UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Geodésica, Cartográfica y Topográfica

Desarrollo de una aplicación móvil para la ayuda a la regularización y restitución de tierras en Colombia

Trabajo Fin de Grado

Grado en Ingeniería Geomática y Topografía

AUTOR/A: Terevinto Charquero, Diego

Tutor/a: Femenia Ribera, Carmen

Cotutor/a: Mora Navarro, Joaquín Gaspar

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA GEODÉSICA CARTOGRÁFICA Y TOPOGRÁFICA

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA LA AYUDA A LA REGULARIZACIÓN Y RESTITUCIÓN DE TIERRAS EN COLOMBIA

Grado en Ingeniería en Geomática y Topografía
Fecha: Julio 2024
Curso: 2023-2024

Autor: Diego Terevinto Charquero
Tutores: Carmen Femenia Ribera
Joaquín Gaspar Mora Navarro

COMPROMISO:

"El presente documento ha sido realizado completamente por el firmante; no ha sido entregado como otro trabajo académico previo y todo el material tomado de otras fuentes ha sido convenientemente entrecomillado y citado su origen en el texto, así como referenciado en la bibliografía"

RESUMEN:

Este proyecto describe el desarrollo de una aplicación móvil diseñada para simplificar los procesos de restitución y regularización de tierras en Colombia, un problema que afecta aproximadamente al 60% de las tierras del país. La aplicación, desarrollada en el marco del proyecto MetaTierras y financiada por la UPV, permite a los campesinos capturar y enviar datos georreferenciados directamente desde sus campos a un servidor central. Esta información será posteriormente procesada por técnicos especializados.

La aplicación, que es instalable en cualquier dispositivo con sistema operativo Android, utiliza tecnologías de código abierto y ha sido implementada utilizando Angular y SQLite, lo que permite el almacenamiento de datos en el dispositivo, aún sin conexión a internet. Estos datos almacenados pueden ser enviados al servidor una vez que el dispositivo se conecte a internet. Mediante la aplicación, los propietarios pueden declarar dónde están sus propiedades, aportando de una forma sencilla una posición aproximada de sus linderos, así como los datos de la propiedad, y sus propietarios

El desarrollo de esta aplicación ha sido realizado en colaboración con la Fundación Forjando Futuros (FFF) y la Asamblea de Cooperación por la Paz (ACPP), dos ONG que operan en Colombia.

Palabras clave: catastro, restitución de tierras, desarrollo aplicaciones móviles, app, Angular, SQLite, georreferenciación, código abierto, Colombia, MetaTierras, Fundación Forjando Futuros (FFF), Asamblea de Cooperación por la Paz (ACPP).

RESUM:

Aquest projecte descriu el desenvolupament d'una aplicació mòbil dissenyada per a simplificar els processos de restitució i regularització de terres a Colòmbia, un problema que afecta aproximadament el 60% de les terres del país. L'aplicació, desenvolupada en el marc del projecte MetaTierras i finançada per la UPV, permet als llauradors capturar i enviar dades georeferenciades directament des dels seus camps a un servidor central. Aquesta informació és posteriorment processada per tècnics especialitzats.

L'aplicació, que es pot instal·lar en qualsevol dispositiu amb sistema operatiu Android, utilitza tecnologies de codi obert i ha estat implementada utilitzant Angular i SQLite, la qual cosa permet l'emmagatzematge de dades en el dispositiu, encara sense connexió a internet. Aquestes dades emmagatzemades poden ser enviades al servidor una vegada que el dispositiu se connecte a internet. Mitjançant l'aplicació, els propietaris poden declarar on estan les seues propietats, aportant de manera senzilla una posició aproximada dels seus límits, així com les dades de la propietat i els seus propietaris.

El desenvolupament d'aquesta aplicació ha sigut realitzat en col·laboració amb la Fundación Forjando Futuros (FFF) i l'Asamblea de Cooperación por la Paz (ACPP), dues ONG que operen a Colòmbia.

Paraules clau: cadastre, restitució de terres, desenvolupament d'aplicacions mòbils, app, Angular, SQLite, georeferenciació, codi obert, Colòmbia, MetaTierras, Fundación Forjando Futuros (FFF), Asamblea de Cooperación por la Paz (ACPP).

ABSTRACT:

This paper describes the development of a mobile application designed to simplify the processes of land restitution and regularization in Colombia, a problem that affects approximately 60% of the land in the country. The application, developed under the MetaTierras project and funded by the UPV, allows peasants to capture and send georeferenced data directly from their fields to a central server. This information is subsequently processed by specialized technicians.

The application, which can be installed on any device with an Android operating system, uses open-source technologies and has been implemented using Angular and SQLite, allowing data storage on the device, even without an internet connection. These stored data can be sent to the server once the device connects to the internet. Through the application, owners can declare where their properties are, providing a simple approximation of their boundaries, as well as the property data and its owners.

The development of this application has been done in collaboration with the Fundación Forjando Futuros (FFF) and the Asamblea de Cooperación por la Paz (ACPP), two NGOs operating in Colombia.

Keywords: cadastre, land restitution, mobile application development, app, Angular, SQLite, georeferencing, open source, Colombia, MetaTierras, Fundación Forjando Futuros (FFF), Asamblea de Cooperación por la Paz (ACPP).

ÍNDICE IMÁGENES

ÍNDICE DE TABLAS

ÍNDICE

1. OBJETIVOS

El objetivo principal de este proyecto ha sido crear una aplicación móvil que facilite la toma de datos en campo por parte de los campesinos en Colombia. Estos datos incluyen información de los interesados (nombre completo, porcentaje de propiedad, correo electrónico, etc.), detalles del predio (municipio, departamento, tipo de predio, etc.), y la geometría del predio (puntos que delimitan el predio, tomados con el GPS del móvil). La aplicación está diseñada para capturar, almacenar y enviar datos georreferenciados directamente desde el campo a un servidor central. Los técnicos especializados recibirán estos datos y completarán la información sobre cartografía oficial mediante digitalización en softwares especializados como QGIS.

Para alcanzar este objetivo principal, se han establecido los siguientes objetivos específicos:

- Desarrollar una interfaz de usuario intuitiva y accesible:
    - Diseñar una interfaz que sea fácil de usar para los campesinos, considerando sus diversas necesidades y limitaciones tecnológicas.
- Implementar funcionalidades de captura de datos georreferenciados:
    - Integrar la opción de capturar datos utilizando el GPS del móvil con la mayor precisión posible.
    - Permitir la captura de datos descriptivos textuales adicionales para complementar la información georreferenciada.
- Desarrollar un sistema de almacenamiento local de datos:
    - Utilizar SQLite para asegurar que los datos capturados se almacenen localmente en el dispositivo móvil, incluso en áreas sin cobertura de red.
- Facilitar la integración de los datos capturados con sistemas de información geográfica (SIG):
    - Asegurar que los datos obtenidos con la aplicación puedan ser fácilmente importados y utilizados en softwares de SIG como QGIS.

2. INTRODUCCIÓN

En Colombia, aproximadamente el 60% de las tierras no están registradas formalmente, lo que genera numerosos problemas legales y económicos. La falta de registros precisos y actualizados de la propiedad de la tierra se desencadena en conflictos territoriales, dificultades en la planificación del uso de la tierra y obstáculos para el desarrollo económico del país. Este problema se ve agravado por el histórico conflicto armado con las FARC, que desplazó a miles de personas de sus tierras y complicó aún más la situación del catastro.

Para abordar estos desafíos, se ha desarrollado una aplicación móvil en colaboración con la Fundación Forjando Futuros (FFF), y la Asamblea de Cooperación por la Paz (ACPP), dos ONGs que trabajan en Colombia en el marco del proyecto MetaTierras, financiado por la UPV. Por lo tanto, esta aplicación está diseñada para permitir que los campesinos declaren la ubicación de sus propiedades, proporcionando una posición aproximada de sus linderos y datos de propiedad, facilitando así la restitución y regularización de tierras.

2.1. IMPORTANCIA DEL CATASTRO

El catastro es un registro administrativo que contiene la descripción de los bienes inmuebles rurales y urbanos, con su respectiva ubicación georreferenciada. La importancia del catastro radica en su capacidad para proveer información precisa y actualizada sobre la propiedad de la tierra, lo cual es fundamental para la planificación del desarrollo urbano y rural, la administración tributaria, la protección del medio ambiente, proteger a los propietarios contra ocupaciones, y asegurar el tráfico inmobiliario.

En Colombia, de la gestión catastral se encarga el Instituto Geográfico Agustín Codazzi (IGAC), una entidad encargada de la elaboración, actualización y conservación de la cartografía básica y temática del país.

Imagen 1. Logo del IGAC

El IGAC propone el Catastro Multipropósito, una herramienta para el desarrollo del territorio que sirve como recurso para la formulación e implementación de políticas públicas, proporcionando así mayor seguridad jurídica, eficiencia del mercado inmobiliario, desarrollo y ordenamiento territorial.

Los beneficios de este catastro multipropósito son los siguientes:

- Facilita la toma de decisiones y la construcción de políticas públicas en el territorio.
- Constituye la base para la elaboración o revisión de los Planes de Ordenamiento Territorial (POT).
- Fortalece las finanzas públicas mediante la creación de oportunidades de recaudo y la implementación de otros mecanismos de captura de valor.
- Permite el análisis e impacto de la implementación de instrumentos de gestión, planificación y financiación del desarrollo territorial.
- Constituye la línea base para el diseño e implementación de estrategias masivas para la formalización, regularización o legalización de la propiedad.
- Favorece el desarrollo del mercado inmobiliario, generando seguridad y confianza en sus transacciones.
- Facilita el diseño e implementación de políticas públicas asociadas a la debida atención y prevención de riesgos y desastres.
- Permite contar con información veraz y confiable para la construcción de redes e infraestructuras asociadas a servicios públicos.

Fuente: Instituto Geográfico Agustín Codazzi. (2023). Catastro Multipropósito. https://igac.gov.co/catastro-multiproposito/catastro-multiproposito

La importancia de la implementación de este catastro multipropósito se vuelve aún más crucial al considerar los desafíos históricos y actuales relacionados con la tierra que persisten en el país colombiano.

2.2. EL PROBLEMA DE LAS TIERRAS EN COLOMBIA

La historia de Colombia está marcada por un prolongado conflicto armado, particularmente con las Fuerzas Armadas Revolucionarias de Colombia (FARC). Este conflicto ha tenido un impacto significativo en la propiedad y el uso de la tierra. La guerra, que duró más de cincuenta años, produjo el desplazamiento de miles de personas de sus tierras, originando una gran cantidad de propiedades no registradas y de conflictos territoriales que actualmente necesitan ser resueltos para lograr un correcto y próspero desarrollo sostenible en el país y la paz.

Fue en 1964 cuando se fundaron las FARC como una organización guerrillera de ideología marxista-leninista y llevaron a cabo una insurgencia contra el gobierno colombiano resultando en miles de muertos y desplazados. Durante ese periodo, el campo colombiano se vio enormemente afectado, con miles de familias perdiendo sus tierras y hogares. Este conflicto prolongado afectó gravemente la estructura agraria del país, aumentando así los problemas relacionados con la propiedad de la tierra.

Imagen 2. Bandera de las FARC

Como se ha mencionado antes, uno de los mayores y principales problemas que enfrenta Colombia actualmente es la falta de registro formal de la propiedad de la tierra, afectando aproximadamente al 60% del territorio. Esto crea grandes obstáculos para el desarrollo económico y la seguridad jurídica de los propietarios de las tierras, ya que, sin un registro adecuado, es muy complicado para los propietarios demostrar la titularidad, acceder a créditos y beneficiarse de programas gubernamentales. Por lo tanto, este problema de registro no afecta solo a los individuos, sino que también limita las oportunidades de desarrollo y planificación territorial a nivel nacional.

2.2.1. RESTITUCIÓN DE TIERRAS

En respuesta a estos desafíos mencionados, Colombia ha implementado un proceso de restitución de tierras que busca devolver a las personas desplazadas por el conflicto armado sus propiedades originales. Esto es muy importante para la recuperación y progreso en el país, ya que su éxito permitiría que las familias que han sido desplazadas puedan regresar a sus tierras y reconstruir sus vidas.

Una pieza clave de este proceso es la Ley 1448 de 2011, conocida como la Ley de Víctimas y Restitución de Tierras. Esta ley establece un marco integral para la reparación de las

víctimas del conflicto armado y la restitución de tierras despojadas. La ley reconoce como víctimas a todas aquellas personas que han sufrido violaciones a sus derechos humanos y al derecho internacional humanitario debido al conflicto armado.

Entre las medidas de reparación integral, la ley contempla la restitución de tierras como un componente central. Este proceso implica la devolución de las propiedades a las víctimas a través de un procedimiento administrativo y judicial que incluye la identificación de los predios y la entrega formal a los legítimos propietarios. Además, la Unidad de Restitución de Tierras es la entidad encargada de llevar a cabo este proceso, asegurando la protección jurídica de los reclamantes y brindando apoyo técnico y logístico.

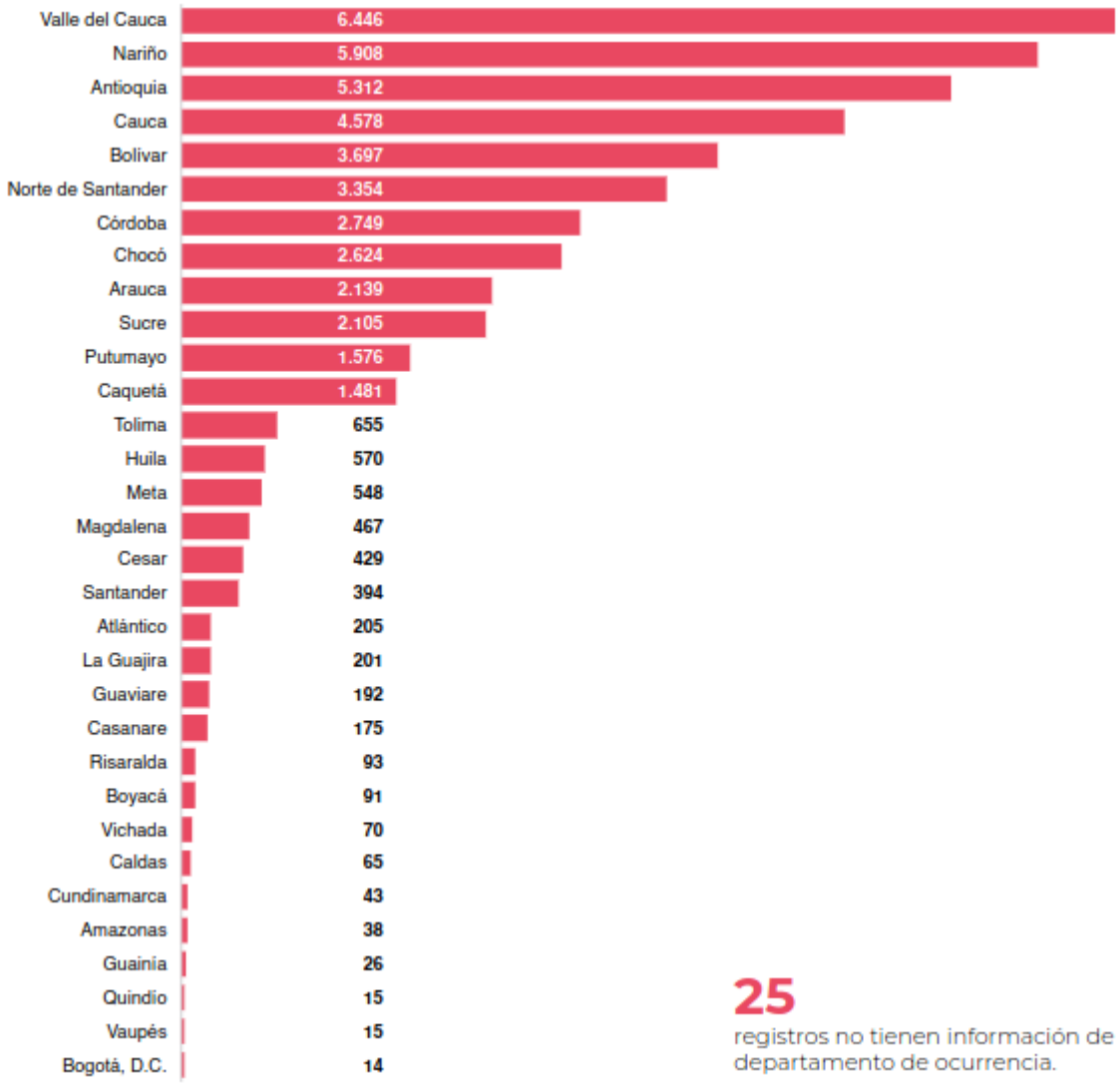


Imagen 3. Víctimas de desplazamiento por departamento en el primer semestre de 2023. Fuente: Unidad para las Víctimas. (2023). Informe sobre desplazamiento 2023.

Aun así, este proceso de restitución que se está llevando a cabo va demasiado lento. En 13 años de aplicación de esta Ley de Víctimas y Restitución de Tierras, de 6,5 millones de hectáreas de tierra despojada, solo se ha restituido el 10%. Al ritmo actual, se calcula que se tardaría más de 100 años en cumplir el objetivo del gobierno.

2.2.2. REGULARIZACIÓN DE TIERRAS

Paralelamente a la restitución de tierras, la regularización de tierras se centra en la formalización de la propiedad para aquellas personas que actualmente ocupan la tierra, pero no tienen títulos oficiales que acrediten su titularidad. Este proceso es fundamental para proporcionar seguridad jurídica a los ocupantes y fomentar el desarrollo económico del país.

La falta de títulos de propiedad afecta a una gran parte de la población rural en Colombia. Muchos campesinos han ocupado tierras durante años, incluso décadas, sin contar con la documentación legal que respalde su posesión. Esta situación genera incertidumbre y vulnerabilidad, ya que los ocupantes pueden ser desalojados o enfrentar conflictos legales en cualquier momento.

3. EL MODELO LADM-COL

El modelo LADM-COL es una adaptación del estándar internacional Land Administration Domain Model (LADM) a las necesidades específicas de Colombia. Este modelo permite la integración de datos catastrales y registrales, facilitando un catastro multipropósito que puede ser utilizado para diversas aplicaciones, incluyendo la planificación territorial, la gestión de recursos naturales y la administración de tierras.

El objetivo del LADM es estandarizar la forma en que se registran, gestionan y comparten los datos de tierras a nivel global. Esto facilita la interoperabilidad entre diferentes sistemas de información de tierras, y apoya la integración de datos geoespaciales. LADM-COL, como versión colombiana de este estándar, ha sido adaptada, entre otras entidades, por el Instituto Geográfico Agustín Codazzi (IGAC).

3.1. INTEGRACIÓN DEL LADM-COL CON EL CATASTRO MULTIPROPÓSITO

El modelo LADM-COL se integra perfectamente con el catastro multipropósito, permitiendo que la información catastral sea utilizada para múltiples propósitos:

- Toma de decisiones y políticas públicas: Facilita la planificación y gestión del territorio, aportando datos esenciales para la elaboración de Planes de Ordenamiento Territorial (POT).
- Seguridad jurídica: Mejora la protección de los derechos de propiedad al proporcionar una base de datos precisa y confiable de los bienes inmuebles.
- Desarrollo del mercado inmobiliario: Fomenta la confianza en las transacciones inmobiliarias al asegurar que la información sobre la propiedad sea accesible y precisa.

3.2. APLICACIÓN DEL LADM-COL

El LADM-COL define un modelo de datos que puede ser implementado en sistemas de información geográfica (SIG) y en bases de datos catastrales. Este modelo incluye:

- Unidades Administrativas: Representan las parcelas de tierra y propiedades, incluyendo información sobre derechos y restricciones.

- Partes Interesadas: Identifican a los propietarios, ocupantes y otras partes con intereses en las propiedades.
- Documentos Legales: Registros de títulos de propiedad, hipotecas y otros documentos relacionados con la propiedad.
- Geometría: Datos geoespaciales que describen la ubicación y los límites de las propiedades.

Entre otras cosas, el LADM-COL se utiliza para realizar el levantamiento catastral, que es el proceso de identificar, medir y registrar la información sobre las propiedades. Este proceso es esencial para actualizar y mantener la base de datos catastral, asegurando que refleje con precisión la situación en el terreno.

3.3. BENEFICIOS DEL LADM-COL

La implementación del LADM-COL ofrece numerosos beneficios, entre ellos:

- Unificación del catastro y registros: El modelo LADM permite la integración bajo un estándar común de la información catastral, registros y derechos asociados. Esto promueve una gestión más eficiente y precisa del territorio.
- Articulación intersectorial en la toma de decisiones: Facilita la colaboración entre diferentes sectores y entidades al operar bajo un mismo modelo. Esto mejora la coordinación y eficacia en la planificación territorial y la gestión de recursos.
- Estímulo a la actualización de información geográfica digital: El modelo invita a las entidades a estandarizar y disponer la información geográfica de acuerdo a sus parámetros. Esto promueve la modernización y la integración digital en la gestión catastral y territorial.
- Mejora en la transparencia y eficiencia económica: Al establecer un modelo común, se facilita el intercambio de información entre sectores y se incrementa la transparencia en el mercado inmobiliario y en las cuestiones fiscales. Esto contribuye a una gestión más justa y equitativa del territorio.

3.4. IMPLEMENTACIÓN DEL LADM-COL

En Colombia, la Unidad Administrativa Especial de Catastro Distrital (UAECD) y el Instituto Geográfico Agustín Codazzi (IGAC) son las entidades encargadas de implementar y mantener el LADM-COL. Estas instituciones trabajan en conjunto para asegurar que el catastro multipropósito sea una herramienta efectiva para la administración de tierras en el país.

La implementación del LADM-COL también incluye la capacitación de técnicos y funcionarios en el uso de herramientas SIG y la gestión de datos catastrales, así como la modernización de las infraestructuras tecnológicas necesarias para soportar el modelo.

4. APLICACIÓN METATIERRASCOL-APP, Y COMPARACIÓN CON ARCGIS FIELD MAPS

De nuevo, el objetivo de esta aplicación es desarrollar una solución móvil que facilite la toma de datos en campo por parte de los campesinos en Colombia. Actualmente, existen soluciones basadas en Esri, que son reconocidas por su funcionalidad avanzada y robustez, pero a menudo resultan caras, y pueden no estar al alcance de todos los usuarios.

Un ejemplo puede ser ArcGIS Field Maps, una solución avanzada, desarrollada por Esri que permite la recopilación y gestión de datos de campo con un conjunto de herramientas integradas en el ecosistema ArcGIS. Sus características principales incluyen:

- Recopilación y Edición de Datos en Campo: Permite a los usuarios capturar datos espaciales y atributos directamente desde dispositivos móviles, y realizar ediciones en tiempo real.
- Mapas y Capas Interactivas: Ofrece acceso a mapas detallados y capas de datos que se pueden personalizar según las necesidades del usuario.
- Soporte Offline: Los usuarios pueden descargar mapas y datos para su uso sin conexión, y sincronizarlos automáticamente cuando se restablece la conectividad.
- Integración con ArcGIS Online y ArcGIS Enterprise: Facilita la integración con otras aplicaciones y servicios de Esri, permitiendo una gestión centralizada y colaborativa de la información.
- Análisis Geoespacial Avanzado: Proporciona herramientas de análisis geoespacial para la interpretación y visualización avanzada de los datos recolectados.

Imagen 4. Logo de la app ArcGIS Field Maps

La aplicación desarrollada utiliza SQLite, para el almacenamiento local de datos, lo que permite a los usuarios recoger información en campo incluso sin conexión a internet. Las principales características de nuestra aplicación incluyen:

- Recopilación de datos en campo: Permite a los campesinos registrar información sobre sus parcelas de tierra, interesados (propietarios y usuarios), y unidades espaciales (datos GPS) de manera intuitiva y eficiente.
- Almacenamiento local persistente: Los datos se almacenan localmente en el dispositivo móvil, garantizando su disponibilidad incluso cuando la aplicación se cierra. En dispositivos Android o iOS, la base de datos persiste hasta que la aplicación se desinstala.
- Sincronización de datos: Una vez que el dispositivo tiene acceso a internet, los datos recogidos pueden ser sincronizados con un servidor central, asegurando la actualización y centralización de la información.

A continuación, se muestran dos tablas comparativas, una sobre aspectos generales de las aplicaciones y otra sobre funcionalidades que se incluyen en cada una.

Aspecto	MetaTierras App	ArcGIS Field Maps
Licencia	Gratuita, utilizando tecnologías de código abierto como SQLite y Angular	Requiere suscripción a ArcGIS Online o ArcGIS Enterprise
Implementación	Costos iniciales limitados al desarrollo y configuración	Puede ser costoso dependiendo de la configuración y personalización necesarias
Mantenimiento	Sin costos de licencias recurrentes; gastos de mantenimiento y actualizaciones según las necesidades del proyecto	Costos recurrentes significativos para soporte, actualizaciones y servicios adicionales
Soporte Técnico	Dependiente del equipo de desarrollo propio	Incluido con la suscripción a ArcGIS; acceso a soporte técnico de Esri
Actualizaciones	Gestionadas por el equipo de desarrollo propio	Incluidas con la suscripción a ArcGIS; actualizaciones regulares proporcionadas por Esri
Capacitación	Necesidad de formación específica para el equipo de desarrollo y usuarios finales	Programas de capacitación y recursos de aprendizaje proporcionados por Esri (a menudo con costo adicional)

Tabla 1. Comparación general de la app ArcGIS Field Maps con la creada para el proyecto.

Funcionalidad	MetaTierras App	ArcGIS Field Maps
Recopilación de Datos en Campo	Sí	Sí
Edición de Datos en Tiempo Real	Sí	Sí
Mapas y Capas Interactivas	Básicas	Avanzadas
Sincronización de Datos	Manual o programada	Automática con ArcGIS Online/Enterprise
Integración con Otros Sistemas	Flexible	Media, preferentemente dentro del ecosistema Esri
Análisis Geoespacial Avanzado	No	Sí

Personalización	Total	Media, sujeto a estructura Esri

Tabla 2. Comparación de funcionalidades entre ArcGIS Field Maps y la app del proyecto MetaTierrasCol-app.

Teniendo en cuenta estas dos tablas elaboradas, está claro que ArcGIS Field Maps de Esri ofrece un conjunto robusto y avanzado de herramientas para la recopilación y gestión de datos de campo, incluyendo funcionalidades como edición de datos en tiempo real, mapas y capas interactivas avanzadas, análisis geoespacial avanzado e integración automática con otros servicios de ArcGIS. Sin embargo, estas ventajas vienen con un costo significativo. La suscripción a ArcGIS Field Maps requiere una cuenta de ArcGIS Online o ArcGIS Enterprise, cuyo precio comienza en aproximadamente $500 USD por usuario al año, sin incluir costos adicionales por soporte técnico y capacitación.

Por otro lado, la aplicación MetaTierrasCol-app es completamente gratuita y utiliza tecnologías de código abierto como SQLite y Angular, lo que elimina los costos recurrentes asociados con las licencias. Aunque puede carecer de algunas de las capacidades avanzadas de análisis y automatización que ofrece ArcGIS Field Maps, proporciona todas las funcionalidades esenciales para la recopilación de datos en campo de manera efectiva y eficiente.

En resumen, si bien ArcGIS Field Maps es una solución poderosa y completa, su alto costo puede ser prohibitivo para muchos usuarios. MetaTierrasCol-app, siendo gratuita y altamente adaptable, representa una alternativa viable y económica, ideal para los campesinos en Colombia y otros usuarios con necesidades específicas de recopilación de datos en campo.

5. TECNOLOGÍAS UTILIZADAS

Como se ha mencionado al principio, todas las tecnologías que se han utilizado para desarrollar esta app son gratuitas, y son:

5.1. ANGULAR

Angular es un framework de desarrollo multiplataforma de código abierto. Se utiliza para crear aplicaciones móviles y web con un enfoque modular y una estructura bien definida. Algunas de las características clave de Angular son:

Imagen 5. Logo del framework Angular. url: angular.dev/overview

. Componentes: Angular utiliza componentes reutilizables, facilitando la organización y mantenimiento del código.

- Data binding.

- Servicios y dependendencias: Angular utiliza la inyección de dependencias, permitiendo gestionar mejor las dependencias entre los distintos componentes de la aplicación
- Routing: Sistema que facilita la navegación entre las distintas vistas de la aplicación.

En este proyecto, Angular se ha utilizado para desarrollar el front-end, es decir, tanto la interfaz del usuario (UI) cómo la lógica que se ocupa de la comunicación con el back-end o la descarga de los datos.

5.2. CAPACITOR

Capacitor es una plataforma que se usa para construir aplicaciones móviles. Está desarrollado por el equipo de Ionic y permite acceder a funcionalidades nativas del dispositivo de manera sencilla. En esta app se ha usado Capacitor para integrar estas funcionalidades nativas cómo la captura de datos georreferenciados utilizando el GPS del móvil. Las características más importantes de Capacitor serían:

Imagen 6. Logo de Capacitor. url: capacitorjs.com/docs

- Acceso a API Nativas: Capacitor proporciona una API para acceder a las funcionalidades del dispositivo como la cámara, GPS, almacenamiento, etc.
- Compatibilidad multiplataforma: Permite que la aplicación pueda ejecutarse tanto en web cómo en Android o iOS utilizando el mismo código base.
- Plugins: Capacitor soporta una gran cantidad de plugins cómo el usado en esta app de 'Geolocation', plugin que permite obtener la posición actual usando el GPS del dispositivo, junto con otros parámetros.

5.3. SQLITE

SQLite es una biblioteca de software que proporciona un sistema de gestión de bases de datos relacional embebido. Es muy popular por su eficiencia y su capacidad para funcionar sin necesidad de un servidor de bases de datos. Algunas de sus características son:

Imagen 7. Logo de SQLite. url: sqlite.org/

- Ligereza: SQLite es muy ligero, aunque puede ser un poco complicado su integración en aplicaciones móviles y webs.
- Independencia de Servidor: No requiere un servidor separado para gestionar la base de datos.
- Alto Rendimiento: A pesar de su tamaño reducido, SQLite ofrece un rendimiento excelente para la mayoría de aplicaciones móviles.

En este proyecto, SQLite se ha utilizado para el almacenamiento local de datos cuando no hay conexión a Internet. Los datos capturados por los usuarios se almacenan en una base de datos SQLite en el dispositivo y se sincronizan con el servidor cuando hay conectividad disponible.

5.4. ANDROID STUDIO

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android. Es desarrollado por Google y proporciona un conjunto completo de herramientas para la construcción de aplicaciones móviles, incluyendo:

Imagen 8. Logo de Android Studio. url: developer.android.com/studio?hl=es-419

Editor de Código: Un potente editor de código con soporte para autocompletado, refactorización y análisis de código.

- Emulador de Android: Permite probar y depurar aplicaciones en un entorno simulado que imita dispositivos Android reales.
- Gestor de SDK: Facilita la instalación y actualización de las herramientas y bibliotecas necesarias para el desarrollo de aplicaciones Android.
- Integración con Version Control: Soporte para sistemas de control de versiones como Git, lo que facilita la colaboración y el manejo de versiones del proyecto.

En este proyecto, Android Studio ha sido utilizado como el entorno principal para compilar, probar, depurar la aplicación móvil y generar la apk que se usa finalmente.

5.5. OPENLAYERS

OpenLayers es una biblioteca JavaScript de código abierto utilizada para mostrar mapas dinámicos en navegadores web. Permite la creación y visualización de mapas interactivos y personalizables, ofreciendo una amplia gama de funcionalidades geoespaciales. Algunas de las características clave de OpenLayers son:

Imagen 9. Logo de OpenLayers.
url: openlayers.org/

- Capas de Mapa: Permite la superposición de múltiples capas de datos geográficos, como capas de vectores y rásteres, facilitando la visualización y análisis de datos complejos.
- Interacciones y Controles: Proporciona herramientas para interactuar con los mapas, como controles de navegación, herramientas de dibujo y selección de características geográficas.
- Compatibilidad: Soporta una gran variedad de formatos de datos geoespaciales y servicios de mapas, incluyendo WMS, WFS, GeoJSON, entre otros.
- Proyecciones: Permite trabajar con diferentes sistemas de coordenadas y proyecciones geográficas, garantizando la precisión en la visualización de datos espaciales.

En este proyecto, OpenLayers se ha utilizado para integrar y gestionar la visualización de un mapa interactivo en la aplicación. Gracias a OpenLayers, los usuarios pueden visualizar en tiempo real los puntos georreferenciados obtenidos mediante el GPS, ya que esta biblioteca permite crear y dibujar estos puntos directamente en el mapa.

6. DISEÑO Y FUNCIONALIDADES DE LA APP

6.1. METODOLOGÍA

En el siguiente diagrama se muestra el proceso general que se ha llevado a cabo en este proyecto.

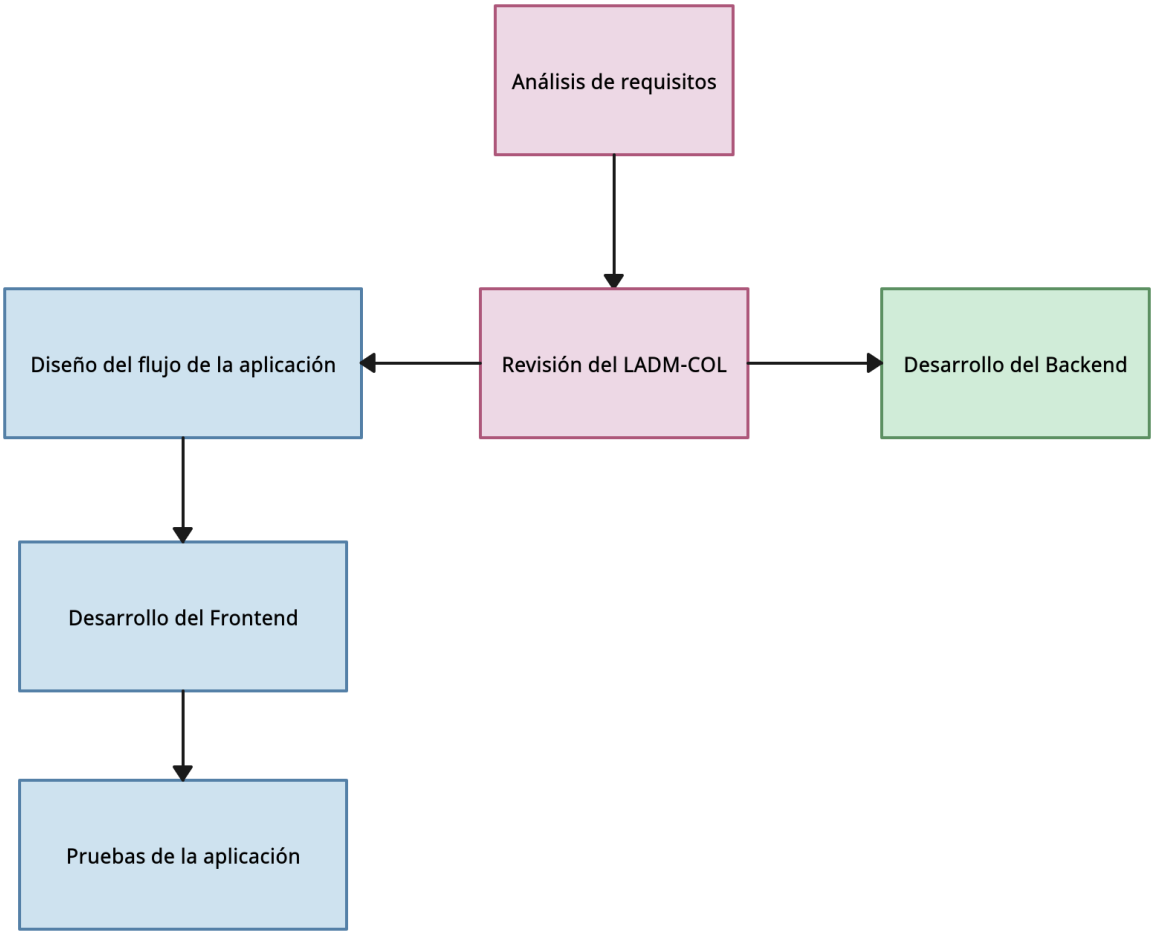


Imagen 10. Esquema de desarrollo seguido

El proyecto comenzó con un análisis de los requisitos que desde la ONG necesitaban para poder trabajar con la app, se definieron los objetivos y el alcance del proyecto al que se llegaba aspirar.

El tutor, Joaquín Gaspar Mora, se encargó del estudio y comprensión del estándar LADM-COL para asegurar una correcta implementación y tras la revisión de la estructura de datos y los requisitos de este modelo comenzó a desarrollar el backend. Implementó el servidor y la base de datos que se usa en la aplicación, y creó la API para la comunicación entre la aplicación y este servidor.

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos que permiten la comunicación entre diferentes softwares. En este proyecto, la API es fundamental para la interacción entre el frontend (la aplicación móvil) y el backend (el servidor). Los endpoints son las URLs específicas a las que se pueden hacer solicitudes para realizar diversas operaciones, como obtener datos de usuarios o enviar información de GPS. Los datos se envían en formato JSON (JavaScript Object Notation) usando el método POST, lo que permite enviar datos estructurados al servidor. A su vez, la API responde con otro JSON que contiene la información solicitada o el resultado de la operación.

Paralelamente al desarrollo del backend, se creó el diseño del flujo detallado entre las pantallas de la aplicación, incluyendo la interacción que debía haber entre ellas. Creado este flujo, la implementación del frontend se pudo hacer de manera estructurada. Se fueron creando los componentes para cada una de las pantallas usando Angular y Capacitor, integrando SQLite para el almacenamiento local de los datos, etc.

Conforme se iba terminando cada uno de los componentes, se realizaban pruebas. Semanalmente se organizaban reuniones donde normalmente nos encontrábamos el tutor Gaspar y yo. En estas reuniones se comentaba el avance de la aplicación y se señalaban errores que podían haber, o se daba el visto bueno del componente creado. Otras veces se reunía todo el equipo, o la mayor parte, incluida la tutora Carmen y el equipo de Colombia. En estas reuniones se discutían temas generales de diseño de la aplicación o sobre el modelo de datos que se debía seguir.

6.2. ELECCIÓN DE LOS DATOS A MANEJAR

Para ilustrar la complejidad y la extensión del modelo LADM-COL, se incluye a continuación una imagen general del mismo. Debido al gran tamaño y detalle de la imagen, no se aprecian todos los elementos en esta vista. Sin embargo, la imagen sirve para mostrar la amplitud y alcance del modelo. Para una visualización más detallada y para descargar la imagen completa, se puede mediante el enlace en el pie de la imagen.

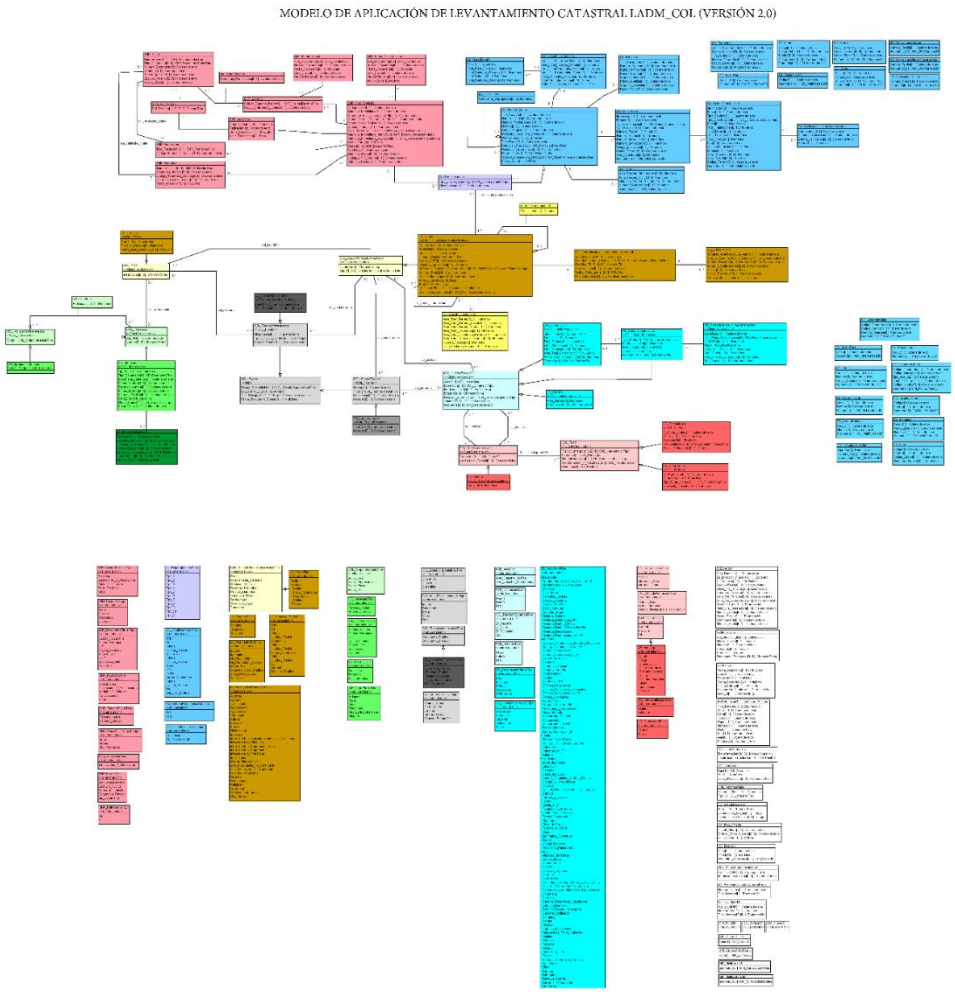


Imagen 11. Modelo Aplicación LADMCOL Levantamiento Catastral V2_0

Fuente: https://gitlab-ladm-col.igac.gov.co/root/LADM_COL/-/blob/lev_cat_v2/Catastro_Multiproposito/2_Aplicacion/1_Levantamiento_Catastral/Modelo_Aplicacion_LADMCOL_Levantamiento_Catastral_V2_0.pdf

Tras haber estudiado el modelo LADM-COL y haber revisado con el equipo colombiano los datos con los que se necesitaba trabajar, se llegó a la conclusión de que serían los siguientes:

6.2.1. DATOS DEL INTERESADO

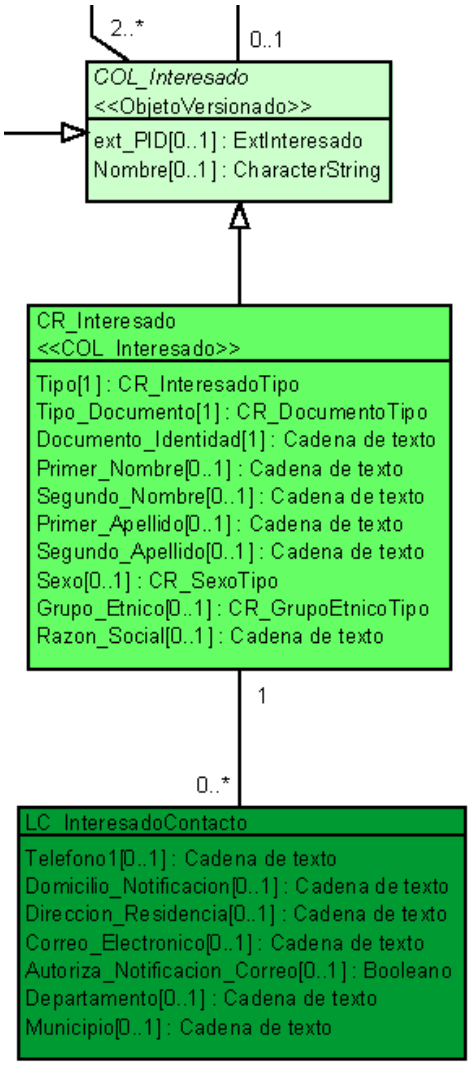


Imagen 12. Modelo de datos de interesados LADM-COL

Los interesados se relacionan con un predio y uno o varios derechos. Según el modelo LADM-COL, se requiere gestionar una amplia gama de datos. Sin embargo, en este proyecto, gestionamos únicamente los datos mínimos necesarios según el modelo, junto con algunos datos adicionales solicitados por la ONG Fundación Forjando Futuros (FFF). Los datos que gestionamos incluyen:

- **Autoriza_procesamiento_datos_personales**: Booleano. Esto es un checkbox que hay que marcar. En caso que no se marque no se deja pasar a la siguiente pantalla. En España esto es obligatorio. Se informa al interesado de por qué se almacenan sus datos, hasta cuándo, y cómo borrar sus datos. El checkbox debe aparecer debajo de un texto donde se detalla:

    • Quién va a gestionar sus datos
    • Fines y legitimación del tratamiento. Se detalla que es en interés del usuario.
    • Criterios de conservación de los datos. ¿Cuánto tiempo se van a conservar los datos?

• Comunicación de los datos: Se pasarán los datos a la ANT, Gestor catastral, etc.
• Derechos que tiene el usuario. Básicamente es poder eliminar sus datos.
• Datos de contacto para ejercer el derecho de eliminación de sus datos.

- **Tipo: CR_InteresadoTipo**. Posibles valores: Persona_Natural, Persona_Juridica
- **Tipo_Documento: CR_DocumentoTipo**. Posibles valores: Cedula_Ciudadania, Cedula_Extranjeria, NIT, Tarjeta_Identidad, Registro_Civil, Secuencial, Pasaporte, Documento_Identidad: texto
- **Primer_Nombre**: texto
- **Segundo_Nombre**: texto
- **Primer_Apellido**: texto
- **Segundo_Apellido**: texto
- **Sexo**: CR_SexoTipo. Posibles valores: Masculino, Femenino, Sin_Determinar
- **Grupo_Etnico**. Posibles valores: Indigena, Rrom, Raizal, Palenquero, Negro_Afrocolombiano, Ninguno
- **Telefono1**: texto
- **Correo_Electronico**: email
- **Autoriza_Notificacion_Correo**: Booleano
- **Departamento**: Lista de departamentos
- **Municipio**: Lista de Municipios

6.2.2. DATOS DEL PREDIO

Se entiende por predio a la unidad de propiedad inmobiliaria, con o sin construcciones y/o edificaciones, perteneciente a personas naturales o jurídicas. Esta unidad de propiedad puede estar destinada a diversos usos, como residencial, comercial, industrial, agrícola, entre otros.

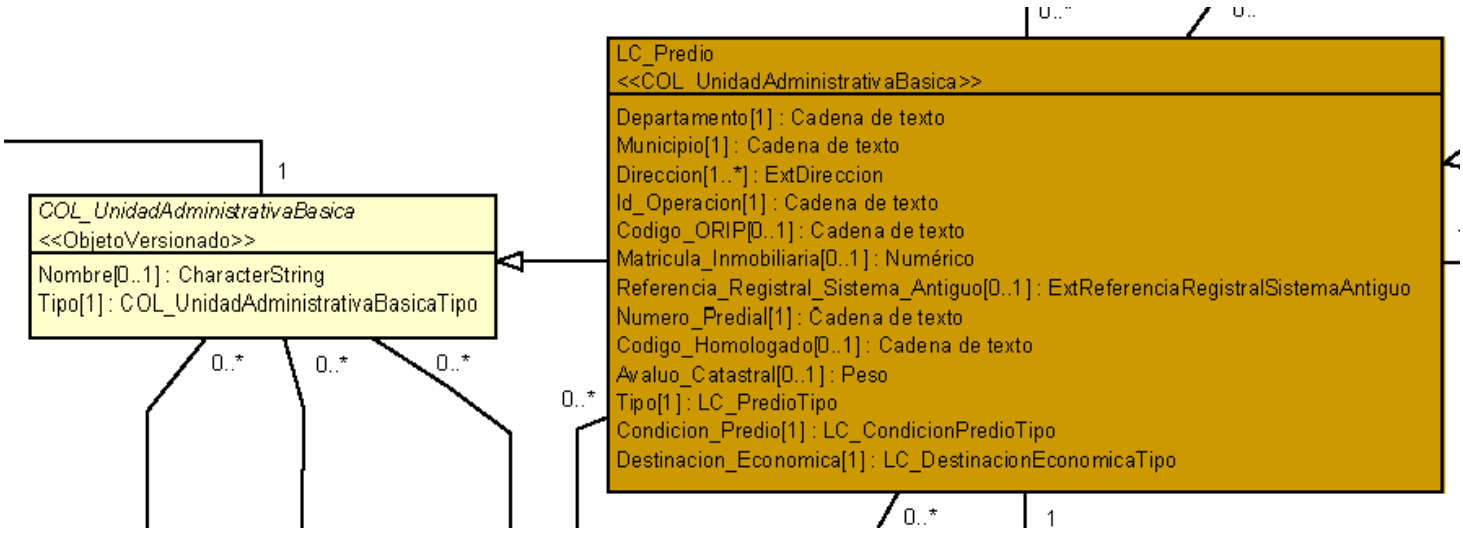


Imagen 13. Modelo de datos de unidades administrativas LADM-COL

Aunque en el modelo se plantean más campos, se decidió que para la aplicación solo interesaban los campos de:

- **Nombre**: Texto

- **Departamento**: Lista de departamentos
- **sector_predio**. Posibles valores: Norte, Sur, Este, Oeste
- **Municipio**: Lista de Municipios
- **Vereda**: Texto
- **numero_predial**: Texto
- **tipo: LC_PredioTipo**. Posibles valores: Baldio, Fiscal_Patrimonial, Uso_Publico, Publico, Privado
- **complemento**: complemento a la dirección. Texto

6.2.3. GEOMETRÍA DEL PREDIO

A cada uno de los predios se le asignará uno o varios polígonos cerrados que representan la geometría de este. El modelo tiene las siguientes clases para la geometría.

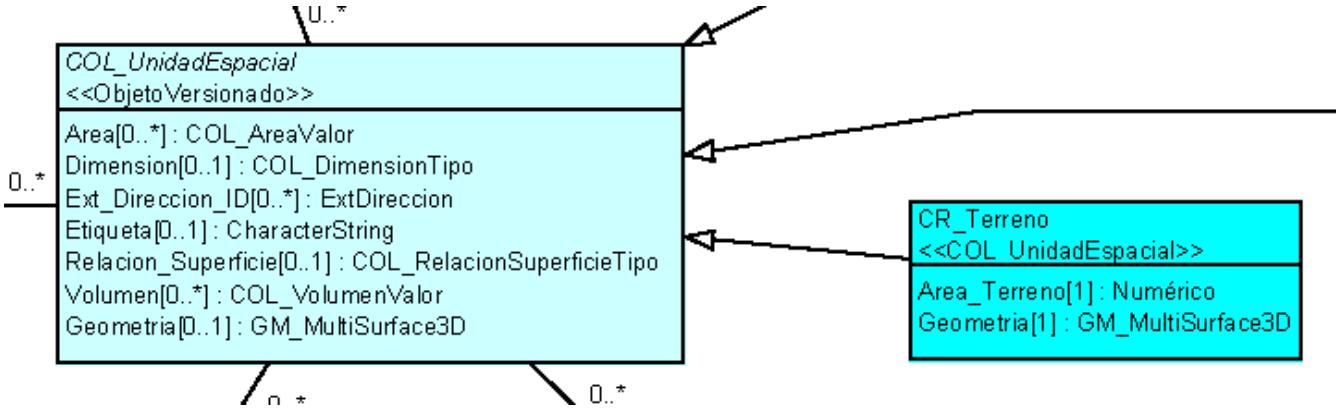


Imagen 14. Modelo de datos de unidades espaciales LADM-COL

Se toma la decisión de que para la delimitación de fincas rústicas solo se necesita gestionar desde la app la geometría:

- geometría: el formato proporcionado es GeoJSON.

6.3. DISEÑO DE LOS MENÚS

Inicialmente, se creó un diseño sobre el flujo de las diferentes pantalla y menús que abarcaba más de los requisitos mínimos que se buscaba, como, por ejemplo, una pantalla para la configuración de los emails a los que se le enviarán los datos, pero desde un principio se dejó claro que eso serían implementaciones futuras, y que no eran realmente necesarias abarcar en la primera versión de la aplicación.

Estos diseños fueron cambiando según surgían nuevas ideas o requisitos.

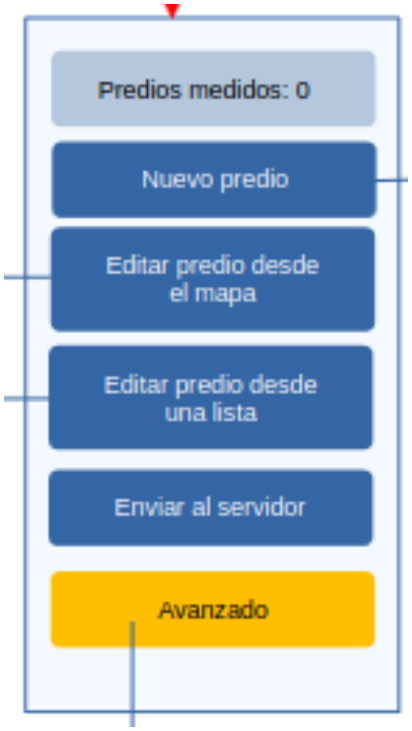


Imagen 15. Diseño inicial pantalla principal

Por ejemplo, en este primer diseño de la pantalla principal se mostraba el número de predios guardados junto con las opciones de editar los predios desde un mapa o desde una lista. Estas opciones se simplificaron en el diseño final mostrando directamente en la pantalla principal la lista de los predios que se iban generando, con la opción de editarlos directamente al presionar en ellos.

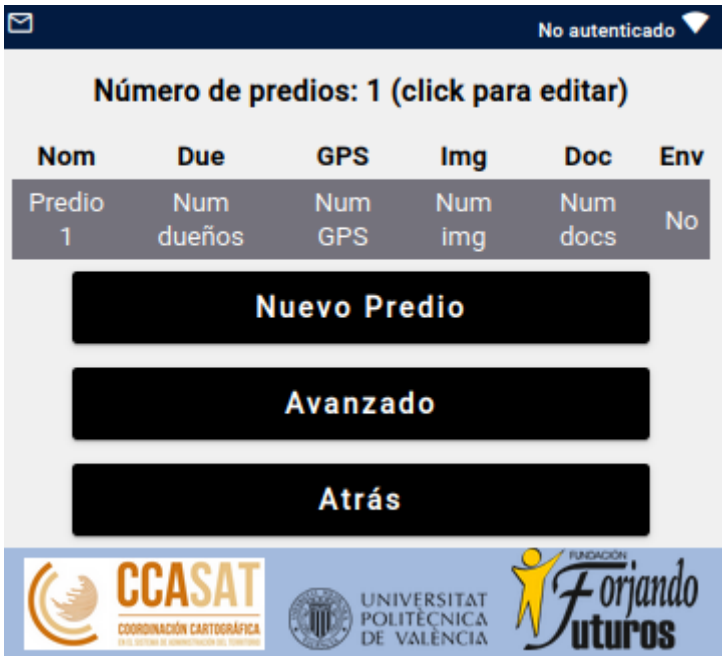


Imagen 16. Diseño final pantalla principal

Inicialmente, la opción de enviar los datos al servidor enviaba todos los predios guardados directamente, al final se decidió que se enviaran los predios individualmente, por lo que el botón de enviar al servidor pasa a la pantalla del predio en vez de a la inicial.

Imagen 17. Diseño inicial pantalla de medición por GPS

En el diseño de la pantalla para medir con el GPS se daba la opción al principio de tomar una foto del terreno, opción que serviría para referenciar un punto medido con una imagen, lo que facilitaría la identificación del punto cuando se revisara. Esta opción se descartó para que fuera más simple ya que, al fin y al cabo, la aplicación está pensada para que los mismos campesinos la puedan usar.

Imagen 18. Diseño inicial pantalla de medición por GPS

Lo que sí que se añadió que no estaba planteado en el diseño inicial es la información sobre la precisión del teléfono en cada momento y la precisión del último punto tomado. Esto es muy importante porque la precisión de los teléfonos móviles depende de varios factores y no son tan precisos cómo un GPS topográfico por lo que con esta funcionalidad el usuario puede esperar a recibir buena señal y por lo tanto una buena precisión antes de tomar el punto. También se informa al usuario de en qué rango de precisiones están los linderos medidos.

La precisión de cada punto se almacena por separado, y se asigna al polígono la peor precisión de los puntos que forman su perímetro.

6.4. FUNCIONALIDADES DE LA APLICACIÓN

Cómo se ha estado mencionando, el objetivo es ayudar con la gestión catastral para facilitar las tareas de restitución y regularización de tierras en Colombia, para eso, y con los datos seleccionados a manejar, las funcionalidades principales de la aplicación incluyen:

6.4.1. TOMA DE DATOS DEL PREDIO

Los usuarios pueden registrar nuevos predios, ingresando la información anteriormente mencionada (nombre, departamento, tipo del predio, etc.). Estos predios pueden ser editados o eliminados según sea necesario, permitiendo mantener la base de datos actualizada y precisa, eliminando registros incorrectos u obsoletos.

Imagen 19. Pantalla de toma de datos del predio sin completar

Se han establecido campos obligatorios que deben ser completados para habilitar la opción de guardar los datos. Hasta que el usuario no complete todos los campos requeridos, el botón de guardar permanecerá desactivado.

6.4.2. TOMA DE DATOS DEL INTERESADO

La aplicación permite ingresar tantos interesados como sea necesario para un mismo predio, ya que un predio puede ser propiedad de una o varias personas. Por este motivo, uno de los

campos es para especificar el porcentaje de propiedad que corresponde a cada interesado sobre el predio en cuestión.

Imagen 20. Pantalla de toma de datos del interesado.

Al igual que con los datos del predio, para almacenar un interesado se necesita rellenar los campos obligatorios, entre ellos las casillas de autorización de notificación por correo electrónico y de tratamiento de datos personales. Por ley, es obligatorio que el usuario acepte estas autorizaciones tras la lectura de los mensajes correspondientes.

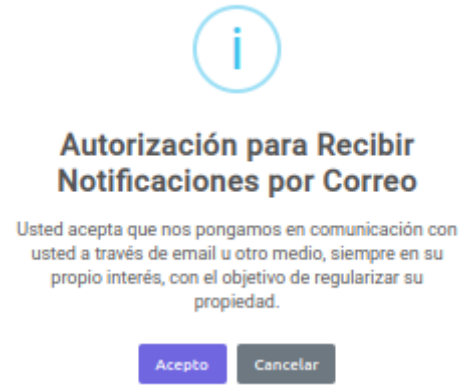


Imagen 21. Aviso para la autorización de notificación por correo.

La autorización de notificación por correo electrónico es esencial para mantener una comunicación eficiente y efectiva con los interesados. Esta opción nos autoriza a enviar notificaciones importantes relacionadas con la regularización de la propiedad, actualizaciones de estado, y cualquier otra información relevante que pueda requerir la atención del interesado

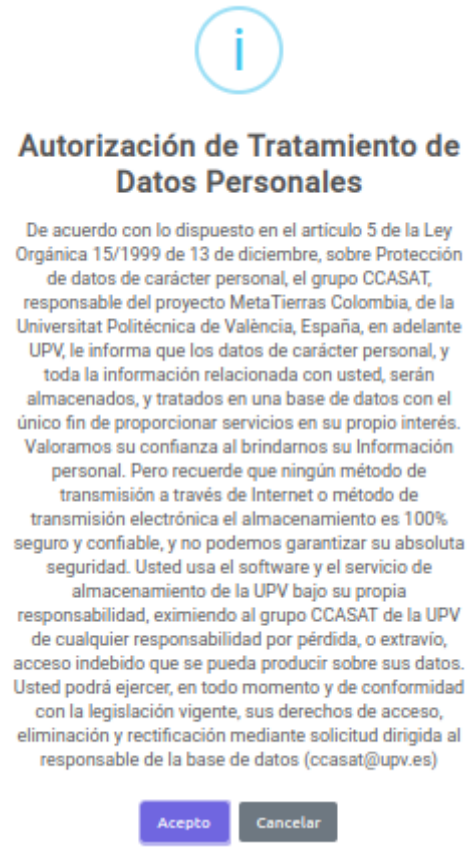


Imagen 22. Aviso para la autorización de tratamiento de datos personales.

La autorización para el tratamiento de datos personales es un requisito legal fundamental que garantiza el cumplimiento de las normativas de protección de datos, como la Ley Orgánica 15/1999 de 13 de diciembre sobre Protección de Datos de Carácter Personal. Esta autorización permite a la aplicación recoger, almacenar y utilizar los datos personales de los

interesados de manera legal y segura. Además, asegura a los usuarios que sus datos serán manejados con la máxima confidencialidad y utilizados únicamente para los fines previstos.

Antes de añadir algún interesado, hay una pantalla intermedia donde se muestra la lista de todos los que se han almacenada para ese predio hasta ahora, permitiendo hacer clic sobre ellos para editar los datos.

Imagen 23. Pantalla con la lista de interesados.

6.4.3. MEDICIONES CON EL GPS

Para cada predio, se crean una o varias unidades espaciales, estas corresponden a las mediciones que se hacen sobre un predio. Puede haber más de una unidad espacial porque un predio puede estar dividido por algún elemento de dominio público.

Entonces, a la hora de medir, se muestra en todo momento la precisión que tiene el GPS del móvil para que el usuario escoja el momento más preciso en el que tomar el punto, esto garantiza que la medición realizada sea lo más acertada y precisa posible a la realidad. (véase imagen 17).

Imagen 24. Pantalla con la lista de puntos medidos.

Es posible que el usuario decida rectificar y volver a tomar un punto o simplemente haya tomado un punto por error, es por eso que está disponible también una pantalla donde se muestran todos los puntos medidos y se puede eliminar el que quiera, o también editar la descripción del punto.

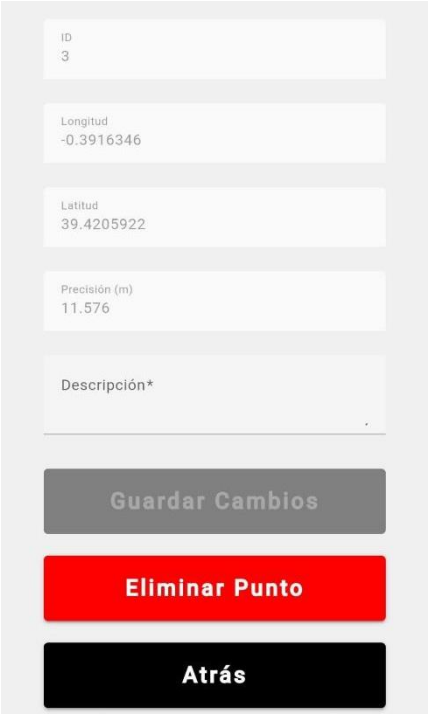


Imagen 25. Pantalla para editar o eliminar un punto.

7. DESARROLLO DE LA APP

7.1. MODELOS

En la aplicación, los modelos representan las entidades fundamentales de datos que se manejan en el sistema. Estos modelos definen la estructura de datos y las operaciones que se pueden realizar sobre ellos y se integran con los componentes y los servicios de la aplicación.

Los siguientes métodos están implementados de manera común en todos los modelos de la aplicación excepto en el de ‘messages’ y ‘user’:

- insert(): Permite insertar un nuevo registro del modelo en la base de datos.
- update(): Actualiza los datos de un registro existente en la base de datos.
- delete(): Elimina un registro del modelo de la base de datos.
- setFromModel(modelo: Modelo): Actualiza los atributos del modelo actual con los del modelo pasado como argumento.
- Otros métodos específicos según el modelo, como setFromId(), asListOfValues(), etc.

```
async insert() : Promise<void>  {  Show usages
    const q : string  = `INSERT INTO interesado (
    baunit_id, tipo_documento, documento_identidad, tipo, primer_nombre, primer_apellido,
    correo_electronico, sexo, departamento, provincia, municipio, porcentaje_propiedad,
    segundo_nombre, segundo_apellido, grupo_etnico, telefono_1, telefono_2,
    notas, estado, autoriza_notificacion_correo,
    autoriza_procesamiento_datos_personales
) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)`;

    await this.sqliteService.db.run(q, this.asListOfValues()) Promise<capSQLiteChanges>
        .then( (r:any) : void  => {
            this.id = r.changes.lastId.toString();
            sendMessages(`Interesado ${this.id} guardado`, this.messageService, this.sqliteService.snackBar);
            console.log(`Interesado ${this.id} guardado`)
        }) Promise<void>
        .catch( (err) : void  =>{
            sendMessages(err.message,this.messageService, this.sqliteService.snackBar)
        })
    await this.sqliteService.updateInteresadosList();
}
```

Imagen 26. Método 'insert' del modelo 'interesado'.

Estos métodos son fundamentales para la manipulación y gestión de datos en la aplicación, garantizando la integridad y consistencia de la información almacenada.

7.1.1. BAUNIT

El modelo ‘Baunit’ representa el predio. Contiene los atributos que describen las características y ubicación del predio y los métodos para manipular y gestionar esos datos en la base de datos.

```
//obligatorios
nombre: string = '';
tipo: LC_PredioTipo = LC_PredioTipo.Baldio;
departamento: string = '';
provincia: string = '';
municipio: string = '';
sector_predio: SectorPredio = SectorPredio.Este;
vereda: string = '';
complemento: string = '';

//opcionales. Se inicializan a '', si es undefined;
numero_predial: string = '';
numero_catastral: string = '';
longitud: string = '';
latitud: string = '';
enviado_servidor: boolean = false;
id: string = '';//el id de sqlite
```

Imagen 27. Atributos modelo 'baunit'.

7.1.2. INTERESADO

El modelo 'interesado' almacena la información de las personas relacionadas con el predio que se está registrando en el momento, incluyendo datos personales y porcentaje de propiedad

```
autoriza_procesamiento_datos_personales: boolean = false;
tipo: CR_InteresadoTipo = CR_InteresadoTipo.Persona_Natural;
tipo_documento: CR_DocumentoTipo = CR_DocumentoTipo.Cedula_Ciudadania;
documento_identidad: string = '';
primer_nombre: string = '';
primer_apellido: string = '';
sexo: CR_SexoTipo = CR_SexoTipo.Sin_Determinar;
correo_electronico: string = '';
autoriza_notificacion_correo: boolean = false;
departamento: string = '';
provincia: string = '';
municipio: string = '';
porcentaje_propiedad: number = 0;
baunit_id: string = '';

// Opcionales
id: string = '';
segundo_nombre: string = '';
segundo_apellido: string = '';
grupo_etnico: Grupo_Etnico = Grupo_Etnico.Ninguno;
telefono_1: string = '';
telefono_2: string = '';
notas: string = '';
estado: Estado = Estado.Soltero;
```

Imagen 28. Atributos modelo 'interesado'.

7.1.3. UNIDAD ESPACIAL

El modelo 'UnidadEspacial' se encarga de gestionar las unidades espaciales asociadas a un predio, es decir, todos los conjuntos de geometrías que conforman un predio. Es importante destacar que SQLite, el sistema de gestión de bases de datos utilizado en este proyecto, no tiene soporte nativo para almacenar geometrías. Por esta razón, las geometrías se guardan en campos de tipo texto en formato GeoJSON.

```
export class UnidadEspacial {  Show
    id: string = '';
    baunit_id: string = '';
    tipo: string = 'gps';
```

Imagen 29. Atributos modelo 'UnidadEspacial'.

7.1.4. CRPUNTOLINDERO

El modelo ‘crPuntoLindero’ representa cada uno de los puntos geográficos que se toman en la medición de un predio, almacenando por lo tanto la relación del punto con el predio y la unidad espacial a la que pertenece y su información geográfica.

```
export class CrPuntoLindero {  Show usages  Diego *
    id: string = '';
    baunit_id: string = '';
    unidad_espacial_id: string = '';
    tipo: string = '';
    lon: number = 0;
    lat: number = 0;
    exactitud_horizontal: number = 0;
    descripcion: string | null = '';
```

Imagen 30. Atributos modelo 'CrPuntoLindero'.

7.1.5. USER

Este modelo define la estructura de datos esperada para un usuario en la aplicación

```
export interface User{
    id: number,
    name: string,
    active: number
}
```

Imagen 31. Atributos del modelo 'User'.

7.1.6. MESSAGE

El modelo ‘message’ se usa para representar los mensajes que proporcionan retroalimentación o información al usuario respecto a operaciones realizadas o al estado general de la aplicación.

```
export class Message {  Show usages
    public ok: boolean;
    public message: string;
    public class: string;
    public state: string;
    public time: string;

    constructor(ok:string, message: string) {
      this.message = message;
      this.time = getTime();
      if (ok=='info'){
        this.ok=true;
        this.state="Info";
        this.class="alert alert-info";
      } else {
        if (ok=="true"){
            this.ok=true;
            this.state="Success";
            this.class="alert alert-success";
        } else {
          this.ok=false;
          this.state="Danger";
          this.class="alert alert-danger";
        }
    }
  }
}
```

Imagen 32. Modelo 'Message'.

7.2. COMPONENTES

En Angular, los componentes son fundamentales para la construcción de la interfaz de usuario. Cada componente se compone de una clase en TypeScript, un archivo de plantilla HTML, un archivo de estilos CSS, y opcionalmente, un archivo de pruebas. Los componentes gestionan una parte específica de la interfaz de usuario y encapsulan tanto la lógica como la presentación. Todo el código fuente se puede encontrar en https://github.com/joamona/metatierrascol-app.

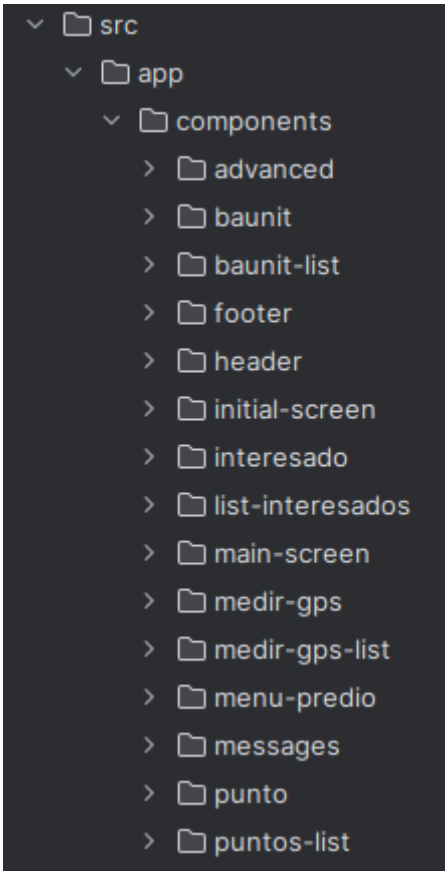


Imagen 33. Componentes dentro de la aplicación en Angular.

Componente	**Descripción**
advanced	Componente para funcionalidades avanzadas de la aplicación.
baunit	Gestiona la información de un predio específico.
baunit-list	Muestra una lista de predios registrados.
footer	Pie de página común para toda la aplicación.
header	Encabezado común para toda la aplicación.
initial-screen	Pantalla inicial que se muestra al abrir la aplicación.
interesado	Gestiona la información de los interesados en un predio.
list-interesado	Muestra una lista de interesados registrados para un predio.
main-screen	Pantalla principal de la aplicación, donde se navega a otras secciones.

medir-gps	Componente para la funcionalidad de medir un predio usando GPS.
medir-gps-list	Muestra una lista las unidades espaciales.
menu-predio	Menú de opciones relacionadas con la gestión de predios (envío, eliminación, etc.).
messages	Muestra mensajes de retroalimentación y notificaciones al usuario.
punto	Gestiona la información de un punto geográfico en la medición de un predio.
puntos-list	Muestra una lista de puntos geográficos registrados en la medición de un predio.
mapa	Se crea y se configura el mapa usando Openlayers

Tabla 3. Descripción breve de los componentes.

1. **baunit**
    - **Función**: Gestiona la información detallada de un predio específico.
    - **Descripción**: Este componente permite ver, editar y actualizar la información de un predio. Utiliza servicios para interactuar con la base de datos y actualizar la información.
    - **Código HTML**: Muestra un formulario con los campos de información del predio.
    - **Código TypeScript**: Contiene la lógica para manejar los datos del formulario, incluyendo validaciones y envío de datos.
2. **interesado**
    - **Función**: Gestiona la información de los interesados asociados a un predio.
    - **Descripción**: Permite añadir, editar y eliminar interesados. Cada interesado incluye información personal y el porcentaje de propiedad.
    - **Código HTML**: Formulario para la entrada de datos del interesado.
    - **Código TypeScript**: Lógica para manejar los datos del formulario, interactuar con la base de datos y actualizar la lista de interesados.
3. **medir-gps**
    - **Función**: Permite la medición de predios usando GPS.
    - **Descripción**: Este componente captura las coordenadas GPS y las almacena como puntos geográficos asociados a un predio.
    - **Código HTML**: Interfaz para iniciar y detener la captura de coordenadas.
    - **Código TypeScript**: Lógica para interactuar con el dispositivo GPS, capturar datos y almacenarlos.
4. **messages**
    - **Función**: Muestra mensajes de retroalimentación y notificaciones.
    - **Descripción**: Este componente se utiliza para mostrar mensajes informativos, de éxito o de error al usuario.
    - **Código TypeScript**: Lógica para recibir y mostrar mensajes desde diferentes partes de la aplicación.

5. **mapa**
    - **Función**: Visualiza y maneja datos geoespaciales en un mapa interactivo.
    - **Descripción**: Este componente utiliza OpenLayers para mostrar un mapa donde se pueden visualizar y dibujar las geometrías de los predios y puntos capturados mediante GPS. Permite interactuar con diferentes capas y realizar mediciones geoespaciales precisas.
    - **Código HTML**: Contiene el contenedor del mapa.
    - **Código TypeScript**: Configura el mapa, define las capas y estilos, y añade interacciones de dibujo.

Componentes Comunes y Listas

- **header y footer**
    - **Función**: Elementos comunes de la interfaz presentes en todas las pantallas de la aplicación.
    - **Descripción**: Contienen elementos como el título de la aplicación, navegación y pie de página con información general.
- **Componentes "-list"**
    - **Función**: Muestran listas de entidades (predios, interesados, mediciones, puntos, etc.).
    - **Descripción**: Estos componentes tienen una estructura similar. Utilizan una tabla o lista para mostrar múltiples registros y permiten acciones como ver detalles, editar y eliminar.
    - **Código HTML**: Plantilla para mostrar la lista de elementos.
    - **Código TypeScript**: Lógica para manejar la carga, visualización y manipulación de la lista de elementos.

7.3. SERVICIOS

En Angular, los servicios son una pieza clave para compartir datos y funcionalidades entre diferentes partes de la aplicación. Se utilizan para encapsular la lógica de negocio, manejar la comunicación con APIs, gestionar el estado de la aplicación. Las siguientes secciones describen los servicios creados:

7.3.1. CONFIGURACIÓN DE SQLITE.

Siguiendo la documentación, ha sido necesaria la creación de tres servicios para la configuración de SQLite en entornos nativos y navegadores web, el primero “NativeSQLiteService”, se usa para la gestión de la conexión y operaciones con la base de datos SQLite en plataformas móviles.

```
export class NativeSqliteService {
  public databaseName: string='metatierrascol';
  public sqliteConnection: SQLiteConnection;
  public db!: SQLiteDBConnection;
  public  isConnection: boolean | undefined = false;
  public messages:string[]=[];//solo para hacer las primeras pruebas

  constructor(public messageService:MessageService){   no usages
    this.sqliteConnection = new SQLiteConnection(CapacitorSQLite);
  }

  async initializeDb() : Promise<SQLiteDBConnection> {  Show usages

    sendMessages( message: 'Creando la conexión nativa Sqlite', this.messageService);
    this.db = await this.sqliteConnection.createConnection(this.databaseName, encrypted: false, mode: 'no-encryption', version: 1, readonly: false);
    sendMessages( message: 'Conexión Sqlite nativa creada', this.messageService);

    return this.db;

  }
}
```

Imagen 34. Función 'initializeDB' del servicio 'NativeSqliteService'.

La funcionalidad principal es mediante el método ‘initalizeDb()’, que crea y abre una conexión con la base de datos SQLite.

El segundo servicio necesario para la configuración de SQLite es “sqlite.service.ts”, este servicio se encarga de la mayor parte de operaciones con SQLite, incluyendo la creación y gestión de las tablas, así como la actualización de los objetos que representan los datos almacenados. Las funciones principales son:

- **Creación de Tablas**: Define y crea las tablas necesarias en la base de datos, como baunit, interesado, unidad_espacial, y cr_puntolindero.
- **Actualización de Listas**: Métodos como updateBaunitList(), updateInteresadosList(), updateUnidadEspacialList(), y updateCrPuntoLinderoList() se utilizan para sincronizar las listas de objetos en la aplicación con los datos almacenados en la base de datos.
- **Eliminación y Actualización de Datos**: Métodos como deleteBaunit() y marcarPredioComoEnviado() permiten la manipulación de registros específicos en la base de datos.

Imagen 35. Función (parcial) para la creación de las tablas de la base de datos SQLite.

Dentro de los servicios relacionados con SQLite, este es el más importante ya que es el que permite interactuar realmente con la base de datos, asegurando que todas las operaciones CRUD (crear, leer, actualizar y eliminar) se realicen de manera eficiente.

El último servicio relacionado con SQLite es el de “web-sqlite.service.ts”. Este servicio es necesario para la utilización de esta base de datos en entornos web.

```
export class WebSqliteService {
    sqlite!: SQLiteConnection;
    isService: boolean = false;
    platform!: string;
    sqlitePlugin: any;
    native: boolean = false;

    constructor() {  no usages
    }
    /**
     * Plugin Initialization
     */
    initializePlugin(): Promise<boolean> {  Show usages
        return new Promise ( executor: resolve => {
            this.platform = Capacitor.getPlatform();
            if(this.platform === 'ios' || this.platform === 'android') this.native = true;
            this.sqlitePlugin = CapacitorSQLite;
            this.sqlite = new SQLiteConnection(this.sqlitePlugin);
            this.isService = true;
            resolve( value: true);
        });
    }
```

Imagen 36. Función 'initializePlugin' del servicio 'WebSqliteService'.

La funcionalidad principal se apoya en el método ‘initializePlugin()’, que configura el entorno para usar SQLite en una plataforma web, determinando si la aplicación está en un entorno nativo o web y ajustando la configuración en consecuencia.

7.3.2. AUTENTICACIÓN

Otro de los servicios más importantes es el de “auth.service.ts” que sirve para manejar la autenticación de los usuarios, permitiendo el acceso seguro a la API de Django que soporta la aplicación.

Mediante las funciones “almacenaUrlyTokenEnAlmacen” y “borrarUrlyTokenDelAmlacen” se almacena y recupera las credenciales del usuario utilizando el plugin de Capacitor Preferences en el dispositivo. Esto asegura que los datos necesarios para la autenticación del usuario persistan incluso cuando la aplicación se cierra o se reinicia.

```
public checkAuthorizationToken() : void {  Show usages
  if (this.urlDjangoApi == '' || this.token == ''){
    var message : Message =new Message( ok: 'info', message: 'No hay token en el dispositivo. Necesario autenticarse');
    this.messageService.add(message);
    return
  }

  var  headers: HttpHeaders;
  headers = new HttpHeaders( headers: { 'Content-Type': 'application/json', 'Authorization': 'Token ' + this.token});
  this.httpClient.get(
     url: this.urlDjangoApi + 'core/knox/is_valid_token/',
       options: {headers: headers, responseType : 'json',
            reportProgress: false}).subscribe(
         observerOrNext: {
           next: ((data: any) : void =>{
             var message : Message =new Message( ok: 'info', message: 'Datos recibidos: ' + JSON.stringify(data));
             this.messageService.add(message);
             //{"detail":"Valid token.","username":"admin","groups":["admin"]}
             this.setData( isTokenValid: true,data.username,data.groups,this.token,this.urlDjangoApi);
           }),
           error: ((error: any) : void =>{
             manageServerErrors(error,this.messageService,this.snackBar);
             if (error.status == 401){
               if (error.error.detail=="Invalid token."){
                 this.borrarTokenDelAlmacen().then((r : void ) : void =>{
                   sendMessages( message: 'La sesión estaba caducada. Debe iniciar sesión',this.messageService,this.snackBar);
                 });
               }
             }
           })
         });
}
```

Imagen 37. Función 'checkAuthorizationToken' del servicio de autenticación.

“checkAuthorizationToken” verifica la validez del token de autenticación almacenado mediante una solicitud GET a la API de Django. Si el token es válido, actualiza el estado de autenticación de la aplicación; si no, notifica al usuario para que vuelva a autenticarse. Esto asegura que solo usuarios autorizados puedan acceder a los datos protegidos y mediante la función “clearDataFromService” se limpia los datos del usuario al cerrar sesión.

7.3.3. COMUNICACIÓN CON EL SERVIDOR

El servicio “ServerService” es el responsable de manejar la comunicación entre la aplicación y el servidor. Este servicio facilita el envío y la recepción de datos mediante HTTP, asegurando que las solicitudes incluyan la autenticación necesaria.

Los dos métodos que se usan de este servicio son el método ‘post’, que realiza una solicitud POST al servidor con datos que recibe cómo parámetro, incluye el token de autorización en el encabezado de la petición y el método ‘upload’ que sube archivos al servidor utilizando ‘FormData’.

El otro servicio encargado de comunicarse con el servidor es el de “EnviarPredioService”, que se encarga de preparar y enviar los datos relacionados con los predios al servidor. Para enviar los datos, utiliza a su vez el servicio anterior “ServerService”.

Imagen 38. Función 'enviarAlServidor' del servicio 'EnviarPredioService'.

Este servicio, en la función ‘prepararDatosPredio’, prepara los datos del predio especificado por ‘baunitId’ para su envío. Recopila datos de varias tablas locales y los organiza en un archivo ZIP que incluye información en formato JSON y GeoJSON. Por último, en la función ‘enviarAlServidor’, hace uso del servicio ‘ServerService’ para enviar los datos, marcando el predio como “enviado” en la base de datos local después de que el envío se haya realizado con éxito.

Aunque evidentemente todos los servicios creados son igual de importantes porque sin ellos no funcionaría la aplicación, los acabados de explicar son los más relevantes. Los otros servicios existentes son:

- AppGlobalVarsService: se encarga de cambiar la aplicación de modo desarrollador a modo producción y viceversa, cambiando entre ellos las url de la API y ocultando los mensajes de desarrollador.
- GeolocationService: facilita la obtención de las coordenadas actuales del dispositivo utilizando Capacitor y el plugin de Geolocalización.
- MessageService: gestiona la manipulación y visualización de mensajes dentro de la aplicación
- NetStatusService: monitorea el estado de la red y notifica cambios en la conectividad del dispositivo.
- PointService: maneja operaciones relacionadas con puntos en la aplicación, específicamente la eliminación de puntos de linderos (CrPuntoLindero).

7.4. FICHEROS ENVIADOS AL SERVIDOR

El proceso de enviar los ficheros con la información almacenada al servidor implica varios pasos como recopilar los datos del almacenamiento interno (sqlite), preparar los datos y empaquetarlos antes de enviarlos a través de una solicitud HTTP al servidor Django.

Todo comienza con la función ‘prepararDatosPredio’ del servicio ‘EnviarPredioService’, que recopila información específica del predio, como su nombre, departamento, número predial, tipo y otros detalles relevantes. Estos datos se estructuran utilizando un objeto FormData, que facilita la organización y el envío de datos a través de solicitudes HTTP.

Además de los datos del predio, la función también incorpora información sobre los interesados en el predio y las unidades espaciales asociadas. Para cada unidad espacial, se generan geometrías espaciales utilizando coordenadas de puntos de linderos almacenados localmente. Estas geometrías se estructuran como colecciones de elementos GeoJSON

utilizando una clase especializada para representar datos geoespaciales de manera eficiente y estructurada.

Una vez que todos los datos están preparados y estructurados en el objeto FormData, se procede a la creación de un archivo comprimido (ZIP), utilizando la biblioteca JSZip. Este archivo ZIP contiene varios archivos JSON, cada uno representando datos específicos del predio, interesados y geometrías espaciales de unidades. Cada archivo dentro del ZIP se nombra de manera descriptiva para identificar claramente su contenido.

Finalmente, el archivo ZIP generado se adjunta al objeto FormData como un archivo blob. Este objeto FormData, que ahora contiene tanto los datos del predio como el archivo ZIP con toda la información estructurada, se envía al servidor Django a través de una solicitud POST.

```
const formData : FormData  = new FormData();
formData.append( name: 'nombre',  value: datosPredio?.nombre || 'no especificado');
formData.append( name: 'departamento',  value: datosPredio?.departamento || 'no especificado');
formData.append( name: 'provincia',  value: datosPredio?.provincia || 'no especificado');
formData.append( name: 'sector_predio',  value: datosPredio?.sector_predio || 'no especificado');
formData.append( name: 'municipio',  value: datosPredio?.municipio || 'no especificado');
formData.append( name: 'numero_predial',  value: datosPredio?.numero_predial || 'no especificado');
formData.append( name: 'tipo',  value: datosPredio?.tipo || 'no especificado');
formData.append( name: 'complemento',  value: datosPredio?.complemento || 'no especificado');
```

Imagen 39. FormData con los datos del predio.

Además del archivo ZIP descargable, el servidor Django genera automáticamente un correo electrónico de notificación para informar al usuario sobre la subida exitosa del fichero. Este correo electrónico incluye un enlace directo para descargar el fichero ZIP y proporciona detalles clave del predio, extraídos directamente del objeto FormData. Estos detalles son mostrados de manera clara en el correo electrónico para asegurar que el usuario tenga acceso fácil a la información relevante del predio junto con el archivo descargable.

Este enfoque asegura que los datos del predio se envíen de manera estructurada y organizada, manteniendo la integridad y la precisión de la información durante todo el proceso desde la aplicación Angular hasta el servidor Django, así como en la notificación al usuario final.

Este correo que contiene el ZIP descargable con la información del predio la reciben todos los usuarios que desde el servidor Django se han establecido para que los reciban. Se pueden añadir o eliminar tantos usuarios como se desee.

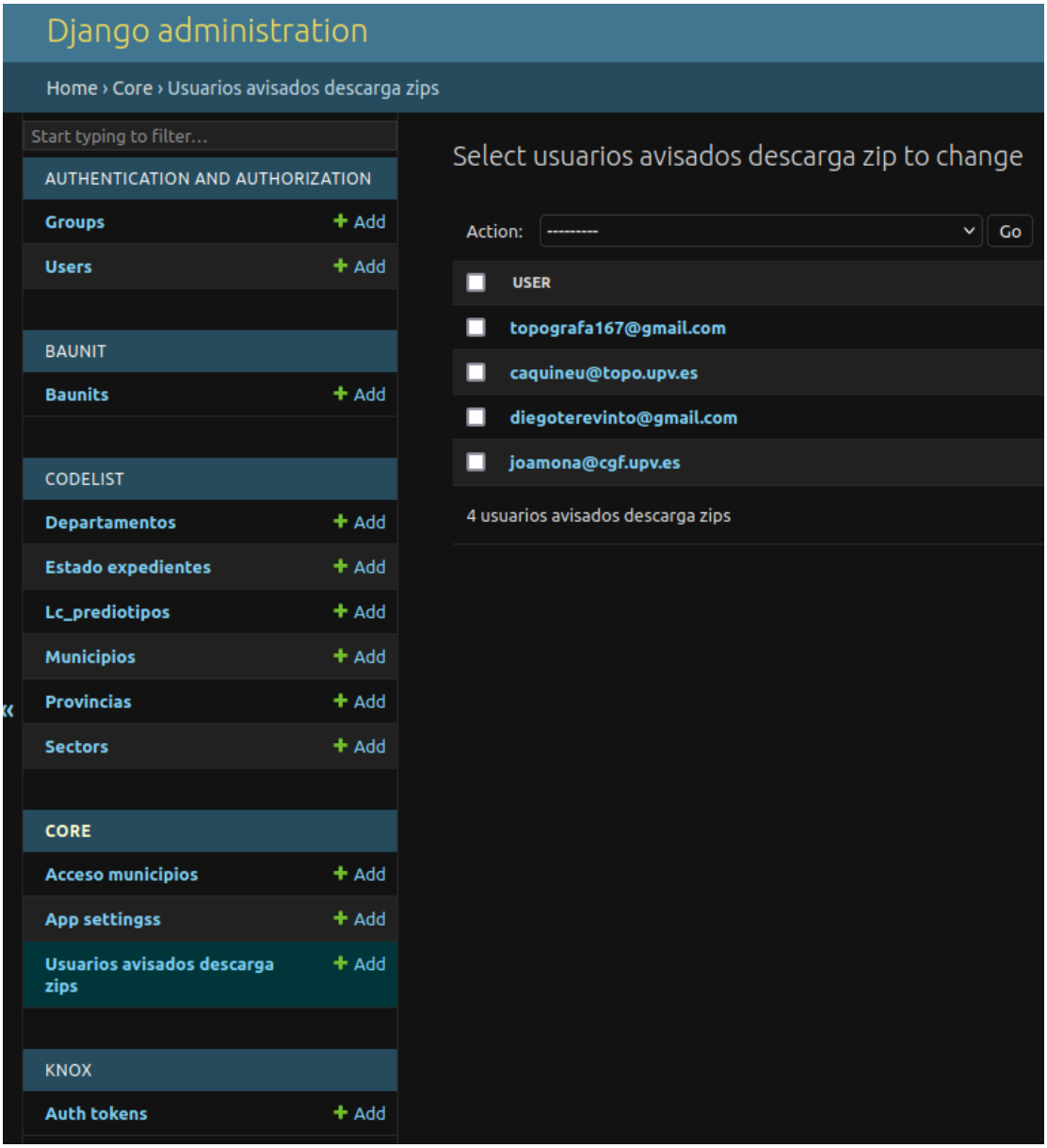


Imagen 40. Pantalla con los usuarios avisados para la descarga del panel de administración de Django.

Además, desde este mismo panel de administración del servidor Django, se puede acceder a todos los ficheros que han sido enviado, pudiendo ver también la fecha de creación, el id del creador, el mismo fichero o una URL de descarga del fichero, que, copiándola en un navegador, iniciará la descarga.

Imagen 41. Últimos ficheros enviados al servidor.

Finalmente, el zip de descarga que recibe el usuario contiene al menos cuatro ficheros: ‘datosPredio.json’, ‘interesados.json’, ‘crPuntosLindero.geojson’, ‘unidadEspacial.geojson’. En el caso en el que el predio tuviera más de una unidad espacial, se generará un GeoJSON por cada unidad espacial.

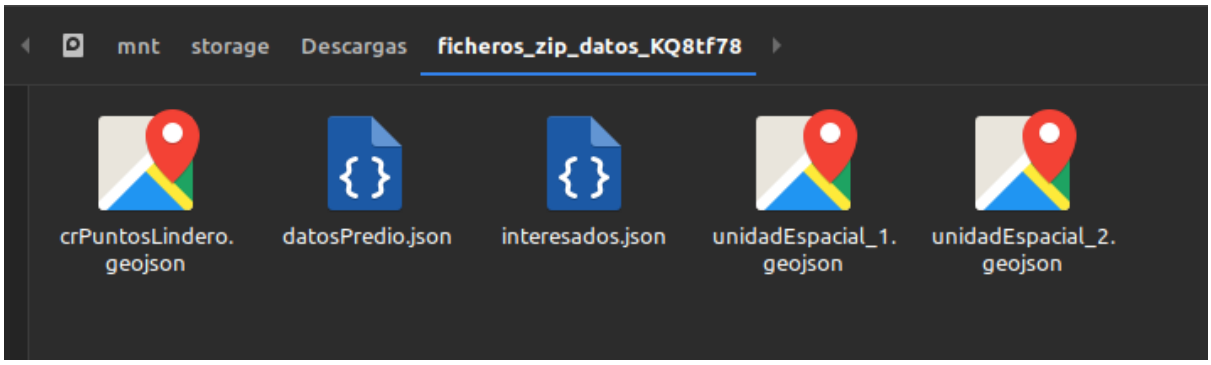


Imagen 42. Carpeta con los ficheros descargados.

7.5. VISUALIZACIÓN DE LOS FICHEROS CON GEOMETRÍA EN QGIS

Para la visualización de los ficheros con geometría en QGIS, es fundamental comprender que los archivos descargados desde la aplicación contienen información geoespacial estructurada en formato GeoJSON.

QGIS es un Sistema de Información Geográfica (SIG) de código abierto ampliamente utilizado para visualizar, analizar y gestionar datos geoespaciales. Cuando se descargan los ficheros desde la aplicación, dos tipos de archivos GeoJSON están disponibles para su visualización en QGIS:

1. ‘crPuntoLindero.geojson’ → Este archivo contiene información detallada sobre los puntos de linderos asociados a cada unidad espacial del predio. Cada punto se representa como un objeto GeoJSON de tipo Point, que incluye coordenadas geográficas (latitud y longitud) y atributos adicionales como precisión horizontal y descripción. El sistema de referencia de coordenadas (EPSG 25830) está definido al inicio del archivo para asegurar la correcta interpretación espacial en QGIS.

```
{
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:EPSG::25830"
    }
  },
  "features": [
    {
      "properties": {
        "id": 1,
        "baunit_id": 1,
        "unidad_espacial_id": 1,
        "tipo": "GPS",
        "lon": -0.5067255,
        "lat": 39.2721189,
        "exactitud_horizontal": 11.475,
        "descripcion": "papelera"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          -0.5067255,
          39.2721189
        ]
      },
      "type": "Feature"
    },
```


Imagen 43. Ejemplo del contenido del fichero 'crPuntoLindero.geojson'.

2. ‘unidadEspacial.geojson” → Este archivo representa las geometrías espaciales de las unidades del predio, como polígonos que delimitan áreas específicas. Cada unidad espacial se define como un objeto GeoJSON de tipo Polygon, con coordenadas que forman el contorno del área y atributos asociados como identificador único (id), identificador del predio (baunit_id), tipo de unidad y precisiones específicas.

```
"features":[
  {
    "properties":{
      "id":1,
      "baunit_id":1,
      "tipo":"gps",
      "Peor_precision":11.475
    },
    "geometry":{
      "type":"Polygon",
      "coordinates":[
        [
          [
            -0.5067255,
            39.2721189
          ],
          [
            -0.5067236,
            39.272113
          ],
          [
            -0.50665,
            39.27204
          ],
          [
            -0.5066466,
            39.2721217
          ]
        ]
      ]
    },
    "type":"Feature"
  }
],
"type":"FeatureCollection"
```


Imagen 44. Ejemplo del contenido del fichero 'unidadEspacial.geojson'.

Por lo tanto, con estos ficheros, el técnico correspondiente que deba trabajar con los datos puede abrirlos directamente en QGIS

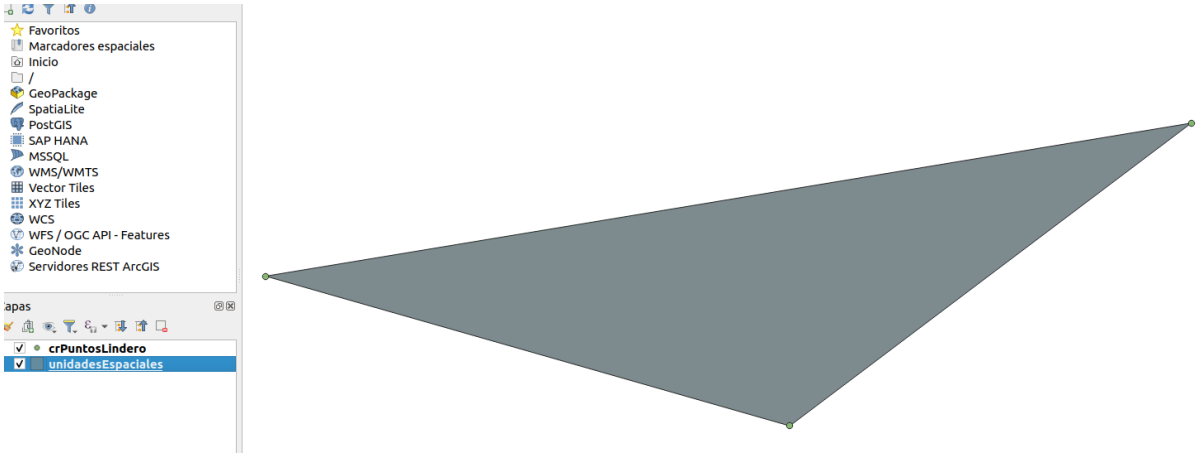
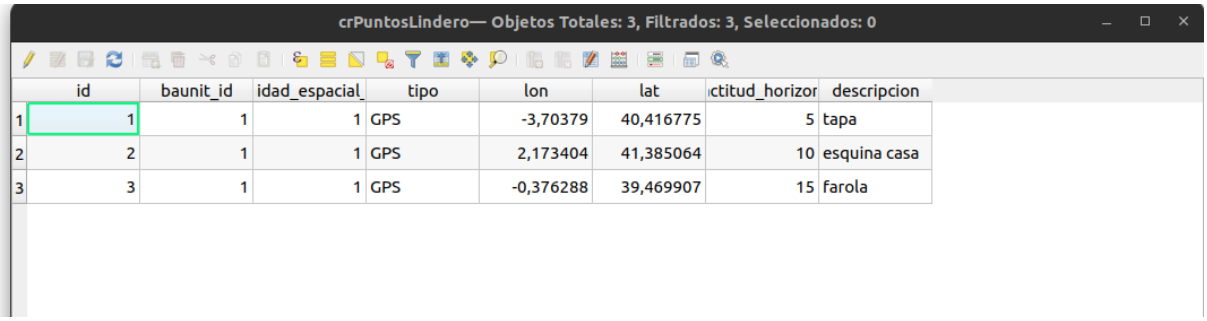


Imagen 45. Visualización en QGIS de geometría obtenida con la aplicación.



	id	baunit_id	idad_espacial_	tipo	lon	lat	ctitud_horizor	descripcion
1	1	1	1	GPS	-3,70379	40,416775	5	tapa
2	2	1	1	GPS	2,173404	41,385064	10	esquina casa
3	3	1	1	GPS	-0,376288	39,469907	15	farola

Imagen 46. Tabla de atributos de la capa 'crPuntosLindero'.

En resumen, los archivos ‘crPuntoLindero.geojson’ y ‘unidadEspacial.geojson’ descargados desde la aplicación contienen datos geoespaciales esenciales para los técnicos y profesionales involucrados en la restitución y regularización de tierras en Colombia. Estos archivos, estructurados en formato GeoJSON y visualizables en QGIS, o cualquier otro SIG, proporcionan las geometrías y las precisiones alcanzadas en los puntos.

8. RESULTADOS

A lo largo del desarrollo y pruebas de la aplicación móvil para la regularización y gestión de tierras en Colombia, se ha observado que la precisión del GPS integrado en los dispositivos móviles puede ser limitada. Generalmente, la precisión de los móviles varía entre 3 y 15 metros, lo cual puede no ser suficiente para aplicaciones que requieren alta exactitud, como la delimitación de predios y la identificación de puntos específicos en un terreno.

No obstante los SIG permiten a los técnicos poner una ortoimagen oficial de base, y fotointerpretar los linderos tomados con la app, con precisiones de +-(2 a 15 m). Los datos tomados con la app son como una declaración del propietario: mi parcela va por aquí. Son luego los técnicos los que completan la información geográfica del predio, digitalizando sobre una cartografía oficial precisa.

Para mejorar la precisión de los datos geoespaciales recolectados existe otra solución viable. Se pueden utilizar dispositivos receptores GNSS de alta precisión que se acoplan a los móviles. Estos dispositivos, como el Garmin GLO 2, Bad Elf GNSS Surveyor, Trimble R1, EOS Arrow Series y Leica Zeno FLX100, permiten obtener precisiones mucho mayores, llegando a ser de submetro o incluso de centímetros, dependiendo del modelo y las condiciones de uso.

Imagen 47. Receptor GNSS Garmin GLO 2.

El uso de estos dispositivos garantiza que los datos geográficos capturados sean lo suficientemente precisos para cumplir con los requisitos de la regularización y gestión de tierras. Esto asegura que la información recopilada sea fiable y útil para los técnicos y especialistas que trabajan en el proyecto, facilitando la tarea de identificación, documentación y análisis de los predios.

Por lo tanto, aunque las precisiones del GPS de los móviles son limitadas, la implementación de receptores GNSS de alta precisión proporciona una solución efectiva, permitiendo que la aplicación cumpla con los estándares necesarios para la regularización de tierras y contribuya de manera significativa al objetivo de la ONG en Colombia.

9. CONCLUSIONES

En este Trabajo Final de Grado, el objetivo principal fue ayudar a Colombia a enfrentar los graves problemas de regularización y gestión de tierras mediante la creación de una aplicación móvil que permita, de manera sencilla, recoger datos personales y geográficos de las parcelas. Tras la finalización del proyecto, se puede afirmar que se ha logrado este objetivo.

El proceso ha sido extenso, pero me ha permitido involucrarme en un proyecto con un impacto real. A lo largo del desarrollo, he aprendido mucho sobre el desarrollo web y móvil, así como sobre los desafíos que enfrentan otros países, problemas que muchos damos por sentados. En particular, me ha sorprendido la magnitud del problema catastral en Colombia, una realidad de la que no era plenamente consciente.

El desarrollo de la aplicación implicó numerosas reuniones y rediseños. La comunicación constante con el tutor Gaspar Mora y el equipo de la ONG en Colombia fue crucial para adaptar y mejorar la aplicación según sus necesidades específicas. El uso de Angular fue especialmente valioso, ya que permitió una estructuración clara y modular del código, facilitando las revisiones y modificaciones necesarias a lo largo del proyecto. Capacitor y SQLite se integraron eficazmente para manejar las funcionalidades nativas y el almacenamiento local de datos, aunque enfrentamos algunos desafíos técnicos que casi nos

hacen desistir. La implementación de SQLite para que funcionara tanto en web como en móvil fue especialmente difícil, pero finalmente lo logramos.

Esta fue la primera vez que me involucraba en un proyecto de esta magnitud. Aunque hubo momentos difíciles, la experiencia fue muy enriquecedora. El proyecto pasó por varios rediseños. Inicialmente, se planteó como un geoportal que eventualmente sería también una app. Posteriormente, se decidió que fuera directamente una aplicación móvil. Sin embargo, al no tener experiencia, estructuré mal el proyecto con Angular. Gracias a mi tutor, que me explicó cómo debía ser la estructura correcta, logramos en la tercera iteración organizarlo adecuadamente, usando SQLite y logrando un proyecto bien estructurado y funcional.

En cuanto a la experiencia de desarrollo, fue en general positiva. Las reuniones semanales proporcionaron una plataforma para el intercambio de ideas y la resolución de problemas. Aunque hubo momentos de dificultad, especialmente al enfrentar problemas técnicos o realizar rediseños importantes, el balance entre esfuerzo y aprendizaje fue muy enriquecedor. Este proyecto no solo mejoró mis habilidades técnicas, sino que también me permitió comprender mejor la importancia de la colaboración y la comunicación efectiva en proyectos de desarrollo.

Aunque se han alcanzado los objetivos mínimos para que la aplicación pueda comenzar a ser utilizada, existen varias mejoras posibles que se seguirán desarrollando paulatinamente. Estas mejoras, planteadas desde el inicio del proyecto, incluyen la capacidad de almacenar documentos como escrituras de terrenos, agregar imágenes para referenciar mejor los puntos tomados, e incorporar servicios WMS, entre otras.

En conclusión, este proyecto ha sido una experiencia integral que combina el aprendizaje técnico con la aplicación práctica en un contexto real y desafiante. La oportunidad de contribuir a una causa tan significativa ha sido profundamente gratificante y ha reforzado mi interés en el desarrollo de soluciones tecnológicas con un impacto social positivo.

10. BIBLIOGRAFÍA

Angular. (2024). Tutorial Angular. https://docs.angular.lat/tutorial

Capacitor Community. (2024). SQLite Web Usage Documentation. https://github.com/capacitor-community/sqlite/blob/master/docs/Web-Usage.md

Concepción del modelo LADM-COL para la gestión catastral. (2024). https://www.igac.gov.co/sites/default/files/2024-05/LADM_3.pdf

Forms in Angular - Learning Angular. (2023). https://www.youtube.com/watch?v=kWbk-dOJaNQ

Fundación Forjando Futuros. (2024). https://www.forjandofuturos.org/

Fundación Forjando Futuros. (2024). Publicación digital sobre la restitución de tierras. https://www.forjandofuturos.org/wp-content/uploads/2024/04/PUBLICACI%C3%93N-DIGITAL-1.pdf

How to Build a Native App from Angular Projects with Capacitor. (2022). https://www.youtube.com/watch?v=V2Wn2JROUEo

Instituto Geográfico Agustín Codazzi. (2024). https://www.igac.gov.co/

Instituto Geográfico Agustín Codazzi. (2023). Beneficios del Catastro Multipropósito. https://www.igac.gov.co/sites/default/files/2023-06/beneficios.pdf

Instituto Geográfico Agustín Codazzi. (2023). Catastro Multipropósito. https://igac.gov.co/catastro-multiproposito/catastro-multiproposito

Ionic enable GPS (native settings) using Capacitor. (2023). https://www.youtube.com/watch?v=NLp1XEMztcQ&t=1659s

Modelo de la Administración del Territorio - Land Administration Domain Model. ICDE. https://www.icde.gov.co/marcos/marco-administracion-del-territorio

OpenLayers. (2024). OpenLayers API Documentation. https://openlayers.org/en/latest/apidoc/

S020 - Introducción al Modelo LADM-COL. (2022). https://www.youtube.com/watch?v=aC9owsNsY-c

TypeScript Tutorial. (2024). https://www.typescripttutorial.net/

Unidad para las Víctimas. (2023). Informe sobre desplazamiento 2023. https://datospaz.unidadvictimas.gov.co/archivos/datosPaz/INFORME%20DESPLAZAMIENTO%202023_VF2.pdf