Forest Inventory Data App

Introduction

This project delivers a tablet-first field application that lets forestry crews navigate to sampling plots and collecting data with ease in modern and intuitive ways. Crews receive a clear list of assigned cruise jobs, open a map, and follow built-in GPS guidance to each plot. As they approach a location, proximity alerts trigger at user-defined distances so users don't miss a point, and the app's built-in validation checks catch entry errors before data is saved.

Designed for remote work, the app automatically downloads the required basemaps and planned cruises for offline use, then synchronizes completed jobs automatically to the ArcGIS server whenever Wi-Fi or cellular service is available. Integration with ArcGIS enables familiar basemaps (satellite, hillshade, roads) and smooth compatibility with existing forestry workflows. The result is a single, intuitive tool for end-to-end timber cruising – plan, navigate, collect, validate, and sync—reducing rework while keeping teams aligned in the field and at the office.

Client Details

Name: Confidential | **Industry:** Forestry | **Location:** Canada

Technologies

Development

- Xamarin.Forms (C# / .NET) for a shared iOS/Android/Windows codebase
- .NET ArcGIS Runtime SDK
- Follows MVVM pattern
- Dependency Injection

Mapping, Location, Tracking

- **Esri ArcGIS Runtime SDK for .NET** (map display, feature layers, offline packs)
- **Basemaps:** Terrain / Topographic / Hillshade / Dark Grey / Light Grey
- **Offline maps:** TPK / TPKX / VTPK Offline load support; mobile geodatabase
- **GPS & background tracking:** Esri location services.
- **Proximity alerts:** Geofence / distance checks to user-selected plots

Data, Storage & Sync

- **Local storage:** SQLite
- **Data validation:** Domain rules on data entry + UI constraints
- **Sync:** Queued offline edits and ArcGIS feature services / REST API when Wi-Fi/Cellular is available
- **Serialization:** JSON (System.Text.Json/Newtonsoft) with conflict resolution

Forest Inventory Data App

Backend & Cloud

- **ArcGIS Online Services** for layers, features, auth & sharing

Auth & Security

- **OAuth 2.0** (ArcGIS IdP) with PKCE
- **Keychain/Keystore** for tokens; HTTPS/TLS everywhere
- **SOC2 Compliant -** Certificate pinning, AES encryption/decryption with randomly generated IV, Jailbreak detection etc.
- **SecureStorage** via Xamarin.Essentials

DevOps

- **CI/CD:** GitHub / Azure
- **Distribution and Code signing via** app center.

Quality & Observability

- **Crash reporting & analytics:** Microsoft App Center
- **Serilog** - Structured logging.

Supported Platform

Android, iOS, UWP

Key Benefits

1. User friendly and compatible with all the major tablet devices (iOS/Android/Win).
2. Seamless data synchronization between the mobile app and AGOL cloud.
3. Cruise offline in remote locations with pre-defined cruises and basemaps.
4. GPS capability built into the mobile app which supports auto tracking and navigation.
5. All-in-one cruise navigation and data entry.
6. Highly configurable for diverse management goals and forest types.
7. Immediate data validation safeguards data integrity.
8. Fully integrated with ArcGIS online.
9. Offline support to navigate, sampling and collecting data in fields.
10. Proximity alerts.
11. Basemaps gets downloaded automatically on app first run, so that it can be used for offline purposes.

Forest Inventory Data App

Use Cases

1. Forest inventory sampling
2. Timber sale cruising
3. Land assessment and appraisal cruising
4. Pre-harvest cruising
    a. Stocking and survival assessments
    b. Natural regeneration
    c. Plantations
    d. Precommercial thins
    e. Commercial thins
    f. Partial harvests
5. Wildlife habitat assessments

Challenges

This application is designed to help field users, primarily cruisers, to easily collect plot-level data and navigate to specific locations using an intuitive interface. Key features include proximity alerts for selected plots, data validation to ensure high-quality inputs, and the ability to access, complete, and sync cruise jobs when connected to Wi-Fi or cellular networks.

Developing this came with several technical and architectural challenges, particularly around ESRI integration and ensuring the app performed reliably in offline, field-based conditions. Below is a summary of the major hurdles we tackled:

1. ArcGIS Authentication

One of the core challenges in the project was implementing secure and reliable OAuth authentication for ArcGIS, using the .NET Runtime ESRI SDK within a mobile environment. While ESRI provides built-in support for OAuth, making it work smoothly in a cross-platform mobile app introduced several complications.

Problems Faced:

The implementation encountered several critical challenges requiring platform-specific solutions. Mobile applications needed custom URI schemes to handle OAuth redirect URIs, unlike web applications, demanding careful platform configuration. Secure token storage required integration with iOS Keychain and Android Keystore while maintaining cross-session accessibility. The ESRI SDK's lack of automatic refresh token handling necessitated precise OAuth flow configuration to prevent repeated user authentication. Account management complexity involved validating existing tokens, handling expired sessions gracefully, and ensuring smooth re-authentication for returning users. Additionally, debugging was hampered by vague error messages when authentication failed

due to misconfigurations, making root-cause analysis time-consuming and challenging for the development team.

How We Solved It

The development team created a custom authentication service extending ESRI's AuthenticationManager to address mobile-specific challenges. The solution implemented a custom OAuthAuthorizeHandler with AuthorizeAsync method for intelligent login UI management, checking existing accounts and attempting silent token refresh before displaying authentication screens. Secure credential storage was achieved using Xamarin's SecureStorage and AccountStore APIs for seamless session restoration. The implementation featured dynamic OAuth flow configuration, selecting appropriate flows based on client secret availability, plus robust error handling with structured try-catch blocks and single sign-in logic to prevent concurrent authentication processes and ensure reliable user experience.

Outcome

The custom authentication layer allowed users to log in once and remain authenticated across app sessions, even offline. It also made the app more secure and compliant with enterprise requirements. This setup was critical for delivering a seamless experience in the field, where connectivity is unreliable and re-authentication needs to be minimal.

2. ESRI Integration

Integrating ESRI's .NET Runtime SDK into a Xamarin mobile application presented several hurdles, particularly around rendering complex spatial data, managing feature services, and ensuring licensing requirements were handled properly. The SDK provided a solid foundation, but several core functionalities required custom implementation to meet the performance and usability needs of field users.

Problems Faced

The implementation encountered three critical challenges requiring specialized ArcGIS SDK solutions. Feature layer loading and rendering presented performance bottlenecks when loading multiple FeatureService layers with varying schemas and data densities, demanding precise control over request timing and lifecycle management. Advanced symbology proved limiting with the SDK's default capabilities, particularly for conditional multi-attribute styling needed to dynamically visualize plots and stands based on status and assignment metadata. Additionally, licensing verification required early validation of ArcGIS Online licenses to enforce proper feature availability and prevent unauthorized access.

How We Solved It

The team addressed these challenges through several targeted solutions. Deferred layer loading used lazy loading strategies for essential layers, reducing initialization times and memory usage. Custom symbology leveraged UniqueValueRenderer and ClassBreaksRenderer for dynamic styling based on plot status and job assignments, improving map readability. License verification at app launch used the ArcGISRuntimeEnvironment.License API to enable or hide features based on license levels for compliance. The solution included linked feature interaction logic that managed selection states across related layers using feature IDs, enabling intuitive plot selection with automatic highlighting.

Outcome

These enhancements made the map view a powerful, interactive tool for field users and not just a static visual aid. The application could render hundreds of layers smoothly, with intuitive symbology and cross-linked data. Licensing was enforced transparently, and the integration with ESRI services remained stable across both online and offline workflows.

3. Basemap Support and Offline Tile Management

Supporting multiple basemap formats - including vector, raster, and tiled basemaps. It was a core requirement for the app, especially to ensure usability in offline or low-connectivity environments. While the ESRI SDK provides tools to manage basemaps, implementing a flexible, performant solution that seamlessly toggled between online and offline sources required several custom strategies.

Problems Faced

Field users required reliable offline access, necessitating pre-bundled tiled basemaps that automatically loaded without network connectivity. Dynamic basemap switching added complexity as users needed to toggle between different styles (imagery, topographic, streets) while preserving operational layers like jobs, plots, and stands, requiring selective replacement without disrupting the map view. Performance issues emerged when downloading large tile packages (TPKs/VTPKs), demanding non-blocking operations to maintain UI responsiveness and coordinated thread management across app sessions for seamless offline preparation and usage.

How We Solved It

To deliver seamless basemap loading with offline support, the team implemented a layered strategy. Priority-based basemap loading checked network connectivity at startup, automatically loading online basemaps or falling back to offline tile packages, ensuring consistent user experience. Modular basemap switching decoupled basemap layers from operational layers,

Forest Inventory Data App

allowing users to switch between styles by replacing only the Map.Basemap property while preserving job, plot, and stand layers intact. Background tile downloading handled downloads in separate threads with intelligent caching to prevent UI blocking and optimize bandwidth. A dedicated basemap management service centralized availability, download progress, and switching logic, improving code maintainability and system reliability.

Outcome

The result was a responsive, user-friendly map experience that worked reliably both online and offline. Field users could switch basemaps as needed, without disrupting their current workflow. Basemaps were downloaded proactively in the background, reducing wait times and ensuring readiness when offline. This implementation significantly enhanced usability and performance in remote areas where connectivity could not be guaranteed.

Auto Tracking And Compass Orientation Across Platforms

Implementing real-time user tracking with dynamic compass orientation was a critical part of enhancing navigation in the field. This feature allowed users to orient themselves toward target plots, navigate confidently in remote areas, and keep their bearings without constantly adjusting the map manually. However, building a consistent and accurate experience across platforms (iOS, Android, and UWP) introduced several challenges related to GPS precision, device sensor inconsistencies, and interaction design.

Problems Faced

The implementation faced cross-platform sensor differences in handling heading and compass data, with varying accuracy across devices. Real-time orientation updates created jitter and lag issues, especially at slower speeds. The app needed intuitive rotation modes (automated and manual) without confusing users, defaulting to north orientation when inactive. Continuous sensor polling required optimization to prevent battery drain and performance issues.

How We Solved It

To address these issues, the team designed a system combining GPS-based heading, user controls, and platform optimizations. GPS-derived bearing replaced magnetometer data, ensuring consistent behavior across iOS, Android, and UWP.

The implementation featured three map orientation modes: north by default, dynamic rotation during navigation (integrated with Follow Me), and manual override via two-finger gestures with

compass reset. A three-state Follow Me tool provided: deactivated (stationary map), location centered (zoomed to GPS), and navigation mode (automatic bearing-based rotation).

Orientation smoothing filtered heading changes to eliminate jitter, set movement thresholds, and throttled updates to preserve battery while maintaining responsiveness.

Outcome

The result was a responsive and user-friendly tracking experience that worked seamlessly across devices and platforms. Field users could trust the map to orient itself in the right direction during navigation, reset easily to true north, and stay centered as they moved. By offloading orientation calculations to GPS and optimizing update cycles, we ensured high performance without draining battery or compromising accuracy.

Offline Plot Navigation in Remote Environments

Enabling offline, GPS-based plot navigation was one of the most technically demanding and critical aspects of the project. Field users often operate in areas without cellular coverage, so the application needed to provide accurate, turn-by-turn-style guidance to plot locations using on-device GPS.

Problems Faced

The implementation faced offline navigation challenges. Traditional tools rely on cloud services, but the app required complete offline functionality with local storage of maps, coordinates, and real-time calculations. Proximity alerts needed two threshold levels (general and precise) requiring GPS polling and notification systems. Navigation flow demanded intuitive step-by-step guidance with visual path display and synchronized orientation updates. Alert management required non-intrusive background functionality respecting user-defined thresholds without interrupting data collection.

How We Solved It

To meet these requirements, the team developed a fully offline navigation system integrated with device GPS and plot management. GPS-based vector navigation drew a direct blue line from user location to plot center using vector geometry, working offline without routing dependency. Dynamic plot location stats displayed real-time navigation metrics including heading, distance, and bearing angle, providing full situational awareness during navigation.

Configurable proximity alerts used a notification queue system managing alerts based on user-defined thresholds: Plot Proximity Alert (e.g., 30m radius) and Plot Center Alert (e.g., 5m threshold) with an "Enter Plot" button to mark arrival. The app automatically recorded entry/exit timestamps in the local database when users entered or exited plots, ensuring traceability for later viewing in

web application inventory reports.

Outcome

The offline navigation tool gave field users a reliable, intuitive way to reach target plots, even in remote forests with zero connectivity. By combining GPS tracking, map orientation, and intelligent alerts, the app enabled confident movement through unfamiliar terrain. The system was accurate, customizable, and lightweight, running smoothly across iOS, Android, and UWP devices.

Offline Plot and Tree-Level Data Collection

The fundamental functionality of this application was to enable full offline data collection at both the plot and tree level. Cruisers often operate in remote locations without internet connectivity, so the app needed to function as a fully self-contained data capture tool, while still honoring validation rules, maintaining data integrity, and offering a smooth user experience.

Problems Faced

The implementation faced significant offline data collection challenges. Offline-first design required users to download jobs with correct sampling templates, collect plot and tree data, and sync later without losing context. Dynamic forms presented complexity as data collection forms were defined in the web application's Configure module, varying significantly between jobs with complex field types, section headers, and conditional validation rules.

Field validation needed real-time enforcement of business rules defined online but without server access, requiring the app to determine required fields, threshold checks, and user guidance during entry. Error and warning management had to surface validation issues clearly without blocking fieldwork, supporting both hard errors (red fields preventing sync) and soft warnings (yellow fields that are optional but recommended).

How We Solved It

To ensure seamless offline data collection, the team implemented several key measures. Users pre-download assigned jobs and configuration JSON files containing form fields, validation rules, and job structures for local storage. A dynamic form rendering engine reads configuration structures and renders native mobile forms for both plot and tree data.

Forest Inventory Data App

The validation framework checks field entries in real-time, marking yellow fields as optional/recommended and red fields as invalid (required fields blank, values below/above thresholds). All data persists locally in a secure Geodatabase, with incremental saves allowing pause/resume without loss across restarts. Upon form exit, a validation report dialog lists unresolved issues, letting users fix or ignore them while remaining flexible enough to sync validated data.

Outcome

The offline data collection system proved to be stable, reliable, and intuitive for field users, even with limited device training. It supported flexible, web application defined configuration – without sacrificing performance or usability. Validation rules enforced high data quality in real time, while still offering field crews the flexibility to continue work when non-critical issues arose. Most importantly, users can collect, review, and sync complex plot and tree data, even days after going offline.

Geo Notes Support for Field-Based Collaboration

The Geo Notes feature in the Prism mobile application was designed to allow field users to create georeferenced notes – points, lines, or polygons – with associated comments that could be viewed by other mobile users and office staff via ArcGIS Online. While the concept was straightforward, the implementation posed several technical and UX challenges, especially around offline use, editing rights, syncing, and maintaining data consistency across devices.

Problems Faced

The implementation faced five critical Geo Notes challenges requiring specialized ArcGIS integration. Feature service dependency demanded correct ArcGIS Online configuration, permissions, and sync settings to prevent data loss during field use. Offline availability required field cruisers to pre-download updated feature services and queue new notes for later synchronization without connectivity. Complex geometry capture needed flexible drawing tools supporting multiple types (point, line, polygon) with proper validation for vertices and shape integrity. Ownership and edit rights enforcement prevented unauthorized editing while maintaining team-wide viewing and commenting. Seamless sync integration ensured Geo Notes synchronized with plot and tree data without conflicts when multiple users updated the same feature service simultaneously.

Forest Inventory Data App

How We Solved It

To deliver reliable Geo Notes functionality, the team built a system integrating with ArcGIS Online while simplifying field operations. Users download and sync the feature service before going offline, ensuring access to current data. Flexible drawing tools support point (one tap), line (two+ vertices), and polygon (three+ vertices with auto-close) geometry types. Users select note types and add descriptive comments for easy categorization. Ownership-based editing ensures only creators can modify or delete their notes, maintaining accountability. UI visibility controls allow users to toggle notes on/off and view details via map popups. All offline edits are queued locally and synchronized with plot and tree data when connectivity returns, minimizing conflicts.

Outcome

The Geo Notes system gave mobile users a lightweight but powerful way to document spatial observations in the field, such as blocked access roads, special conditions, or notes for other cruisers, without relying on email or messaging. It worked smoothly offline, respected user permissions, and required minimal training. It kept the field and office teams aligned by syncing up all note data through the existing ArcGIS Online server.

In-Field Digital Measuring Tool for Distance and Area

Incorporating a digital measuring tool into the application gave users the ability to make quick assessments in the field, such as estimating the size of an unmarked bog or gauging the length of a transect line. This feature was essential for field efficiency and decision making. Making it user-friendly, simple to use and reliable – came with challenges related to UX design, spatial accuracy, and map interaction.

Problems Faced

The implementation faced four critical measurement tool challenges. Non-persistent measurements needed clear separation from synced Geo Notes to prevent confusion. Geometry drawing required responsive, intuitive touch-screen tools for lines and polygons. Unit configuration demanded support for multiple measurement units (meters, feet, acres, hectares) with precise conversions. Clean workflows enabled users to remove points or clear measurements without restarting, requiring robust state management for on-the-fly adjustments during field operations.

Forest Inventory Data App

How We Solved It

To meet these goals, the team designed a lightweight offline measurement system that complements core functionality without sync conflicts. The measure tool, accessed via a map button, offers two modes: Distance (line segments from current location or custom multi-segment lines showing segment and total length) and Area (polygon creation with three+ points displaying calculated area). Users can delete the last point or clear all drawings for rapid field adjustments. Linear and area units are configurable via settings for regional compatibility. All measurements remain local and never sync to ArcGIS Online, clearly separating them from permanent Geo Notes to prevent confusion.

Outcome

The in-field measuring tool became a valuable, quick-assessment feature for cruisers needing to estimate distances or areas without formal data collection. It provided fast visual feedback, simple controls, and real-time results. By separating it cleanly from persistent features like Geo Notes, the implementation remained lightweight, intuitive, and reliable, especially for cruisers operating deep in the field.

AppCenter Analytics Integration for Logs And Crash Reports

To support effective diagnostics and long-term stability of any application, implementation of telemetry systems that could track errors, exceptions, and crashes in real time, are of utmost importance. This needs to be done without compromising performance and user privacy. Logging needed to work across all the supported platforms and provide enough detail for remote troubleshooting, even without immediate access to the user's device.

Problems Faced

The implementation faced four logging challenges. Rich contextual logging needed metadata attachments (variable names, file paths, line numbers) for actionable debugging. Crash reporting required separating soft exceptions from hard crashes needing investigation. Data attachments enabled user-controlled diagnostic packages without exposing sensitive information. Performance and privacy demanded lightweight, asynchronous logging avoiding PII exposure while complying with security guidelines.

Forest Inventory Data App

How We Solved It

To implement a robust, privacy-aware telemetry system, the team built a custom logging and crash reporting service combining Serilog for structured local logging with Microsoft AppCenter Analytics and Crashes for remote diagnostics. Unified error tracking utilities automatically collected exception details and caller context (file name, method, line number) throughout the app for consistent reporting. Users could manually trigger log uploads with optional geodatabase attachments for troubleshooting. Logs and crash reports in AppCenter were filtered and tagged by device type, enabling easy identification of device-specific issues and platform regressions.

Outcome

With this logging and diagnostics system in place, the app team gained real-time visibility into crash trends, high-severity bugs, and user-impacting issues – often before users reported them. The ability to attach contextual metadata and geodatabase files during error reporting made remote debugging significantly faster and more accurate. Importantly, all logging was implemented in a way that respected user privacy, and provided developers with exactly the insights they needed to maintain app quality.

Conclusion

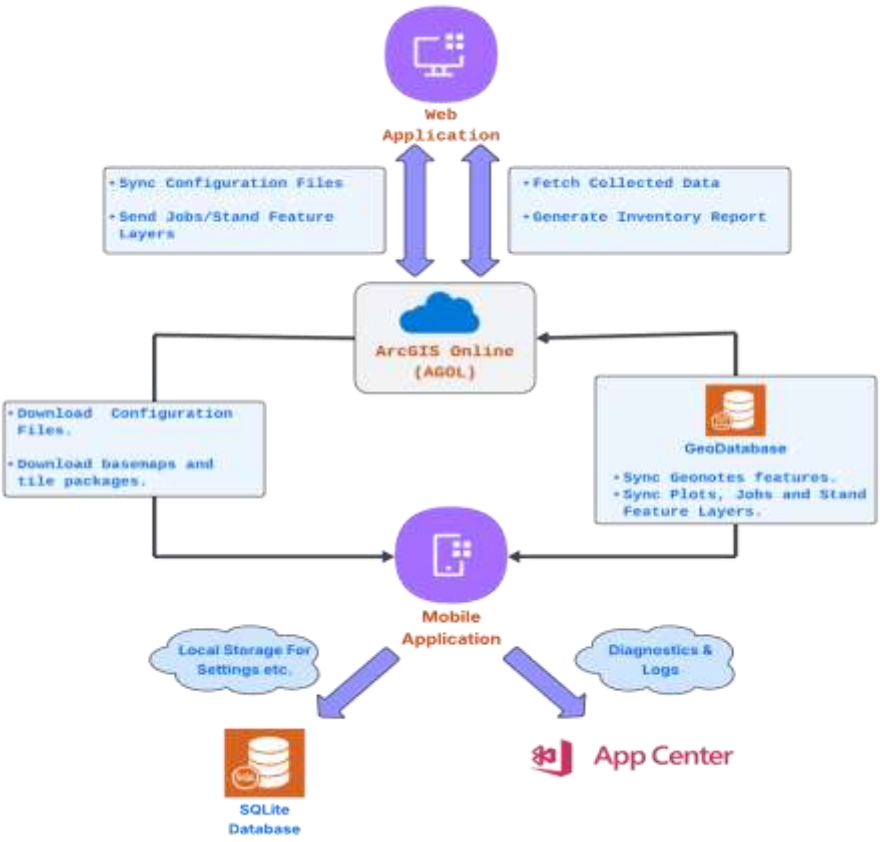
Building this application was a complex, multi-layered engineering effort that required solving a wide range of technical challenges. From integrating secure ArcGIS OAuth authentication and using ESRI SDK efficiently, to supporting offline data collection, GPS-based navigation, and real-time orientation, etc. each feature demanded custom solutions and careful architectural design.

We implemented dynamic symbology, efficient basemap switching with offline fallback, and real-time GPS tracking to guide users in remote environments. Advanced tools like the digital measuring feature and Geo Notes enabled qualitative field observations without disrupting core workflows. Meanwhile, our offline-first form engine, validation system, and telemetry integration ensured both data quality and ongoing maintainability.

By focusing on usability, performance, and platform stability, we delivered a field-ready app that empowers users to collect accurate, high-integrity forestry data, regardless of connectivity. The lessons learned in balancing complexity, performance, and user experience are now core to how we approach mobile development in critical field-based environments.

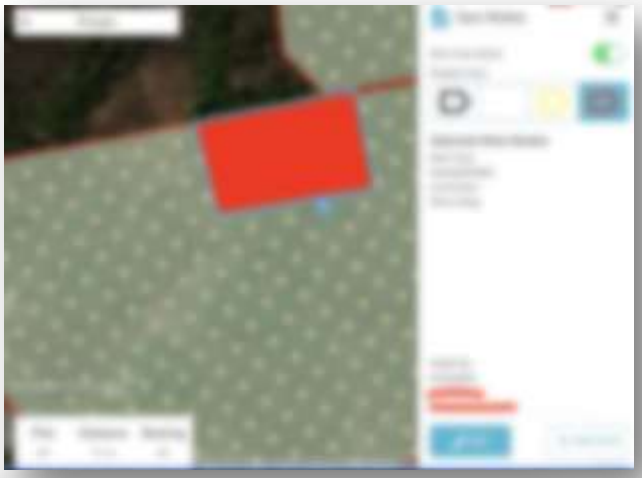
Forest Inventory Data App

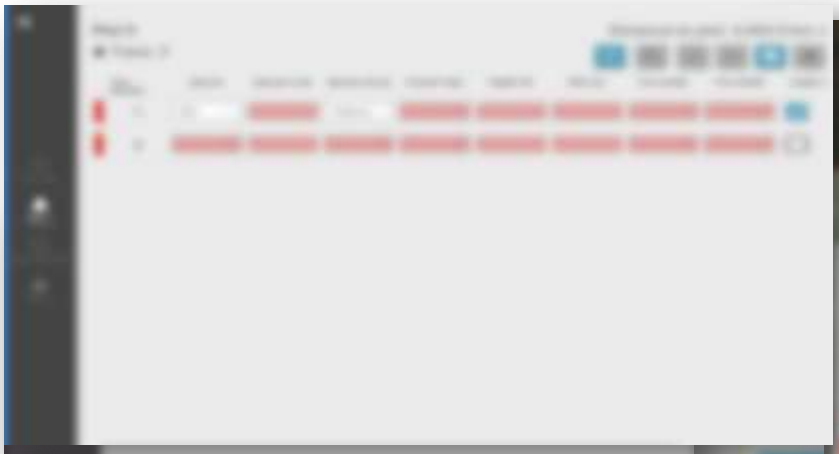
Architectural Design

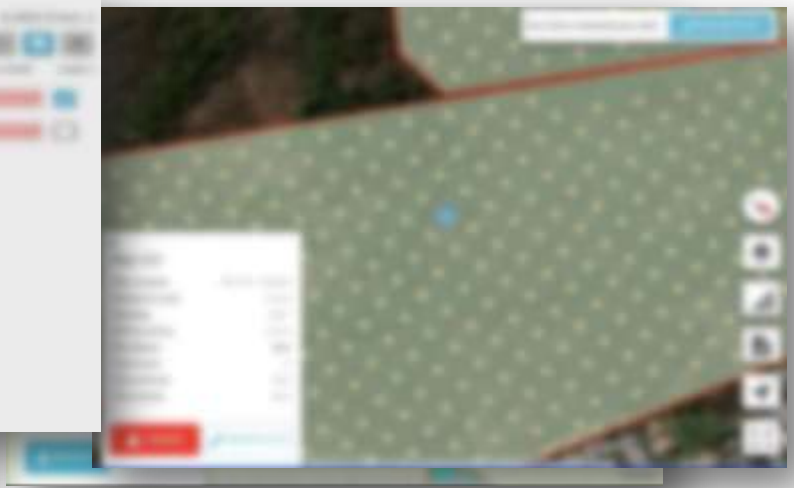


Forest Inventory Data App

Screenshots







Forest Inventory Data App