.683	1818061.775	-5984657.719	1244529.133	0.0112	0.004
23	44001041	RIOHACHA	LA GUAJIRA	Materializado	2013	11.33557265	-73.09862218	5.634	1818350.047	-5984377.338	1245424.008	0.0112	0.004
24	44001042	RIOHACHA	LA GUAJIRA	Materializado	2013	11.36549818	-73.08679234	-4.051	1819393.302	-5983369.371	1248667.638	0.0113	0.004
25	44001043	RIOHACHA	LA GUAJIRA	Materializado	2013	11.37355735	-73.08298974	-6.924	1819738.442	-5983077.775	1249541.062	0.0113	0.004
26	44001044	RIOHACHA	LA GUAJIRA	Materializado	2013	11.38468768	-73.0763731	-8.635	1820358.168	-5982633.585	1250747.731	0.0113	0.004
27	44001045	RIOHACHA	LA GUAJIRA	Materializado	2013	11.39415792	-73.07046466	-3.275	1820916.388	-5982252.955	1251775.734	0.0113	0.004
28	44001046	RIOHACHA	LA GUAJIRA	Materializado	2013	11.40435705	-73.06520022	-6.826	1821400.075	-5981868.977	1252880.98	0.0113	0.004

Ilustración 35 Muestra de Datos red Pasiva del IGAC

Fuente: (Instituto Geográfico Agustín Codazzi IGAC, 2020)

Realizada la implementación de la base de datos de forma local dentro de la aplicación, se procedió a generar los objetos para realizar el despliegue de los datos.

```
class redPIGACPuntos{
  int FID;
  dynamic FECHA;
  String Nomenclatu, Municipio, Departamen,Estado_pun;
  double Latitud, Longitud, Altura_eli, X, Y, Z, Vx, Vy, Vz,Ondulacion;
  redPIGACPuntos({this.FID, this.Nomenclatu, this.Municipio, this.Departamen,
this.Estado_pun, this.FECHA, this.Latitud, this.Longitud, this.Altura_eli, this.X, this.Y,
this.Z, this.Vx, this.Vy, this.Vz, this.Ondulacion});
  factory redPIGACPuntos.fromMap(Map<String, dynamic> json) => new redPIGACPuntos(
      FID: json["FID"],
      Nomenclatu: json["Nomenclatu"],
      Municipio: json["Municipio"],
      Departamen: json["Departamen"],
      Estado_pun: json["Estado_pun"],
      FECHA: json["FECHA"],
      Latitud: json["Latitud"],
      Longitud: json["Longitud"],
      Altura_eli: json["Altura_eli"],
      X: json["X"],
      Y: json["Y"],
      Z: json["Z"],
      Vx: json["Vx"],
      Vy: json["Vy"],
      Vz: json["Vz"],
      Ondulacion: json["Ondulacion"]
  );
}
```

El modelo datos está diseñado para generar objetos a partir de la consulta de los datos en la base de datos. La consulta se realiza mediante una función asincrónica dentro de la base datos retornando como resultado una lista de objetos. En este caso, una capa de puntos de la red pasiva, que posteriormente serán estructurados como entidades de geometría tipo punto para su representación dentro del MapView.

```
Future<List<redPIGACPuntos>> getPuntosRedPIGAC() async {
  WidgetsFlutterBinding.ensureInitialized();
  Database db = await mbBasedeDatos();
  var response = await db.rawQuery('SELECT * FROM RED_PASIVA_IGAC');
  List<redPIGACPuntos> listaPuntosRedPasivaIGAC =
      response.map((c) => redPIGACPuntos.fromMap(c)).toList();
  return listaPuntosRedPasivaIGAC;
}
```

Obtenidos los datos de forma estructurada para su representación dentro de un MapView, se procedió a generar la vista de navegación de los puntos. Por lo cual, se utilizó la librería flutter_map, en su documentación oficial expone ejemplos de cómo representar en el MapView diferentes geometrías con variados tipos de datos. Para este caso específico, se creó un MapView dentro de un Scaffold mediante una función asincrónica que realiza la consulta en la base de datos de los puntos de la red pasiva, y los almacena dentro de una lista de objetos tipo Marker. Estos datos son los utilizados para realizar la representación de puntos dentro del MapView, teniendo la facilidad de implementar acciones dentro del método onTap(), en donde se implementó una ventana flotante que muestra toda la información del punto seleccionado.

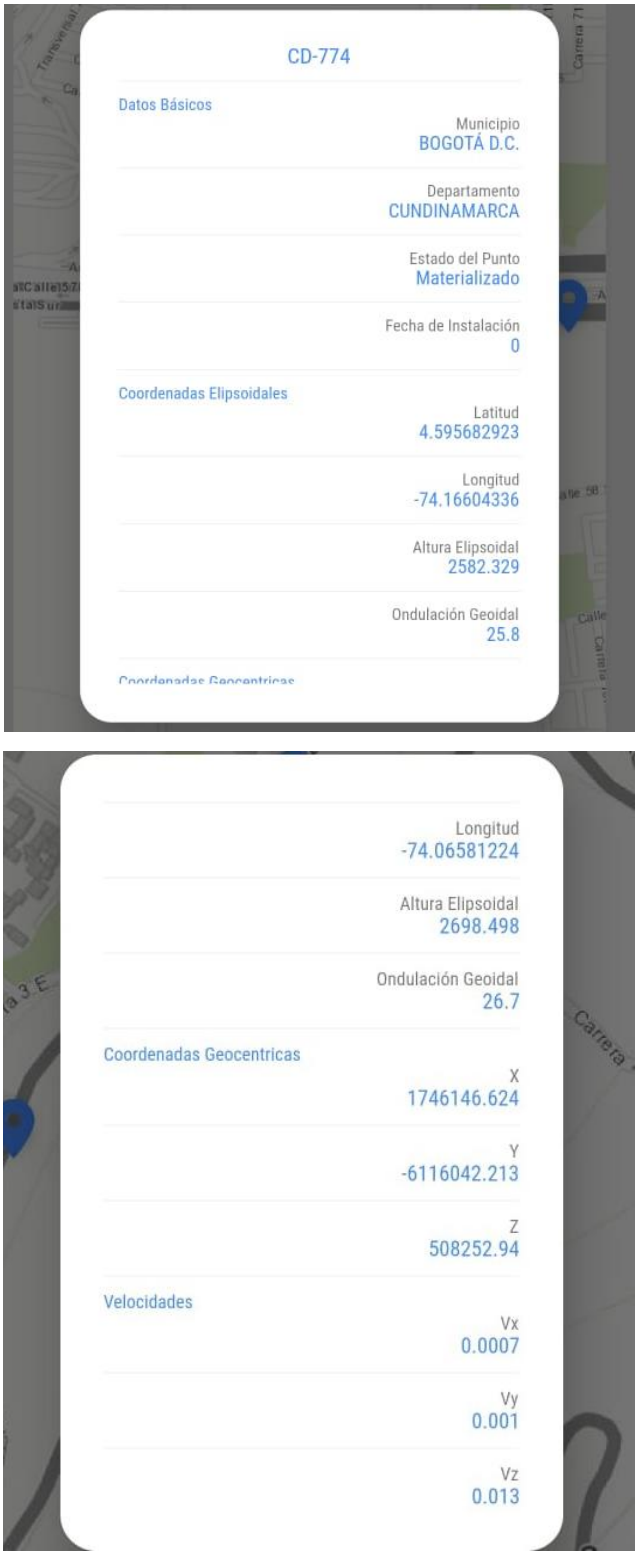


Ilustración 36 Información Punto de Referencia Red Pasiva IGAC

Fuente: Elaboración Propia

Adicionalmente, se agrega mediante servicio WMS el mapa base topográfico de ESRI con el fin de facilitar la navegación de los puntos; siempre y cuando el usuario tenga acceso a internet. En caso contrario, no se visualizará el mapa base, pero si la ubicación de los puntos.

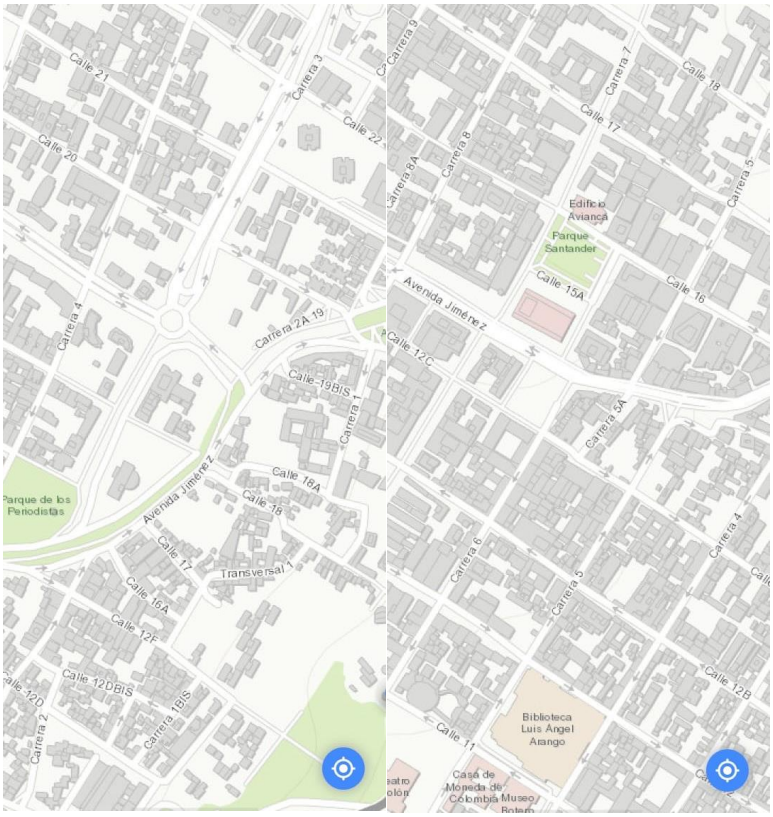


Ilustración 37 Mapa Base Navegación

Fuente: Elaboración Propia

Del mismo modo, se usa la librería Flutter_Map_Clauster para realizar la agrupación de los puntos con el fin de facilitar la navegación de los mismos, ya que al tener tantos registros no es posible visualizar el mapa si no realiza este tipo de agrupamiento.

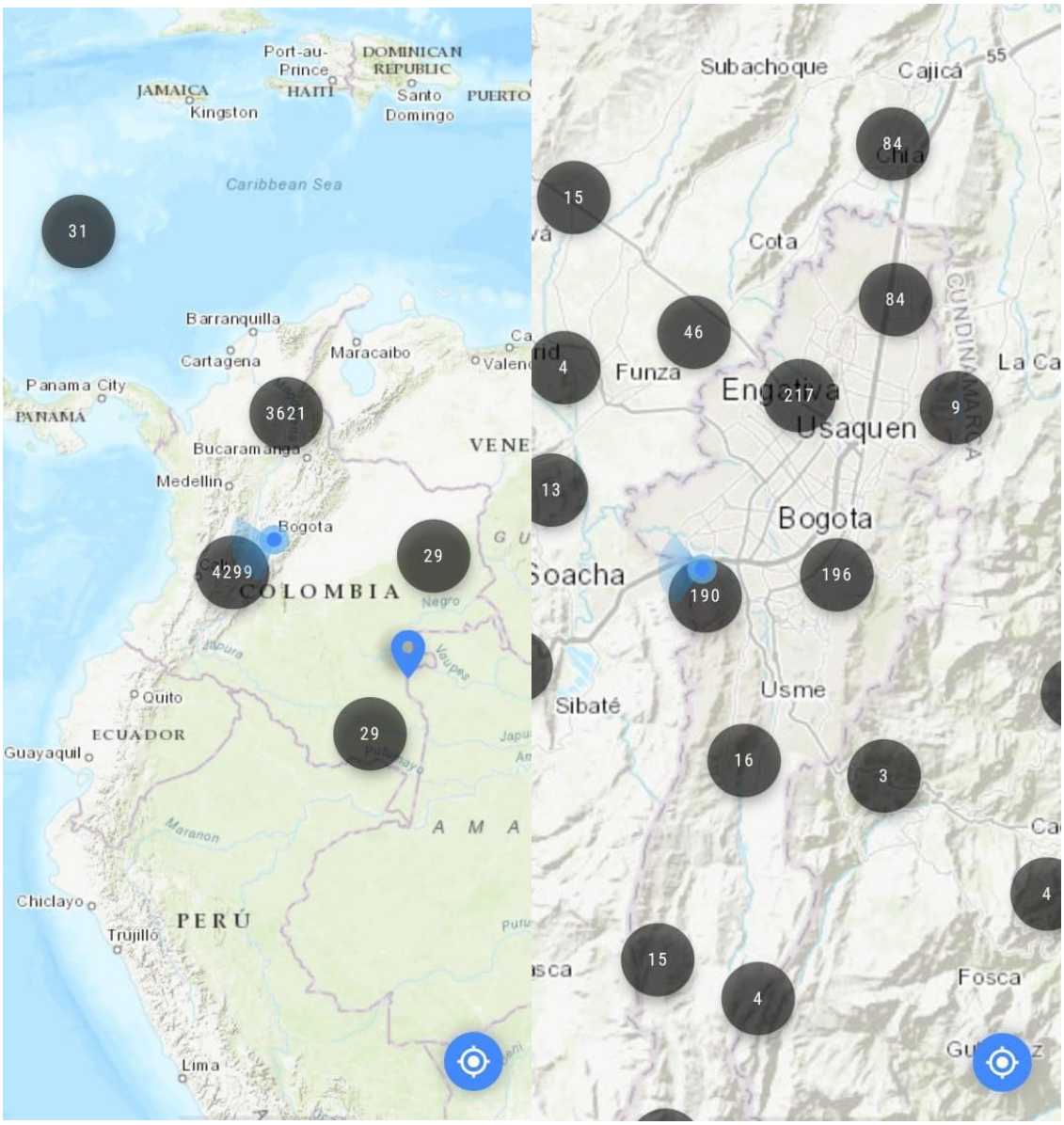


Ilustración 38 clúster de Puntos Red Pasiva IGAC

Fuente: Elaboración Propia

Finalmente, se realiza la implementación de un botón para acceder a la ubicación del usuario, el cual tiene como fin hacer un acercamiento a la posición del usuario, mostrando los puntos más cercanos a este.

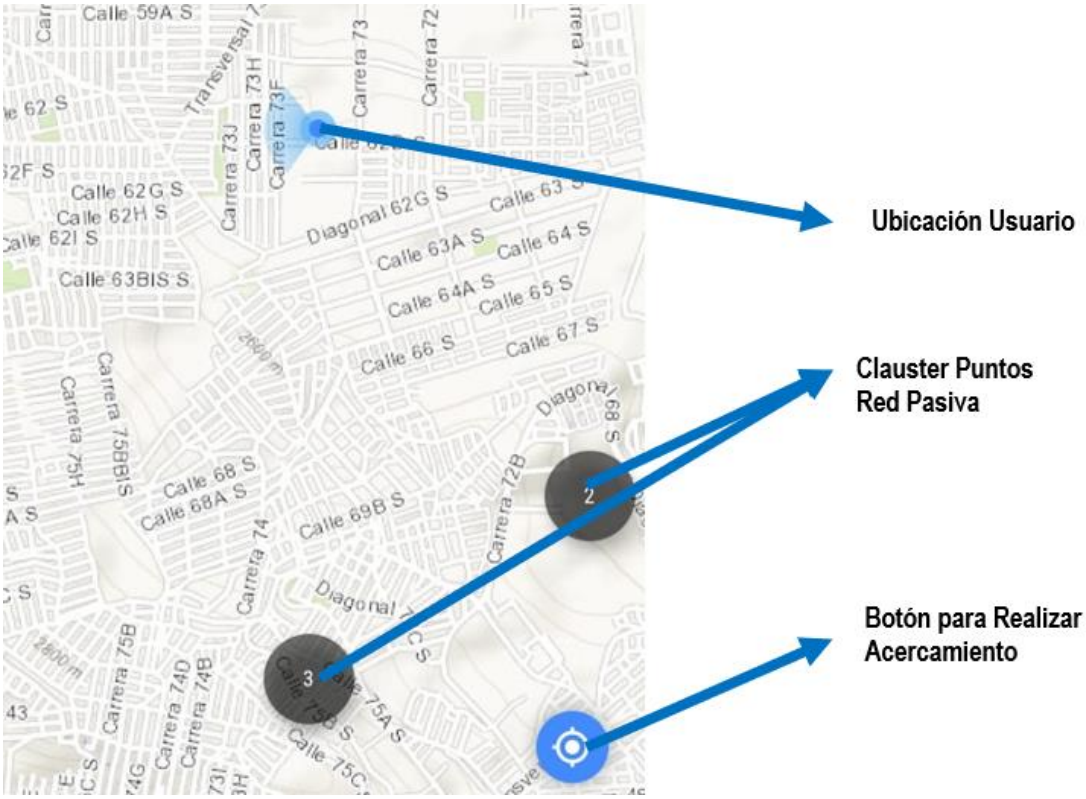


Ilustración 39 Ubicación Usuario Navegación Red Pasiva

Fuente: Elaboración Propia

El funcionamiento de la actividad está fundamentado por estados; los cuales se verán reflejados en la interfaz del usuario según la etapa del proceso en el que se encuentre en la navegación de la aplicación:

- Inicio: El estado inicial hace referencia al estado generado por la aplicación cuando el usuario ingresa a la actividad de navegación de los puntos. Iniciando con la verificación de permisos de ubicación y el acceso a memoria e Internet. Cuando la aplicación logra verificar estos permisos procede a generar el estado de carga de datos.
- Carga de datos: En este estado, la aplicación muestra en pantalla un indicador de carga circular, mientras realiza la consulta a la base de datos de los puntos de la Red Pasiva y genera los objetos de tipo Marker para ser representados en el MapView. Una vez realizada la carga de información, procede a generar el estado de despliegue de los datos.

- Despliegue de Datos: En este estado, la aplicación mostrará el mapa base y los clúster de los puntos disponibles, así como las opciones de ubicación del usuario y el funcionamiento del método onTap().

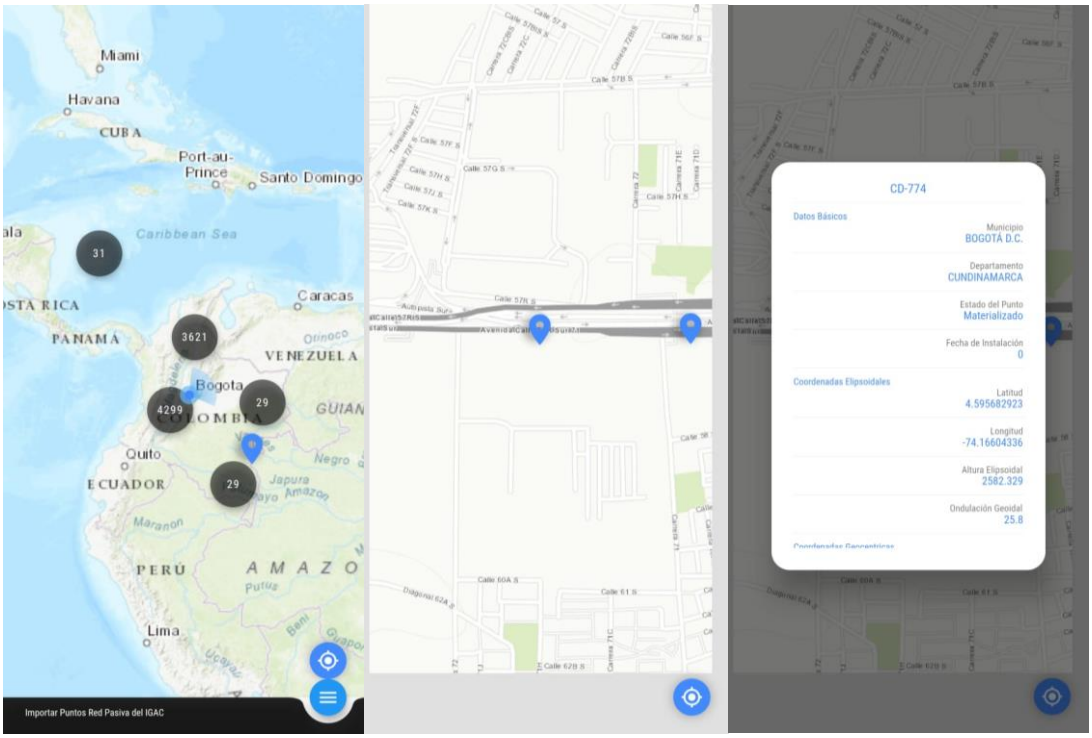


Ilustración 40 Navegación Red Pasiva del IGAC

Fuente: Elaboración Propia

Finalmente, el usuario está en la capacidad de obtener su ubicación mediante la utilización del GPS interno del celular. Esta acción activará un indicador de la posición del usuario y de forma automática se realizará un zoom in a la ubicación del mismo, con el fin de facilitar la búsqueda de puntos que se encuentren cercanos a él. Adicionalmente, si el usuario hace tap en un punto podrá visualizar la información atributiva que lo constituye.

4.2.2 Implementación módulo para el cálculo de Tiempos de Rastreo

El módulo de cálculo de tiempos de rastreo está fundamentado en el cálculo de la distancia geodésica entre el punto donde el usuario planea realizar la observación GNSS y las

estaciones de referencia más cercanas, para este fin se desarrolló la siguiente UX.



Ilustración 41 Selección de Red y Norma de Referencia Cálculo de Tiempos de Rastreo

Fuente: Elaboración Propia

Se desarrollo una primera actividad, en la cual se le da escoger al usuario la red de estaciones que desea consultar. Existen dos posibles opciones; la red Magna ECO del IGAC y GeoRED del Servicio Geológico Colombiano. Para la selección de las normas de referencia, se establecieron inicialmente tres normas; resolución IGAC 643 del 30/05/2018, normas IDU para la elaboración de estudios topográficos y la norma NS-030 de la Empresa de Acueducto y Alcantarillado de Bogotá. Una vez el usuario seleccione la red de estaciones de referencia y la norma con la cual desea calcular los tiempos de rastreo se mostrar la segunda actividad.

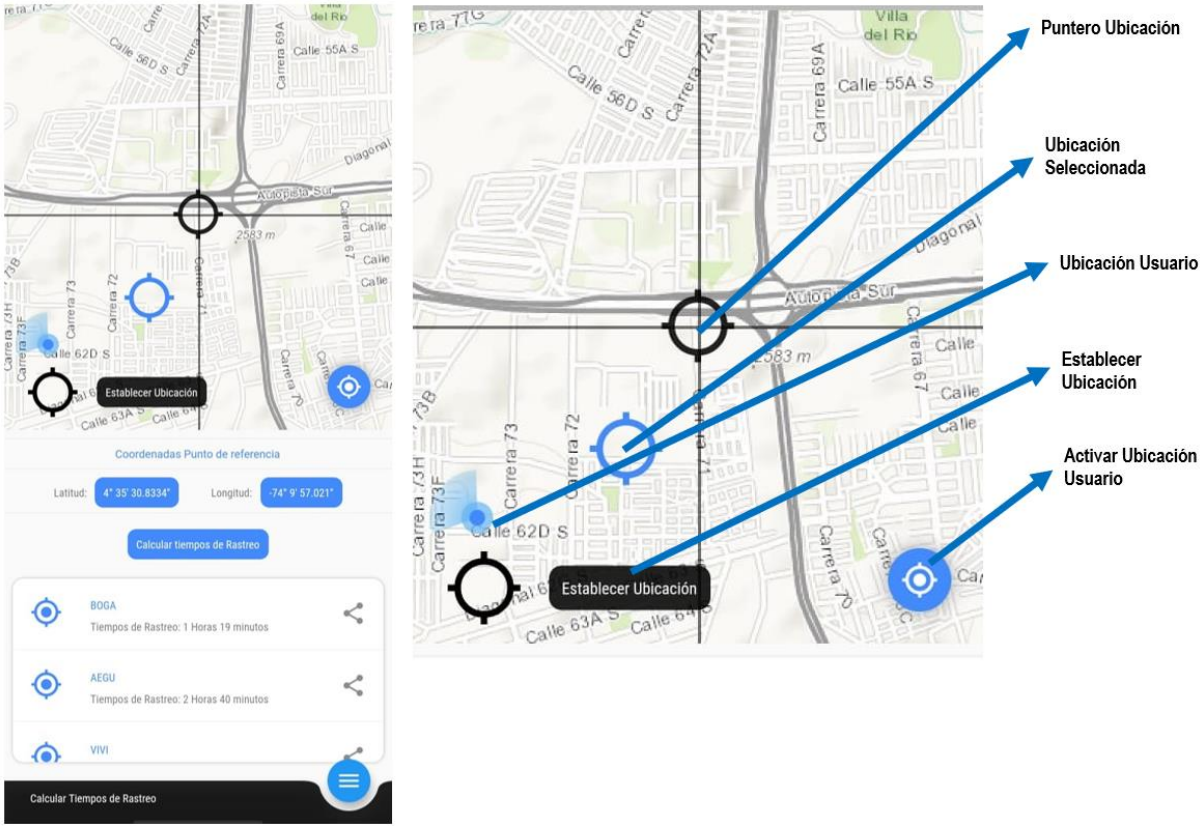


Ilustración 42 Establecimiento de Ubicación para cálculo de Rastreo

Fuente: Elaboración Propia

La interfaz para calcular los tiempos de rastreo esta divida en dos zonas principales; en la parte superior, se encuentran las herramientas para realizar la ubicación del punto desde el cual se realizará la observación GNSS en metodología estático diferencial, mientras que en la parte inferior se muestran los resultados obtenidos.

Dentro de la interfaz creada para facilitar el establecimiento de la ubicación del punto a observar, se crearon un total de 5 componentes:

1. Puntero de ubicación: Como su nombre lo indica permite establecer la ubicación del punto que se desea observar, siempre se encuentra ubicado en la mitad de la pantalla y el usuario ajusta mapa base hasta encontrar la ubicación deseada.

2. Ubicación Seleccionada: Es un icono que muestra la ubicación espacial del punto seleccionado para realizar el rastreo.
3. Establecer ubicación: Es un botón encargado de establecer y/o actualizar la ubicación del punto seleccionado para realizar el rastreo.
4. Ubicación del Usuario: Este componente muestra la ubicación del usuario obtenida de la antena GNSS integrada dentro del celular. Adicionalmente, muestra una circunferencia con dimensiones equivalentes a la precisión de la ubicación que se está obteniendo, así como una sombra que indica la dirección de avance del usuario cuando este se está desplazando.
5. Activar Ubicación Usuario: Este componente permite activar la ubicación del usuario y adicionalmente le permite realizar un acercamiento a su posición.

Realizada la programación de las funcionalidades de los diferentes componentes, se procedió a implementar la consulta en la base de datos y el modelamiento del objeto de referencia mediante el cual se almacena la información espacial del punto para posteriormente ser representado en el MapView.

```
class Estaciones{
  int id;
  String nombre;
  double latitud, longitud;

  Estaciones({this.id, this.nombre, this.latitud, this.longitud});

  factory Estaciones.fromMap(Map<String, dynamic> json) => new Estaciones(
      id: json["id"],
      nombre: json["nombre"],
      latitud: json["latitud"],
      longitud: json["longitud"],
  );
}


  Future<List<MagnaEco>> getMagnaECO() async {
    WidgetsFlutterBinding.ensureInitialized();
    Database db = await mbBasedeDatos();
    var response = await db.rawQuery('SELECT * FROM MAGNAECO');
    List<MagnaEco> listaMagnaECO =
        response.map((c) => MagnaEco.fromMap(c)).toList();
    return listaMagnaECO;
  }

  Future<List<GeoRED>> getGeoRED() async {
    WidgetsFlutterBinding.ensureInitialized();
    Database db = await mbBasedeDatos();
    var response = await db.rawQuery('SELECT * FROM GEORED');
    List<GeoRED> listaGeoRED = response.map((c) => GeoRED.fromMap(c)).toList();
    return listaGeoRED;
  }
}
```

Con los objetos creados, se procedió a realizar la programación para mostrar esta información dentro del MapView, y realizar el cálculo de la distancia elipsoidal entre la ubicación del usuario y las estaciones de referencia seleccionadas por el mismo.

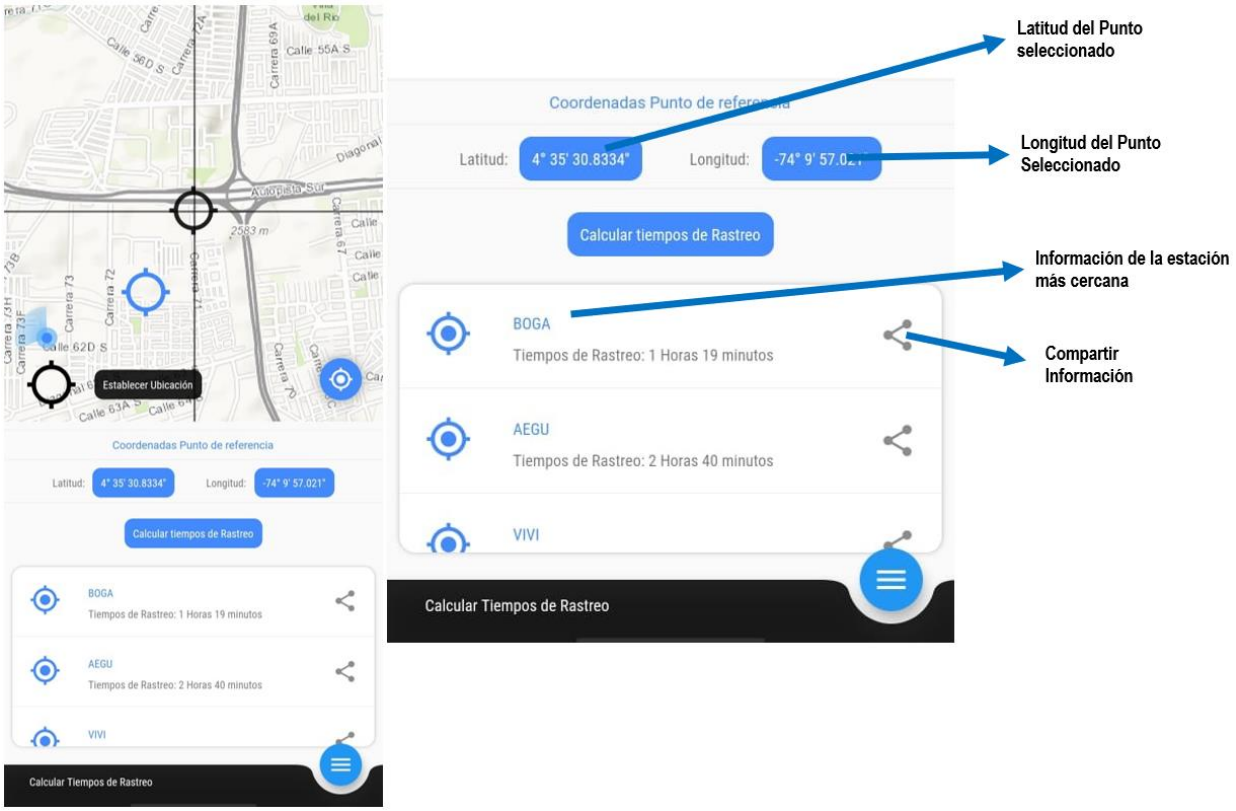


Ilustración 43 Cálculo de Tiempos de Rastreo

Fuente: Elaboración Propia

La interfaz en la parte inferior muestra la información correspondiente al cálculo de los tiempos realizados, con base a las estaciones de referencia más cercanas permitiendo inclusive compartir la información como se aprecia a continuación.

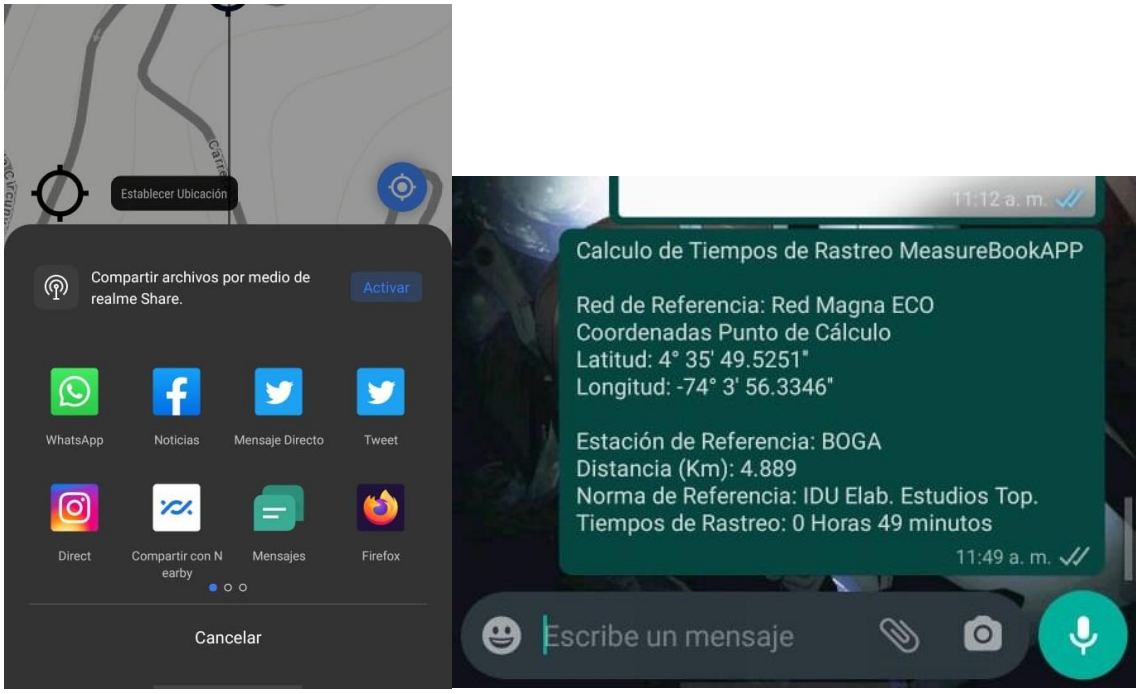


Ilustración 44 Compartir resultado cálculo tiempo de Rastreo

Fuente: Elaboración Propia

4.2.3 Conversión de Coordenadas

El desarrollo de la funcionalidad que permite la conversión de coordenadas se realizó mediante la implementación de dos módulos; en el primero se podrá realizar la conversión de puntos de forma individual mientras que, en la segunda opción el usuario podrá realizar el cargue de un archivo en formato CSV con el listado de las coordenadas que desea convertir.

A continuación, se muestran los objetos creados para la conversión de coordenadas.

Origen Cartesiano:

```
class CartesianasCS {
   String NOMBRE;
   double LATITUD, LONGITUD, NORTE, ESTE, PLANO_PROY;
   CartesianasCS({ this.NOMBRE, this.LATITUD, this.LONGITUD, this.NORTE, this.ESTE,
this.PLANO_PROY});

   factory CartesianasCS.fromMap(Map<String, dynamic> json) => new CartesianasCS(
    NOMBRE: json["NOMBRE"],
```

```
    LATITUD: json['LATITUD'],
    LONGITUD: json['LONGITUD'],
    NORTE: json['NORTE'],
    ESTE: json['ESTE'],
    PLANO_PROY: json['PLANO_PROY']
   );
 }
```

Coordenadas Cartesianas:

```
class CoordenadasCartesianas {
  double norte, este, altura;
  String idPunto, tipoAltura;
  CoordenadasCartesianas ({ this.idPunto, this.norte, this.este, this.altura,
this.tipoAltura});

  factory CoordenadasCartesianas.fromMap(Map<String, dynamic> json) => new
CoordenadasCartesianas(
    idPunto: json['idPunto'],
    norte: json["norte"],
    este: json["este"],
    altura: json["altura"],
    tipoAltura: json["tipoAltura"],
  );

}
```

Coordenadas Elipsoidales:

```
class CoordenadasElipsoidales {
  double latitud, longitud, altitud;

  CoordenadasElipsoidales ({this.latitud, this.longitud, this.altitud});

  factory CoordenadasElipsoidales.fromMap(Map<String, dynamic> json) =>
new CoordenadasElipsoidales(
    latitud: json["latitud"],
    longitud: json["longitud"],
    altitud: json["altitud"]
  );

}
```

Coordenadas Geocéntricas:

```
class CoordenadasGeocentricas {
  double x,y,z;

  CoordenadasGeocentricas ({this.x, this.y, this.z});

    factory CoordenadasGeocentricas.fromMap(Map<String, dynamic> json) => new
CoordenadasGeocentricas(
    x: json["x"],
    y: json["y"],
    z: json["z"]
  );
}
```

Coordenadas Gauss:

```
class CoordenadasGauss {
  double norte, este, altura;
  String idPunto, tipoAltura;
  CoordenadasGauss ({ this.idPunto, this.norte, this.este, this.altura, this.tipoAltura});

  factory CoordenadasGauss.fromMap(Map<String, dynamic> json) => new CoordenadasGauss(
    idPunto: json['idPunto'],
    norte: json["norte"],
    este: json["este"],
    altura: json["altura"],
    tipoAltura: json["tipoAltura"],
  );

}
```

Origen Gauss:

```
class GaussCS {
    String NOMBRE;
   double LATITUD, LONGITUD, NORTE, ESTE, N0;
   GaussCS({ this.NOMBRE, this.LATITUD, this.LONGITUD, this.NORTE, this.ESTE, this.N0});

   factory GaussCS.fromMap(Map<String, dynamic> json) => new GaussCS(
    NOMBRE: json["NOMBRE"],
    LATITUD: json['LATITUD'],
    LONGITUD: json['LONGITUD'],
    NORTE: json['NORTE'],
    ESTE: json['ESTE'],
    N0: json['N0']
   );
 }
```

Los modelos de datos anteriores están diseñados para almacenar los datos obtenidos de consulta en la base de datos.

La conversión de coordenadas fue programada dentro de una clase, a continuación, se muestran los modelos matemáticos usados para realizar la conversión:

Conversión de Geocéntricas a Elipsoidales.

```
// Conversión de Coordenadas Geocéntricas a Elipsoidales
CoordenadasElipsoidales geocentricas2Elipsoidales
(CoordenadasGeocentricas coordenadasGeocentricas){
  //datos de Entrada
  double _X = coordenadasGeocentricas.x;
  double _Y = coordenadasGeocentricas.y;
  double _Z = coordenadasGeocentricas.z;

  // Variables para conversión.
  double _longitud = radianes2Grados(m.atan((_Y)/(_X)));
  double _a = 6378137.000;
  double _b = 6356752.31414;
```

```
  double _e1 = 0.00669438002290;
  double _e2 = 0.00673949677548;
  double _x2 = m.pow(_X, 2);
  double _y2 = m.pow(_Y, 2);
  double _v = (_Z*_a)/(m.sqrt(_x2+_y2)*_b);
  double _latitud = radianes2Grados(m.atan(
      (_Z+_e2*_b*m.pow(m.sin(_v),3))/(m.sqrt(_x2+_y2)-
_e1*_a*m.pow(m.cos(_v),3))
      ));
  double _N = (_a)/(m.sqrt(1-
_e1*m.pow(m.sin(grados2Radianes(_latitud)),2)));
  double _altura =
((m.sqrt(_x2+_y2))/(m.cos(grados2Radianes(_latitud))))-_N;
  CoordenadasElipsoidales _coordenadasElipsoidales =
CoordenadasElipsoidales();
  _coordenadasElipsoidales.altitud = _altura;
  _coordenadasElipsoidales.latitud = _latitud;
  _coordenadasElipsoidales.longitud = _longitud;

  return _coordenadasElipsoidales;
}
```

Conversión de Coordenadas Elipsoidales a Geocéntricas.

```
CoordenadasGeocentricas elipsoidales2Geocentricas
(CoordenadasElipsoidales _coordenadasElipsoidales){
double _a = 6378137.000;
double _e1 = 0.00669438002290;
double _latitud = _coordenadasElipsoidales.latitud;
double _longitud = _coordenadasElipsoidales.longitud;
double _altura = _coordenadasElipsoidales.altitud;
double _N = (_a)/(m.sqrt(1-
_e1*m.pow(m.sin(grados2Radianes(_latitud)),2)));
double _X =
(_N+_altura)*m.cos(grados2Radianes(_latitud))*m.cos(grados2Radianes(_long
itud));
double _Y =
(_N+_altura)*m.cos(grados2Radianes(_latitud))*m.sin(grados2Radianes(_long
itud));
double _Z = ((1-_e1)*_N+_altura)*m.sin(grados2Radianes(_latitud));
CoordenadasGeocentricas _coordenadasGeocentricas =
CoordenadasGeocentricas();
_coordenadasGeocentricas.x = _X;
_coordenadasGeocentricas.y = _Y;
_coordenadasGeocentricas.z = _Z;
return _coordenadasGeocentricas;
}
```

Conversión de coordinadas Elipsoidales a Gauss

```
CoordenadasGauss elipsoidales2Gauss(CoordenadasElipsoidales
_elipsoidales, GaussCS _gaussOrigen) {
  double _latitudPunto = _elipsoidales.latitud;
```

```
  double _longitudPunto = _elipsoidales.longitud;
  double _alturaPunto = _elipsoidales.altitud;
  // Constantes Elipsoide
  double _a = 6378137.000;
  double _b = 6356752.31414;
  double _e1 = 0.00669438002290;
  double _e2 = 0.00673949677548;
  double _latitudOrigen = _gaussOrigen.LATITUD;
  double _longitudOrigen = _gaussOrigen.LONGITUD;
  double _laRad = grados2Radianes(_latitudPunto);
  double _loRad = grados2Radianes(_longitudPunto);
  double _laORad = grados2Radianes(_latitudOrigen);
  double _loORad = grados2Radianes(_longitudOrigen);
  double _l = _loRad - _loORad;
  double _t = m.tan(_laRad);
  double _eta2 = _e2 * m.pow(m.cos(_laRad), 2);
  double _n = (_a - _b) / (_a + _b);
  double _sinLaRad = m.sin(_laRad);
  double _n1 = 1 - _e1 * m.pow(_sinLaRad, 2);
  double _N = _a / m.sqrt(_n1);
  double _A = (_a + _b) / 2 * (1 + m.pow(_n, 2) / 4 + m.pow(_n, 4) / 64);
  double _B = -(3 * _n / 2) + 9 * m.pow(_n, 3) / 16 - 3 * m.pow(_n, 5) /
32;
  double _C = 15 * m.pow(_n, 2) / 16 - 15 * m.pow(_n, 4) / 32;
  double _D = -(35 * m.pow(_n, 3) / 48) + 105 * m.pow(_n, 5) / 256;
  double _E = 315 * m.pow(_n, 4) / 512;
  double _arcoMP = _A * (_laRad + _B * m.sin(2 * _laRad) + _C * m.sin(4 *
_laRad) + _D * m.sin(6 * _laRad) + _E * m.sin(8 * _laRad));
  double _arcoMO = _A * (_laORad + _B * m.sin(2 * _laORad) + _C * m.sin(4
* _laORad) + _D * m.sin(6 * _laORad) + _E * m.sin(8 * _laORad));
  double _Darcos = _arcoMP - _arcoMO;
  double _N1 = _t * _N / 2 * m.pow(_l, 2) * m.pow(m.cos(_laRad), 2);
  double _N2 = _t * _N / 24 * m.pow(m.cos(_laRad), 4) * (5 - m.pow(_t, 2)
+ 9 * _eta2 + 4 * m.pow(_eta2, 2)) * m.pow(_l, 4);
  double _N3 = _t * _N / 720 * m.pow(m.cos(_laRad), 6) * (61 - 58 *
m.pow(_t, 2) + m.pow(_t, 4) + 270 * _eta2 - 330 * _eta2 * m.pow(_t, 2)) *
m.pow(_l, 6);
  double _N4 = _t * _N / 40320 * m.pow(m.cos(_laRad), 8) * (1385 - 3111 *
m.pow(_t, 2) + 543 * m.pow(_t, 4) - m.pow(_t, 6)) * m.pow(_l, 8);
  double _norte = 1000000 + _Darcos + _N1 + _N2 + _N3 + _N4;
  double _E1 = _N * _l * m.cos(_laRad);
  double _E2 = _N / 6 * m.pow(m.cos(_laRad), 3) * (1 - m.pow(_t, 2) +
_eta2) * m.pow(_l, 3);
  double _E3 = _N / 120 * m.pow(m.cos(_laRad), 5) * (5 - 18 * m.pow(_t,
2) + m.pow(_t, 4) + 14 * _eta2 - 58 * m.pow(_t, 2) * _eta2) * m.pow(_l,
5);
  double _E4 = _N / 5040 * m.pow(m.cos(_laRad), 7) * (61 - 479 *
m.pow(_t, 2) + 179 * m.pow(_t, 4) - m.pow(_t, 6)) * m.pow(_l, 7);
  double _east = 1000000 + _E1 + _E2 + _E3 + _E4;
  CoordenadasGauss coordenadasGauss = CoordenadasGauss();
  coordenadasGauss.norte = _norte;
  coordenadasGauss.este = _east;
  coordenadasGauss.altura = _alturaPunto;
  return coordenadasGauss;
}
```

Conversión de Coordenadas Gauss a Elipsoidales

```
CoordenadasElipsoidales gauss2Elipsoidales (GaussCS _origenGauss,
CoordenadasGauss _gauss) {
  double _alturaPunto = _gauss.altura;
  double _No = _gauss.norte;
  double _e2 = _gauss.este;
  double _X = 6378137.000;
  double _Y = 6356752.31414;
  double _laIni = 0.00669438002290;
  double _V = 0.00673949677548;
  double _gNO = 491767.5344;
  double _a = _No - _gNO;
  double _A1 = _e2 - 1000000.0;
  double _A2 = (_X - _Y) / (_X + _Y);
  double _J2 = (_X + _Y) / 2 * (1 + m.pow(_A2, 2) / 4 + m.pow(_A2, 4) /
64);
  double _J4 = 3 * _A2 / 2 - 27 * m.pow(_A2, 3) / 32 + 269 * m.pow(_A2,
5) / 512;
  double _J6 = 21 * m.pow(_A2, 2) / 16 - 55 * m.pow(_A2, 4) / 32;
  double _al = 151 * m.pow(_A2, 3) / 96 - 417 * m.pow(_A2, 5) / 128;
  double _be = 1097 * m.pow(_A2, 4) / 512;
  double _y = _a / _J2 + _J4 * m.sin(2 * _a / _J2) + _J6 * m.sin(4 * _a /
_J2) + _al * m.sin(6 * _a / _J2) + _be * m.sin(8 * _a / _J2);
  double _B0 = m.tan(_y);
  double _b = _V * m.pow(m.cos(_y), 2);
  double _C = _X / m.sqrt(1 - _laIni * m.pow(m.sin(_y), 2));
  double _E = _B0 / (2 * m.pow(_C, 2)) * (-1 - _b) * m.pow(_A1, 2);
  double _N = _B0 / (24 * m.pow(_C, 4)) * (5 + 3 * m.pow(_B0, 2) + 6 * _b
- 6 * m.pow(_B0, 2) * _b - 3 * m.pow(_b, 2) - 9 * m.pow(_B0, 2) *
m.pow(_b, 2)) * m.pow(_A1, 4);
  double _sene = _B0 / (720 * m.pow(_C, 6)) * (-61 - 90 * m.pow(_B0, 2) -
45 * m.pow(_B0, 4) - 107 * _b + 162 * m.pow(_B0, 2) * _b + 45 *
m.pow(_B0, 4) * _b) * m.pow(_A1, 6);
  double _di = _B0 / (40320 * m.pow(_C, 8)) * (1385 + 3633 * m.pow(_B0,
2) + 4096 * m.pow(_B0, 4) + 1575 * m.pow(_B0, 6)) * m.pow(_A1, 8);
  double _T = _y + _E + _N + _sene + _di;
  double _latitude = radianes2Grados(_T);
  double _loIni = 1 / (_C * m.cos(_y)) * _A1;
  double _lo = 1 / (6 * m.pow(_C, 3) * m.cos(_y)) * (-1 - 2 * m.pow(_B0,
2) - _b) * m.pow(_A1, 3);
  double _lo3 = 1 / (120 * m.pow(_C, 5) * m.cos(_y)) * (5 - 28 *
m.pow(_B0, 2) + 24 * m.pow(_B0, 4) + 6 * _b + 8 * m.pow(_B0, 2) * _b) *
m.pow(_A1, 5);
  double _lo4 = 1 / (5040 * m.pow(_C, 7) * m.cos(_y)) * (-61 - 662 *
m.pow(_B0, 2) - 1320 * m.pow(_B0, 4) - 720 * m.pow(_B0, 6)) * m.pow(_A1,
7);
  double _lon = grados2Radianes(_origenGauss.LONGITUD) + _loIni + _lo +
_lo3 + _lo4;
  double _longitude = radianes2Grados(_lon);
  CoordenadasElipsoidales elipCoor = CoordenadasElipsoidales();
  elipCoor.longitud = _longitude;
  elipCoor.latitud = _latitude;
```

```
  elipCoor.altitud= _alturaPunto;

  return elipCoor;
}
```

Conversión de Coordenadas Elipsoidales a Planas Cartesianas

```
CoordenadasCartesianas elipsoidales2Cartesianas (CartesianasCS
_cartesianasO, CoordenadasElipsoidales _coordenadasElipsoidales) {
  double _la = _coordenadasElipsoidales.latitud;
  double _lo = _coordenadasElipsoidales.longitud;
  double _alturaP = _coordenadasElipsoidales.altitud;
  double _a = 6378137.000;
  double _e1 = 0.00669438002290;
  double _laORad = grados2Radianes(_cartesianasO.LATITUD);
  double _loORad = grados2Radianes(_cartesianasO.LONGITUD);
  double _laRad = grados2Radianes(_la);
  double _loRad = grados2Radianes(_lo);
  double _pP = _cartesianasO.PLANO_PROY;
  double _deltaLa = _laRad - _laORad;
  double _deltaLo = _loRad - _loORad;
  double _laM = (_laORad + _laRad) / 2;
  double _sinLaRad = m.sin(_laRad);
  double _n1 = 1 - _e1 * m.pow(_sinLaRad, 2);
  double _N = _a / m.sqrt(_n1);
  double _sinLaORad = m.sin(_laORad);
  double _no = 1 - _e1 * m.pow(_sinLaORad, 2);
  double _No = _a / m.sqrt(_no);
  double _sinLaMRad = m.sin(_laM);
  double _n3 = 1 - _e1 * m.pow(_sinLaMRad, 2);
  double _M = _a * (1 - _e1) / m.pow(_n3, 1.5);
  double _Mo = _a * (1 - _e1) / m.pow(_no, 1.5);
  double _N1 = _deltaLa + m.tan(_laORad) * m.pow(_deltaLo, 2) * m.pow(_N,
2) * m.pow(m.cos(_laRad), 2) / (2 * _Mo * _No);
  double _N2 = 1 + _pP / _Mo;
  double _north = _Mo * _N1 * _N2 + 109320.965;
  double _E1 = _deltaLo * _N * m.cos(_laRad);
  double _E2 = 1 + _pP / _No;
  double _east = _E1 * _E2 + 92334.879;

  CoordenadasCartesianas _coordenadasCartesianas =
CoordenadasCartesianas();
  _coordenadasCartesianas.norte =  _north;
  _coordenadasCartesianas.este = _east;
  _coordenadasCartesianas.altura = _alturaP;

  return _coordenadasCartesianas;
}
```

Conversión de Coordenadas Planas Cartesianas a Elipsoidales

```
CoordenadasElipsoidales cartesianas2Elipoidales(CoordenadasCartesianas
_coordenadasCartesianas, CartesianasCS _cartesianasCS) {
```

```
  double _No = _coordenadasCartesianas.norte;
  double _e2 = _coordenadasCartesianas.este;
  double _alturaPunto = _coordenadasCartesianas.altura;
  double _X = 6378137.000;
  double _Y = 6356752.31414;
  double _laIni = 0.00669438002290;
  double _V = 0.00673949677548;
  double _a = grados2Radianes(_cartesianasCS.LATITUD);
  double _A1 = grados2Radianes(_cartesianasCS.LONGITUD);
  double _A2 = _cartesianasCS.PLANO_PROY;
  //Falso Norte
  double _J2 = _No - _cartesianasCS.NORTE;
  //Falso Este
  double _J4 = _e2 - _cartesianasCS.ESTE;
  double _J6 = m.sin(_a);
  double _al = 1 - _laIni * m.pow(_J6, 2);
  double _be = _X / m.sqrt(_al);
  double _y = _X * (1 - _laIni) / m.pow(_al, 1.5);
  double _B0 = _J2 / ((1 + _A2 / (_X * (1 - _laIni))) * _y);
  double _b = m.tan(_a) / (2 * _be * _y);
  double _C = m.pow(_J4 / (1 + _A2 / _X), 2);
  double _E = _B0 - _b * _C;
  double _N = _a + _E;
  double _latitude = _N * 57.29577951308232;
  double _sinLaRad = m.sin(_N);
  double _n2 = 1 - _laIni * m.pow(_sinLaRad, 2);
  double _Np = _X / m.sqrt(_n2);
  double _loIni = _J4 / (_Np * m.cos(_N) * (1 + _A2 / _X));
  double _lo = _A1 + _loIni;
  double _lon = _lo * 57.29577951308232;
  CoordenadasElipsoidales _coordenadasElipsoidales =
CoordenadasElipsoidales();
  _coordenadasElipsoidales.latitud = _latitude;
  _coordenadasElipsoidales.longitud = _lon;
  _coordenadasElipsoidales.altitud = _alturaPunto;
  return _coordenadasElipsoidales;
}
```

Conversión de coordenadas Elipsoidales a Transversal de Mercator (CTM12)

```
CoordenadasON elipsoidales2GaussNuevo(CoordenadasElipsoidales
_coordenadasElipsoidales) {
  double _latitudPunto = _coordenadasElipsoidales.latitud;
  double _longitudPunto = _coordenadasElipsoidales.longitud;
  double _alturaPunto = _coordenadasElipsoidales.altitud;
  // Factor de Escala
  double k = 0.9992;
  // Constantes Elipsoide
  double _a = 6378137.000;
  double _b = 6356752.31414;
  double _e1 = 0.00669438002290;
  double _e2 = 0.00673949677548;
  double _latitudOrigen = 4;
  double _longitudOrigen = -73;
  double _laRad = grados2Radianes(_latitudPunto);
```

```
  double _loRad = grados2Radianes(_longitudPunto);
  double _laORad = grados2Radianes(_latitudOrigen);
  double _loORad = grados2Radianes(_longitudOrigen);
  double _l = _loRad - _loORad;
  double _t = m.tan(_laRad);
  double _eta2 = _e2 * m.pow(m.cos(_laRad), 2);
  double _n = (_a - _b) / (_a + _b);
  double _sinLaRad = m.sin(_laRad);
  double _n1 = 1 - _e1 * m.pow(_sinLaRad, 2);
  double _N = _a / m.sqrt(_n1);
  double _A = (_a + _b) / 2 * (1 + m.pow(_n, 2) / 4 + m.pow(_n, 4) / 64);
  double _B = -(3 * _n / 2) + 9 * m.pow(_n, 3) / 16 - 3 * m.pow(_n, 5) /
32;
  double _C = 15 * m.pow(_n, 2) / 16 - 15 * m.pow(_n, 4) / 32;
  double _D = -(35 * m.pow(_n, 3) / 48) + 105 * m.pow(_n, 5) / 256;
  double _E = 315 * m.pow(_n, 4) / 512;
  double _arcoMP = _A * (_laRad + _B * m.sin(2 * _laRad) + _C * m.sin(4 *
_laRad) + _D * m.sin(6 * _laRad) + _E * m.sin(8 * _laRad));
  double _arcoMO = _A * (_laORad + _B * m.sin(2 * _laORad) + _C * m.sin(4
* _laORad) + _D * m.sin(6 * _laORad) + _E * m.sin(8 * _laORad));
  double _Darcos = _arcoMP - _arcoMO;
  double _N1 = _t * _N / 2 * m.pow(_l, 2) * m.pow(m.cos(_laRad), 2);
  double _N2 = _t * _N / 24 * m.pow(m.cos(_laRad), 4) * (5 - m.pow(_t, 2)
+ 9 * _eta2 + 4 * m.pow(_eta2, 2)) * m.pow(_l, 4);
  double _N3 = _t * _N / 720 * m.pow(m.cos(_laRad), 6) * (61 - 58 *
m.pow(_t, 2) + m.pow(_t, 4) + 270 * _eta2 - 330 * _eta2 * m.pow(_t, 2)) *
m.pow(_l, 6);
  double _N4 = _t * _N / 40320 * m.pow(m.cos(_laRad), 8) * (1385 - 3111 *
m.pow(_t, 2) + 543 * m.pow(_t, 4) - m.pow(_t, 6)) * m.pow(_l, 8);
  double _norte = 2000000 + (_Darcos + _N1 + _N2 + _N3 + _N4)*k;
  double _E1 = _N * _l * m.cos(_laRad);
  double _E2 = _N / 6 * m.pow(m.cos(_laRad), 3) * (1 - m.pow(_t, 2) +
_eta2) * m.pow(_l, 3);
  double _E3 = _N / 120 * m.pow(m.cos(_laRad), 5) * (5 - 18 * m.pow(_t,
2) + m.pow(_t, 4) + 14 * _eta2 - 58 * m.pow(_t, 2) * _eta2) * m.pow(_l,
5);
  double _E4 = _N / 5040 * m.pow(m.cos(_laRad), 7) * (61 - 479 *
m.pow(_t, 2) + 179 * m.pow(_t, 4) - m.pow(_t, 6)) * m.pow(_l, 7);
  double _east = 5000000 + (_E1 + _E2 + _E3 + _E4)*k;
  CoordenadasON coordenadasOn = CoordenadasON();
  coordenadasOn.norte = _norte;
  coordenadasOn.este = _east;
  coordenadasOn.altura = _alturaPunto;
  return coordenadasOn;
  }
```

Conversión de Coordenadas Transversal de Mercator (CTM12) a Elipsoidales

```
CoordenadasElipsoidales origenNacional2Elipsoidales(CoordenadasON
coordenadasOn){
    double norte = coordenadasOn.norte;
    double este = coordenadasOn.este;
    double altura = coordenadasOn.altura;
    double latitudOrigen = 4;
    double longitudOrigen = -73;
```

```
    double falsoNorte = 2000000;
    double falsoEste = 5000000;
    double k = 0.9992;
    double a = 6378137.000;
    double b = 6356752.31414;
    double f = 1-b/a;
    double e2 = 2*f-m.pow(f, 2);
    double Np = norte - falsoNorte;
    // Función Punto Guia
    double puntoGuia(double latitud){
      double A0 = 1-(e2/4)-((3*m.pow(e2, 2))/64)-((5*m.pow(e2, 3))/256);
      double A2 = (3/8)*(e2+((m.pow(e2, 2))/4)+((15*m.pow(e2, 3))/128));
      double A4 = (15/256)*(m.pow(e2, 2)+((3*m.pow(e2, 3))/4));
      double A6 = (35*m.pow(e2, 3)/3072);
      double lat = grados2Radianes(latitud);
      double M = a*(A0*lat -A2*m.sin(2*lat)+A4*m.sin(4*lat)-
A6*m.sin(6*lat));
      return M;
    }
    //punto medio de la tierra
    double latR = grados2Radianes(latitudOrigen);
    double lgnR = grados2Radianes(longitudOrigen);
    double p = (a*(1-e2))/(m.pow((1-e2*m.pow(m.sin(latR),2)), 3/2));
    double v = (a)/(m.pow((1-e2*m.pow(m.sin(latR),2)), 1/2));
    double w = v/p;
    double t = m.tan(latR);
    double Ep = este - falsoEste;
    double mp = puntoGuia(latitudOrigen)+(Np/k);
    //
    double n = (a-b)/(a+b);
    //
    double G = a*(1-n)*(1-m.pow(n, 2))*(1+(9*m.pow(n,
2)/(4))+(225*m.pow(n, 4)/(64)));
    double O = mp/G;
    double latp = O +(3*n/2-(27*m.pow(n, 3))/32)*m.sin(2*O)+(21*m.pow(n,
2)/16-55*m.pow(n, 4)/32)*m.sin(4*O)+
                  (151*m.pow(n, 3)/96)*m.sin(6*O)+(1097*m.pow(n,
4)/512)*m.sin(8*O);
    double pp = (a*(1-e2))/(m.pow((1-e2*m.pow(m.sin(latp),2)), 3/2));
    double vp = (a)/(m.pow((1-e2*m.pow(m.sin(latp),2)), 1/2));
    double wp = vp/pp;
    double tp = m.tan(latp);
    double x = Ep/(k*vp);
    double T1 = (tp*Ep*x)/(k*pp*2);
    double T2 = ((tp*Ep*m.pow(x, 3))/(k*pp*24))*(-4*m.pow(w, 2)+9*w*(1-
m.pow(tp, 2))+12*m.pow(tp, 2));
    double T3 = ((tp*Ep*m.pow(x, 5))/(k*pp*720))*(8*m.pow(w, 4)*(11-
24*m.pow(tp, 2))-12*m.pow(w, 3)*(21-71*m.pow(tp, 2))
                +15*m.pow(wp, 2)*(15-98*m.pow(tp, 2)+15*m.pow(tp,
4))+180*w*(5-m.pow(tp, 2)-3*m.pow(tp, 4))+360*m.pow(t, 4));
    double T4 = ((tp*Ep*m.pow(x, 7))/(k*pp*40320))*(1385+3633*m.pow(tp,
2)+4095*m.pow(tp, 4)+1575*m.pow(tp, 6));
    double latS = latp -T1+T2-T3+T4;
    double latGrados = radianes2Grados(latS);


    // Conversión Longitud
```

```
    double t1 = (x)/(m.cos(latp));
    double t2 = ((m.pow(x, 3))/(6*m.cos(latp)))*(w+2*m.pow(tp, 2));
    double t3 = ((m.pow(x, 5))/(120*m.cos(latp)))*(-4*m.pow(w, 3)*(1-
6*m.pow(tp, 2))+m.pow(w, 2)*(9-68*m.pow(tp, 2))+72*w*m.pow(tp,
2)+24*m.pow(tp, 4));
    double t4 = ((m.pow(x, 7))/(5040*m.cos(latp)))*(61+662*m.pow(tp,
2)+1320*m.pow(t, 4)+720*m.pow(t, 6));
    double lgnS = lgnR +t1-t2+t3-t4;
    double lgnGrados = radianes2Grados(lgnS);

    CoordenadasElipsoidales coordenadasElipsoidales =
CoordenadasElipsoidales();
    coordenadasElipsoidales.latitud = latGrados;
    coordenadasElipsoidales.longitud = lgnGrados;
    coordenadasElipsoidales.altitud = altura;

    return coordenadasElipsoidales;
  }
```

Se crearon dos módulos que mediante swipe que permiten seleccionar el sistema de coordenadas de origen y el sistema de coordenadas de destino tanto para la conversión de un punto individual como la conversión de un archivo.

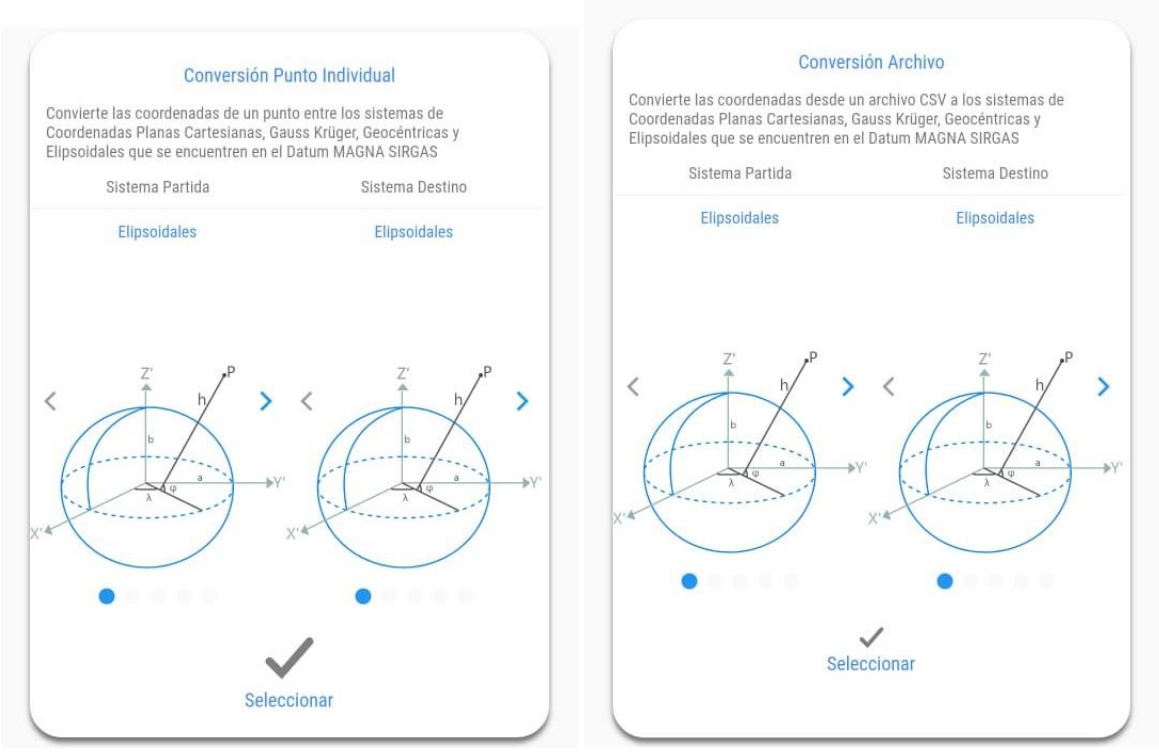


Ilustración 45 Conversión de Coordenadas

Fuente: Elaboración Propia

La conversión de coordenadas se puede realizar entre los diferentes sistemas anteriormente expuestos, tanto para los puntos individuales como para los puntos cargados mediante archivo CSV.

En la siguiente ilustración, se aprecian las diferentes interfaces diseñadas para la conversión de coordenadas de puntos individuales.

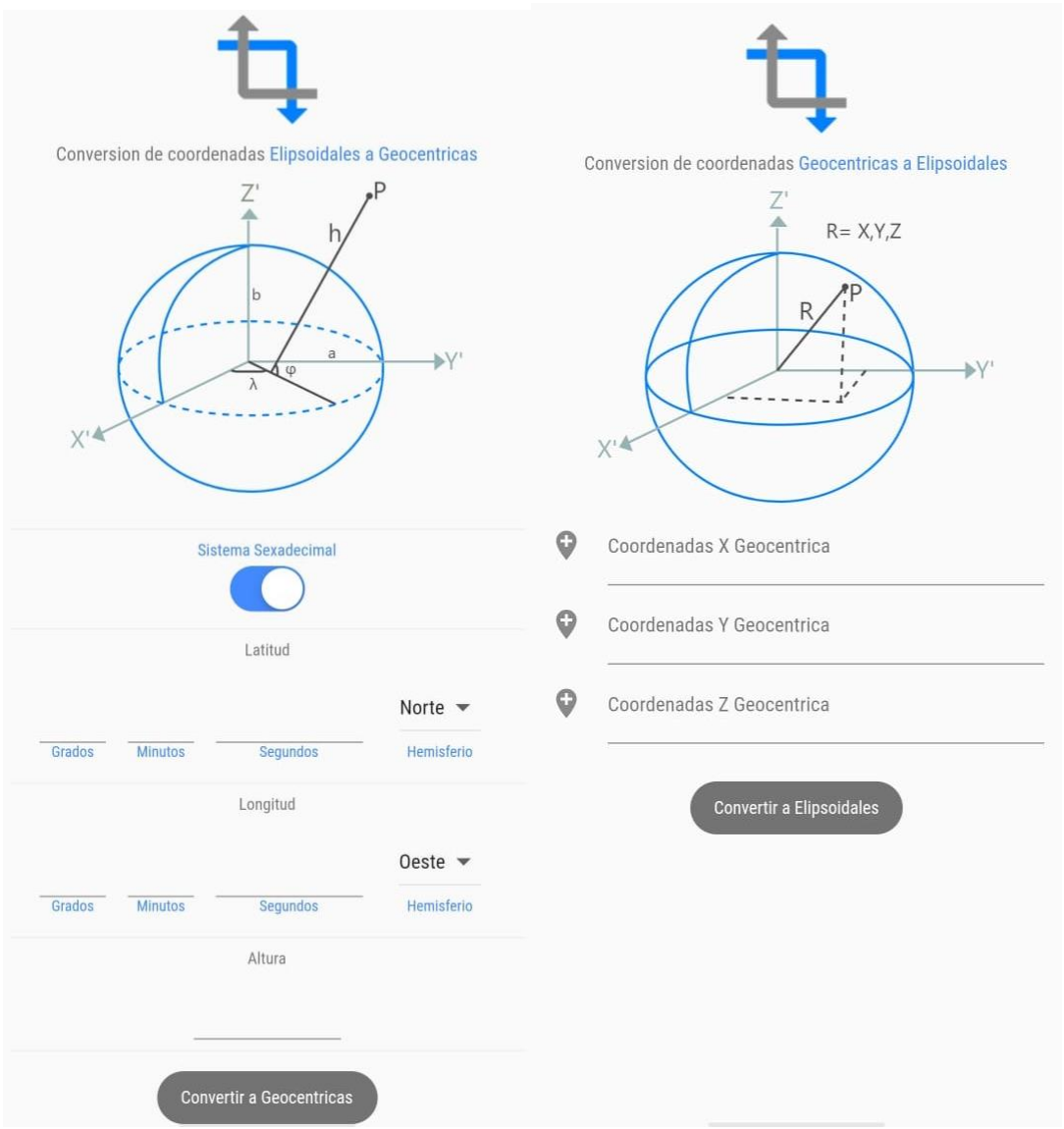


Ilustración 46 Interfaz conversión de Coordenadas Elipsoidales y Geocéntricas Punto Individual

Fuente: Elaboración Propia

Para la conversión de puntos individuales en los sistemas de coordenadas Planos Cartesianos y Gauss Krüger, se tienen componentes de tipo swipe para seleccionar el origen del sistema de coordenadas. Para el caso de los planos cartesianos la selección está configurada para seleccionar primero el departamento y posteriormente el municipio del plano cartesiano a elegir.

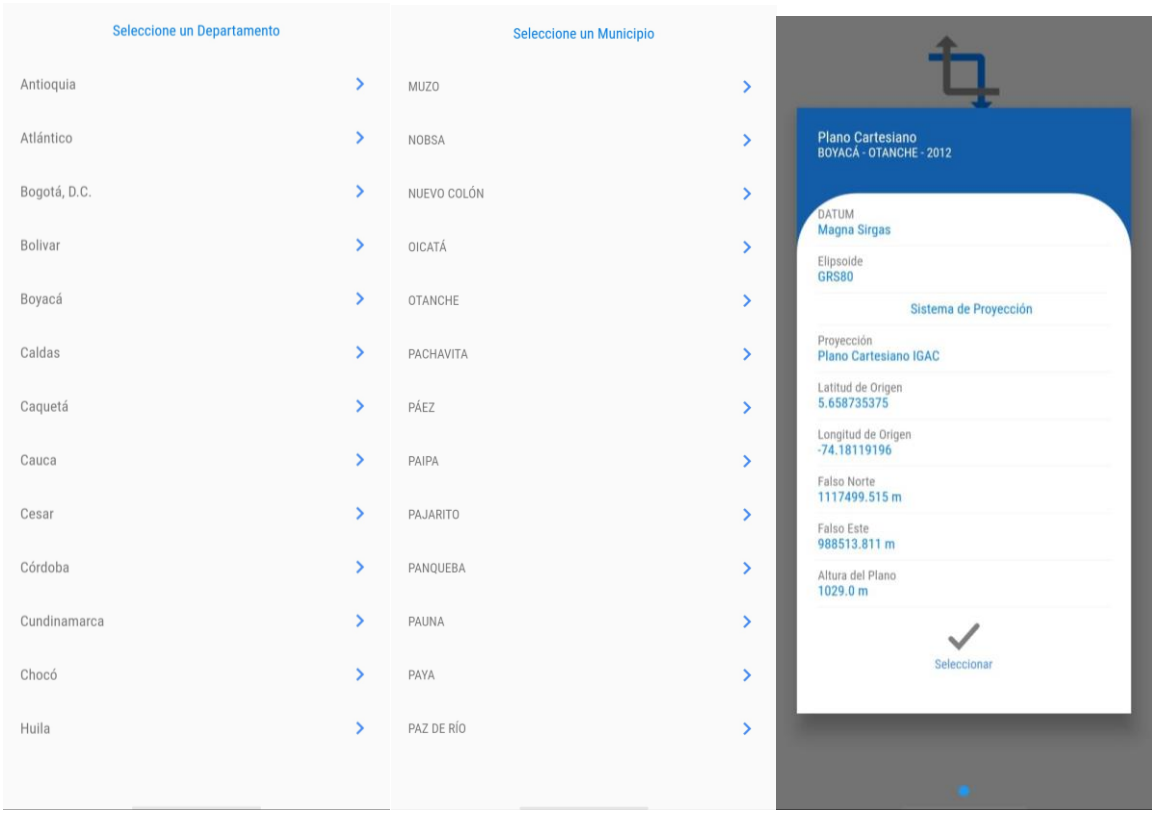


Ilustración 47 Selección de Origen de Plano Cartesiano

Fuente: Elaboración Propia

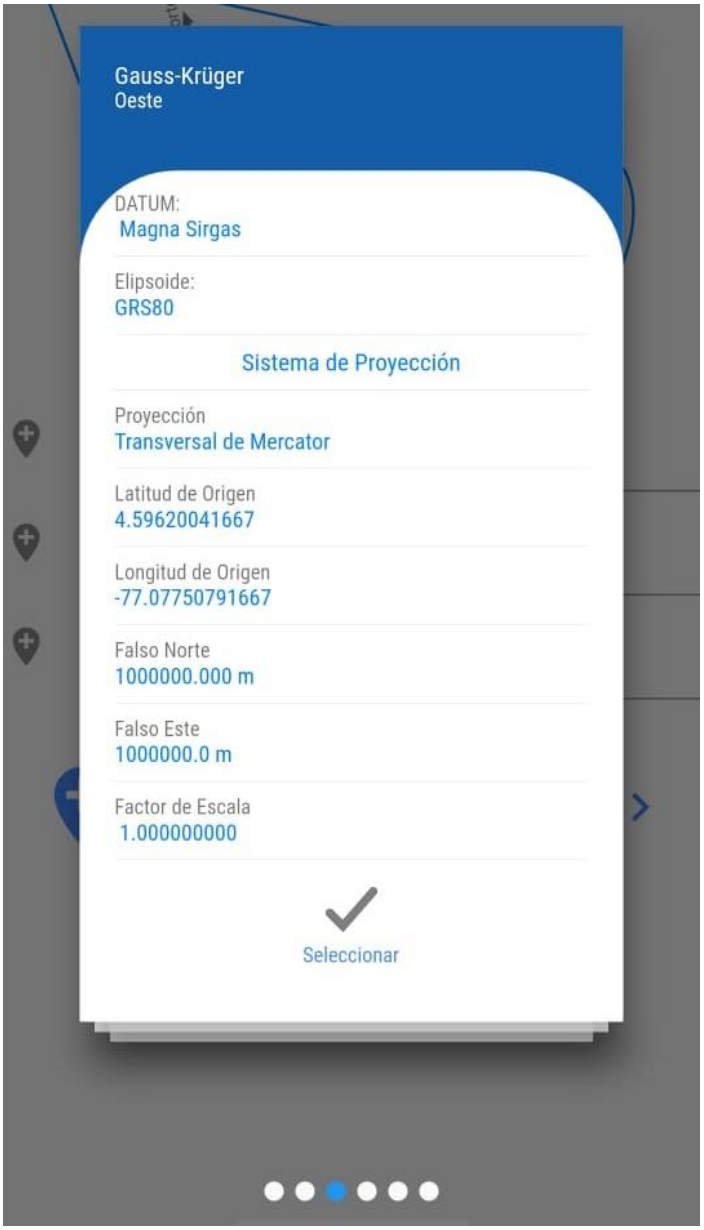


Ilustración 48 Selección de Origen Proyección Gauss Krüger

Fuente: Elaboración Propia

Finalmente, para la conversión de un archivo el usuario puede importar el listado de coordenadas en formato csv y exportarlo en el mismo formato. Se implementó la opción de compartir de tal forma que se facilite compartir las coordenadas en formato CSV mediante cualquier aplicación que se tenga instalada en el dispositivo que permita enviar archivos.



Ilustración 49 Configuración de Archivo Importado

Fuente: Elaboración Propia

Inicialmente, el usuario tendrá que cargar el archivo CSV y posteriormente realizar la configuración del mismo, relacionando en la aplicación los componentes norte, este y altura o el que corresponda dependiendo del sistema de coordenadas de origen seleccionado. Adicionalmente, tendrá que indicar si el archivo tiene encabezado o no, seleccionando la correspondencia entre la columna y el identificador correspondiente.

5. CONCLUSIONES

5.1 Verificación, contraste y evaluación de los objetivos

Realizar una aplicación móvil para optimizar los trabajos topográficos es estar en concordancia con las nuevas tecnologías que se implementan a diario en la geomática como drones, LIDAR, tecnologías PPK, escáner laser, entre otras. La aplicación tenía una meta principal y era mejorar y actualizar la manera en la cual se usa la información geográfica para cualquier posicionamiento GNSS. Lograr que un ingeniero topográfico diseñe e implemente una aplicación móvil para su propio uso, demuestra la relevancia de los conocimientos de lógica de programación, sistemas de información geográfica, bases de datos, programación asincrónica, conocimientos de frameworks, lenguajes de programación asincrónicos, entre otros. Las generaciones de ingenieros topográficos han evolucionado a través del tiempo, de la mano de innovaciones tecnológicas para lograr que la práctica y el análisis de la información en oficina sea más optimo y se disminuya al máximo el error humano. La aplicación móvil desarrollada permitió que con el módulo de la red pasiva del IGAC se facilite la identificación de los puntos cercanos en campo en cualquier apartado rural o urbano del país, así como la información que los constituyen. De la misma manera, el módulo de cálculo de tiempos de rastreo, utilizando las estaciones de la red continua del IGAC y del Servicio Geológico Colombiano permite mejorar la obtención de información técnica en campo ya que se tienen en cuenta las normativas de la Empresa de Acueducto y Alcantarillado de Bogotá EAAB-ESP, el Instituto de Desarrollo Urbano IDU y el Instituto Geográfico Agustín Codazzi. Esto permite tener certeza del tiempo necesario para cumplir con las normas del país sin la posibilidad de error, debido a un cálculo preciso realizado con las fórmulas planteadas en los documentos técnicos de las normas NS-030, norma de elaboración de estudios topográficos y la resolución 643 de 2018 respectivamente. Es importante nombrar que la aplicación tuvo pruebas en campo exitosas especificadas en las historias de usuario. Se implementó la persistencia de datos dentro de la aplicación para lograr su funcionamiento sin internet. Por otra parte, se implementó el módulo de conversión de coordenadas, lo cual permite realizar la conversión de coordenadas de forma individual y masiva mediante la carga de archivos en formato CSV o

individualmente. Los archivos con los resultados de la conversión se podrán enviar por aplicaciones como WhatsApp, Gmail, Outlook o cualquier aplicación que permita compartir archivos.

Las ventajas y la automatización de tareas que presenta MeasureBook representa innovación y preocupación por las actividades propias de un ingeniero topográfico que trabaja en campo y oficina teniendo un equipo de trabajo a cargo. Esto permitirá planificar, cambiar planes en campo y confiar en los resultados que se van a obtener con las opciones que brinda la aplicación.

5.2 Síntesis del modelo Propuesto

Para el desarrollo de este proyecto se utilizaron las siguientes herramientas:

- Visual Studio Code, como IDE para el desarrollo de la Aplicación
- Android Studio, para la emulación de dispositivos android durante la fase de desarrollo
- Librerías de código abierto, como herramientas de software intermedio para la implementación de módulos en concordancia al objetivo del proyecto
- Flutter como framework de trabajo
- Dart como lenguaje de programación asincrónico
- SQLite como base de datos de forma local y asegurar la persistencia de los datos.

6. BIBLIOGRAFÍA

Android. (Octubre de 2019). *Developers*. Obtenido de Android Studio: https://developer.android.com/studio/releases/platform-tools

Android Developers. (10 de Junio de 2020). Obtenido de Android 11 Beta is here: https://developer.android.com/android11

Basterra, B. B. (2016). *Android OS Documentation.* Obtenido de https://readthedocs.org/projects/androidos/downloads/pdf/latest/

Batanero, A., & Sesma, M. (2017). *Apps Híbridas VS Apps Nativas*. Obtenido de Movilidad y Desarrollo de software: https://www.paradigmadigital.com/dev/versus-apps-hibridas-vs-apps-nativas/

Brogdon, A. (12 de Julio de 2019). *Flutter YouTube Channel*. Obtenido de Async/Await - Flutter in Focus: https://www.youtube.com/watch?v=SmTCmDMi4BY

Dart Dev. (2020). *Dart*. Obtenido de https://dart.dev/

Dart Doc. (2019). *Dartium: Chromium with the Dart VM*. Obtenido de https://dartdoc.takyam.com/tools/dartium/

deTopografia. (22 de Septiembre de 2019). *deTopografía.Blogspot*. Obtenido de deTopografía: http://detopografia.blogspot.com/2012/12/la-superficie-de-proyeccion-el-elipsoide.html

Escola Tècnica Superior d'Enginyeria Informàtica. (2012). *Historia de la Informática*. Obtenido de Introducción a Android: http://www.jtech.ua.es/dadm/restringido/android/sesion01-apuntes.pdf

ESRI. (22 de Septiembre de 2019). *Arcgis Desktop*. Obtenido de Proyección Transversal de Mercator: http://desktop.arcgis.com/es/arcmap/10.3/guide-books/map-projections/transverse-mercator.htm

Furones, A. (2010). *Sistema y Marco de Referencia Terrestre.* (G. y. Departamento de Ingeniería Cartográfica, Ed.) Valencia, España: Universidad Politécnica de Valencia. Obtenido de http://www.upv.es/unigeo/index/docencia/etsigct/astronomia/teoria/astronomia210.pdf

Google. (2020). *Flutter*. Obtenido de https://flutter.dev/

Gutiérrez. , J. (2011). *Departamento de Lenguajes y Sistemas Informáticos* . Obtenido de Universidad de Sevilla: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Huerta, E., Mangiaterra, A., & Noguera, G. (2005). *GPS Posicionamiento Satelital.* Rosario, Argentina: Universidad Nacional de Rosario.

Instituto Geográfico Agustín Codazzi IGAC. (2004). *Tipos de Coordenadas Manejados en Colombia .* Obtenido de https://www.igac.gov.co/sites/igac.gov.co/files/anexo_8._tipos_de_coordenadas_manejados_en_colombia_igac._ano_2004.pdf

Instituto Geográfico Agustín Codazzi IGAC. (Noviembre de 2011). Manual de procedimientos. *Levantamientos Topográficos De Precisión* . Bogotá D.C., Colombia.

Instituto Geográfico Agustín Codazzi IGAC. (24 de Septiembre de 2018). *Red de Nivelación.* Obtenido de Sistema Altura Nacional de Referencia: https://www.igac.gov.co/es/contenido/areas-estrategicas/red-de-nivelacion

Instituto Geográfico Agustín Codazzi IGAC. (12 de 09 de 2020). *Red MAGNA ECO*. Obtenido de Red MAGNA ECO: https://www.igac.gov.co/es/contenido/areas-estrategicas/red-magna-eco

Instituto Geográfico Agustin Codazzi IGAC. (3 de Octubre de 2020). *Red pasiva MAGNA-SIRGAS.* Obtenido de Campañas de ocupación 1994, 1995 y 1997: https://www.igac.gov.co/es/contenido/areas-estrategicas/red-pasiva-magna-sirgas

InterWare. (12 de 09 de 2020). *¿Qué es UI?* Obtenido de InterWare: https://www.interware.com.mx/blog/que-es-ui-y-en-que-consiste

IONOS ES. (25 de Septiembre de 2019). *Digital Guide Ionos* . Obtenido de Software Development Kit: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/software-development-kit/

Leler, W. (26 de Febrero de 2018). *Why Flutter Uses Dart*. Obtenido de https://medium.com/hackernoon/why-flutter-uses-dart-dd635a054ebf

Microsoft. (2020). *Visual Studio Code*. Obtenido de Visual Studio Code: https://code.visualstudio.com/docs/containers/overview

Nahuel Delía, L. (Marzo de 2017). *Desarrollo de Aplicaciones Móviles Multiplataforma.* Obtenido de Universidad Nacional de la Plata: https://pdfs.semanticscholar.org/9870/88f325381579daf18a5cc2725f317e755301.pdf

Naukas. (22 de Septiembre de 2019). *Naukas.com*. Obtenido de La verdad sobre el mapa de Peters: https://naukas.com/2012/05/07/el-mapa-de-peters/

NextU. (2018). *Aplicaciones Nativas VS Aplicaciones Híbridas.* Obtenido de https://www.nextu.com/blog/apps-nativas-vs-apps-hibridas/

NTS. (23 de 12 de 2019). *NTS Solutions*. Obtenido de ¿Cuáles son las novedades de Android 10?: https://www.nts-solutions.com/blog/android-10-novedades.php

Platzi. (2019). *Platzi*. Obtenido de Flutter ahora es multiplataforma: https://platzi.com/blog/flutter-multiplataforma/

Ramirez, I. (21 de Marzo de 2019). *xatakandroid*. Obtenido de Historia y evolución de Android: cómo un sistema operativo para cámaras digitales acabó conquistando los móviles: https://www.xatakandroid.com/sistema-operativo/historia-y-evolucion-de-android-como-un-sistema-operativo-para-camaras-digitales-acabo-conquistando-los-moviles

Scholarium. (2018). *Business Process Management (BPM / BPMN)*. Obtenido de https://scholarium.info/business-process-management-bpm-bpmn/

Servicio Geológico Colombiano . (2020). *¿Qué es GeoRED?* Obtenido de https://geored2.sgc.gov.co/Paginas/default.aspx

Servicio Geológico Colombiano. (2020). *Archivos RINEX y coordenadas.* Obtenido de https://geored2.sgc.gov.co/solicitudes/Paginas/default.aspx

Servicio Geológico Colombiano SGC. (s.f.). *Objetivos y funciones.* Obtenido de https://geored2.sgc.gov.co/Paginas/Objetivos-y-funciones.aspx#

SG Buzz Mexico. (12 de 09 de 2020). *SQLite: La Base de Datos Embebida*. Obtenido de SG Buzz Mexico: https://sg.com.mx/revista/17/sqlite-la-base-datos-embebida

Sistema gestor de trabajo Flokzu. (23 de Abril de 2018). *Gestión de procesos de negocios*. Obtenido de https://www.flokzu.com/blog/es/bpm_es/que-es-bpm/

Tutorials Point. (2020). *Flutter*. Obtenido de https://www.tutorialspoint.com/flutter/flutter_tutorial.pdf

Universidad Carlos III de Madrid. (s.f.). *Ingeniería Técnica de Telecomunicación: especialidad Telemática*. Obtenido de Programación en dispositivos móviles portables: https://sites.google.com/site/swcuc3m/home/android/portada

Universidad de Barcelona. (26 de 05 de 2020). *Facultad de informática de Barcelona*. Obtenido de ¿Qué es el lenguaje de programación Dart?: https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart

Universidad Politecnica de Valencia. (12 de 09 de 2020). *Nagarvil*. Obtenido de Metodos de posicionamiento GNSS: https://nagarvil.webs.upv.es/metodos-de-posicionamiento-gnss-gps/