AgroTrace 360

A Web Platform for Crop Management and Traceability

Capstone Project Report
CS 588 – Computer Science Capstone Experience
Master in Computer Science - Worcester Polytechnic Institute (WPI)

Student: Antonela Tamagnini
Submission Date: 04/11/2025
Revision: Final

1. Project Overview

AgroTrace 360 is a web-based application developed to assist small and medium-sized agricultural producers in Argentina in managing and tracking their crop-related activities with ease and transparency. This platform addresses a significant challenge in the agricultural sector: the lack of accessible, digital tools for traceability, particularly among producers with limited technological resources.

The application enables producers to geolocate their fields, register and manage crops, track treatments and inputs, and monitor harvests. Through this platform, producers can generate internal traceability reports for their own oversight. Additionally, each crop is linked to a QR code that, when scanned, allows consumers to view detailed information about that specific crop—such as its origin, treatments, and harvesting data—promoting food transparency and trust.

The target users of the platform are:

- Agricultural Producers (PyMEs) who need a streamlined way to document their agricultural processes and comply with traceability expectations.
- Consumers, who gain access to key crop details by scanning a QR code, offering transparency into food sourcing.

The Figure 1 illustrates part of the AgroTrace 360 dashboard, including a localized weather widget and a map showing the producer's field location—features that help producers make informed decisions based on real-time context.

By bridging traditional agricultural practices with modern digital tools, AgroTrace 360 offers an intuitive, scalable solution to improve traceability, increase operational efficiency, and build stronger producer-consumer trust in Argentina's agricultural supply chain.

2. Interactive Demo

To better understand the platform’s functionality, a live version of AgroTrace 360 was created, and can be accessed through the following link: https://youtu.be/o6CrVpeK_4k

The demo showcases key features available to agricultural producers and customers, including:

- User registration and authentication
- Field geolocation and management

- Crop registration, editing, and harvesting workflows
- Treatment and input tracking
- Dynamic dashboards with production analytics
- Traceability reports with export options
- QR code generation for each harvested crop
- Client-facing component (QR code reading)

This interactive demo offers a complete walkthrough of the app's core features and the user experience from both producer and consumer perspectives.

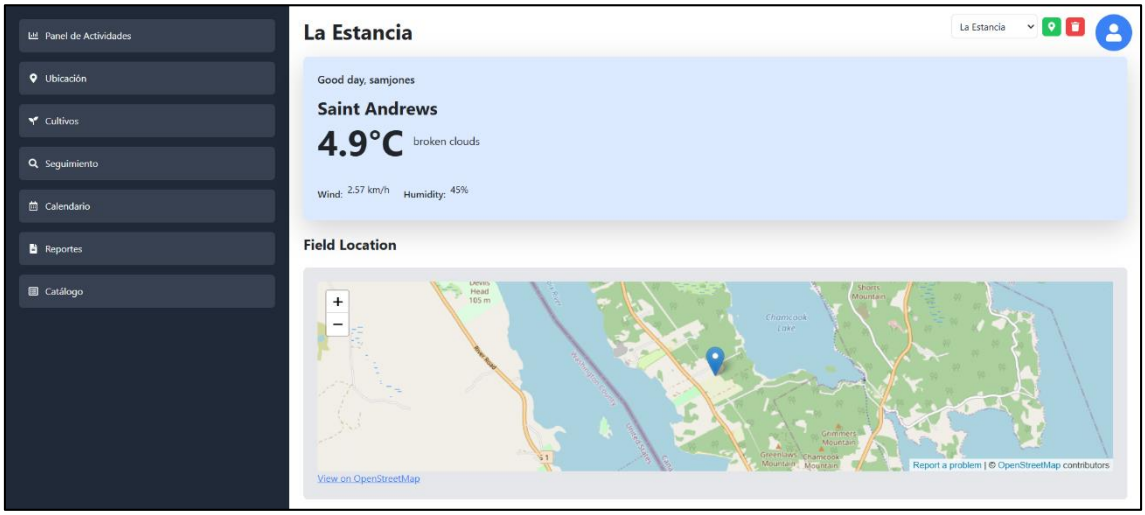

Figure 1. AgroTrace 360 Dashboard Overview

3. Technical Details

AgroTrace 360 is a full-stack, serverless web application designed with scalability, modularity, and usability in mind. It supports agricultural producers in managing field and crop data, while also enabling public access to crop traceability through QR codes.

3.1. Front-End

The front-end is developed using React with TypeScript, providing strong type safety and modular structure for building scalable user interfaces. Key technologies and tools include:

- React Router for client-side navigation across modules like Crops, Tracking, and Reports.
- Tailwind CSS for responsive, utility-first styling.
- Recharts for generating interactive data visualizations.
- Google Maps API for enabling field geolocation and address selection via map interaction or manual input.

The front-end interacts with back-end APIs over HTTPS and dynamically updates the UI based on user roles and data availability.

3.2. Back-End

The application's business logic is implemented in Node.js (JavaScript) and deployed as serverless functions using AWS Lambda. These functions process API requests, handle input validation, execute database operations, and return JSON responses.

- AWS API Gateway is used to expose RESTful endpoints securely to the front-end.
- AWS Lambda hosts the business logic in lightweight, stateless functions.
- Amazon S3 serves both the front-end static site and public crop detail pages (accessed via QR codes).

3.3. Database

The application uses Amazon RDS with MySQL as its relational database. The schema is structured to maintain referential integrity and includes tables for:

- User accounts and authentication details
- Fields and their geolocation data
- Crops and their lifecycle states
- Treatments and input applications
- QR code metadata and access records

SQL queries are executed using a connection pool managed via the Node.js mysql package, with util.promisify used for async/await handling.

3.4. System Interaction Flow

a. The React + TypeScript front-end captures user input and makes API requests via HTTPS.
b. API requests are routed through API Gateway to AWS Lambda functions.
c. Lambda functions handle processing and execute queries against the MySQL database on Amazon RDS.
d. Upon crop harvest, a QR code is generated linking to a public crop detail page hosted on Amazon S3.

This decoupled, serverless architecture ensures ease of deployment, cost-effectiveness, and the ability to scale on demand — while maintaining clean separation between presentation, logic, and data layers.

4. Challenges and Solutions

Throughout the development of AgroTrace 360, several technical and design challenges arose, particularly due to the complexity of integrating geolocation features, handling asynchronous serverless operations, and ensuring data integrity across modules. Below are some of the key challenges and how they were resolved:

4.1. Handling Asynchronous MySQL Queries in AWS Lambda

Challenge: Integrating MySQL with AWS Lambda required careful handling of database connections and asynchronous execution. Since Lambda functions are stateless and can spin up in parallel, improper management of database connections could lead to memory leaks or timeouts.

Solution: To address this, a connection pool was implemented using the mysql Node.js package, and the query method was wrapped with util.promisify to enable async/await syntax. Each function closes connections properly after execution to avoid idle timeouts.

4.2. Geolocation Accuracy and Google Maps Integration

Challenge: Allowing users to accurately register fields using the Google Maps API introduced issues with geolocation precision and address standardization.

Solution: To ensure a seamless way of adding location, the form was enhanced to support both manual address input and click-based selection on the map. This dual-input strategy gave users more flexibility and reduced frustration when default address matching was inaccurate.

4.3. QR Code Linking and Public Crop Detail Pages

Challenge: Once a crop reaches the harvested stage, the system generates a QR code that links to a public crop detail page. Ensuring that this public page remained accessible (but secure and isolated from the main app) was challenging within an S3-hosted environment.

Solution: A lightweight HTML page was dynamically created per harvested crop and uploaded to Amazon S3. This ensured fast, static delivery and separation from the authenticated front-end.

4.4. Designing for Future Scalability Without Overcomplication

Challenge: The system needed to be lightweight for now but capable of scaling in the future — possibly supporting multi-language features, more granular permissions, or analytics modules.

Solution: The application was built with modular React components and serverless functions grouped by feature domain, making it easier to add or modify features without affecting unrelated parts of the codebase. Routing and state management were also structured for scalability.

5. Design and User Experience

The design of AgroTrace 360 focuses on simplicity, clarity, and accessibility, ensuring that users with varying levels of digital literacy can easily navigate and manage their agricultural operations. The user interface was built using Tailwind CSS and designed with a consistent visual hierarchy, intuitive flows, and mobile-responsive layouts.

5.1. UI/UX Principles and Navigation Flow

- The platform uses a sidebar layout with clearly labeled navigation items, making it easy to access key modules such as Fields, Crops, Tracking, Reports, and Catalogue.
- Key actions like crop registration or report generation are centralized within modal forms to reduce page reloads and maintain user context.

5.2. Authentication and Onboarding

Users are welcomed with a clean and modern registration and login interface. Error feedback is displayed when trying to register with an existing username or email, guiding the user to correct input without disruption (Figure 2).

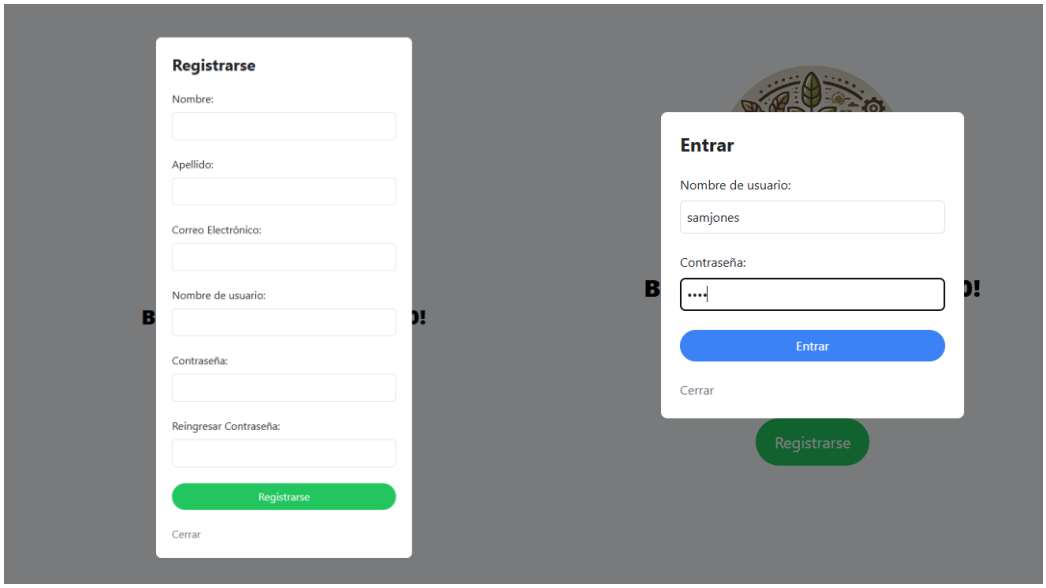

Figure 2. Registration and Login Modals

5.3. Field Registration

Once a producer account is created, the first step is registering at least one field. The field registration screen greets the user by name and prompts for the field's name, size in hectares, and location. A Google Map integration allows users to either input an address or drop a pin directly on the map, accommodating different user preferences and regional data availability (Figure 3).

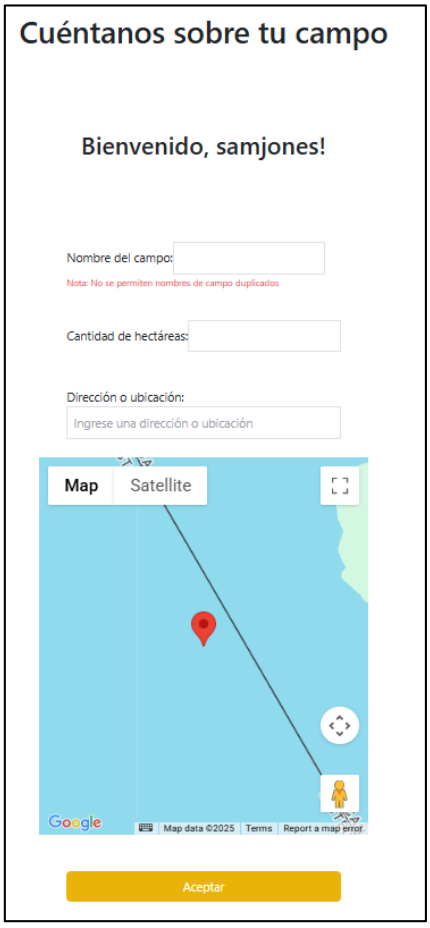

Figure 3. Field Registration with Google Maps Integration

5.4. Data-Driven Dashboard and Visualizations

The Activity Dashboard presents interactive plots that display annual and comparative production statistics by crop type. These visualizations help producers monitor trends and performance over time (Figure 4).

Beneath the graphs, the dashboard includes:

- A summary of currently active crops and their latest statuses.
- Notifications of upcoming actions such as treatments, seeding, or harvesting.

Figure 4. Production Dashboard

5.5. Crop and Field Management

The Crops module allows producers to view, add, and manage detailed crop records (Figure 5). Each crop registration form includes:

- Type, description, and lot number
- Seeding and estimated harvest dates
- Current growth stage
- Seeding and harvest quantities
- Optional notes and attached images

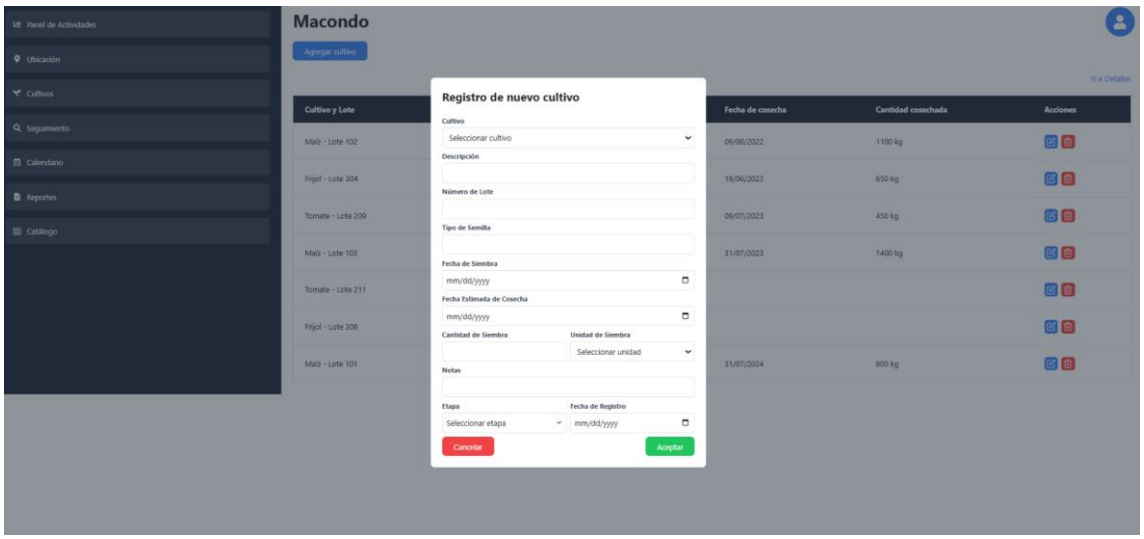

Figure 5. Crop Management Interface

5.6. Reports and Filtering

The Reports module provides powerful filtering and export tools. Users can generate reports by field, crop, seeding and harvest dates, and download CSV files. Tables support sorting to improve accessibility and usability (Figure 6).

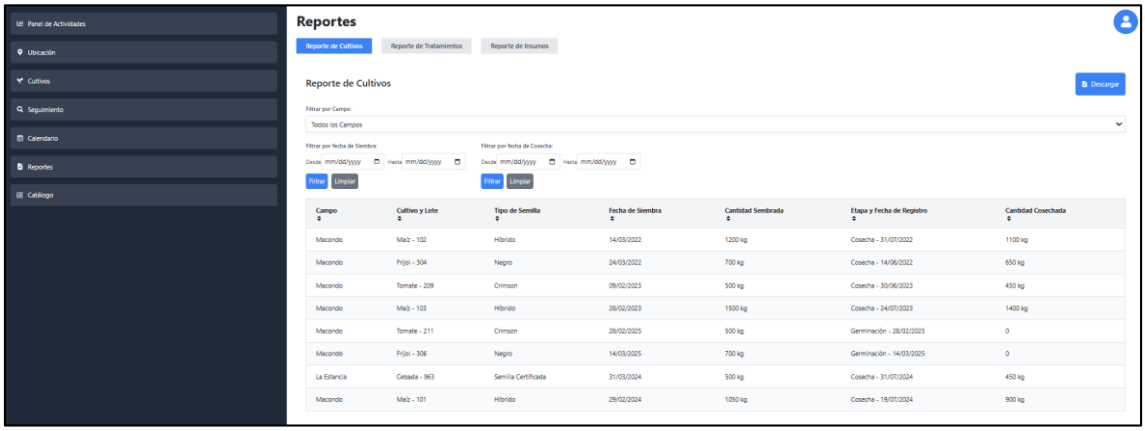

Figure 6: Reports View with Crop Table

5.7. Catalog Access

The Catalogue module displays harvested crops ready for sale (Figure 7). Each crop card includes:

- Crop and field details
- QR code for traceability
- “View Details” links to public crop pages

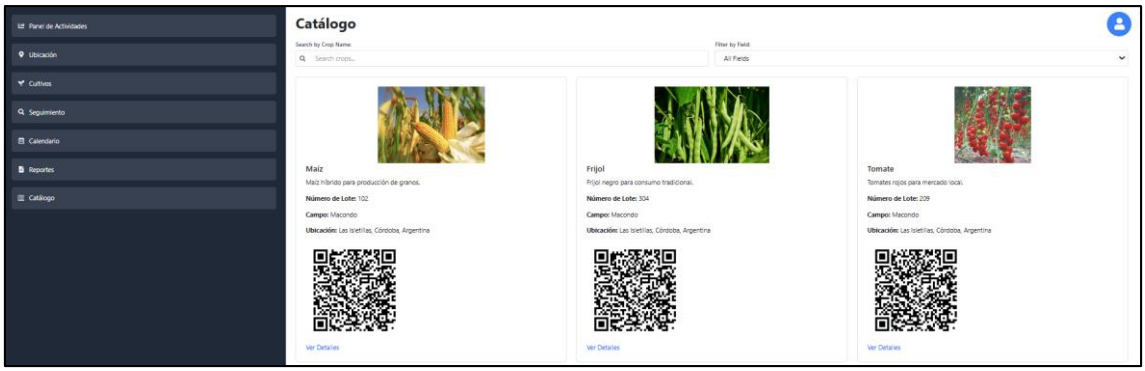

Figure 7. Crop Catalogue

5.8. Consumer Access

When a QR code is scanned by a consumer (Figure 8), a public-facing page hosted on S3 is shown, including:

- Crop details (dates, type, location, quantity)
- Applied treatments
- Available retail locations

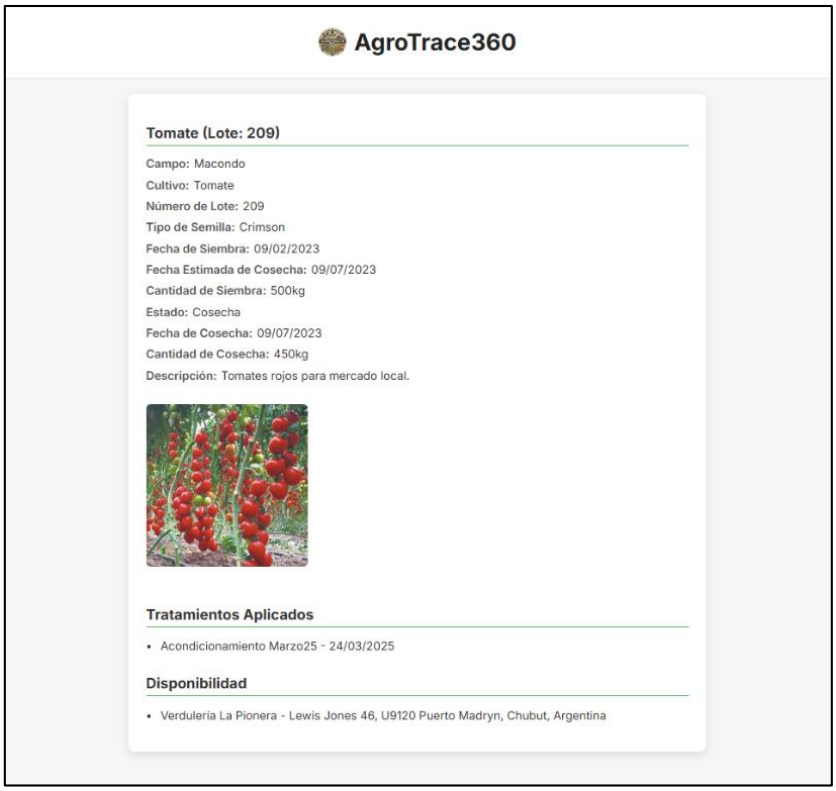

Figure 8. Public Detail Page

6. Testing

To ensure the stability, correctness, and usability of AgroTrace 360, multiple testing strategies were implemented during the development lifecycle.

6.1. Component Testing (Development Stage)

During the early stages of development, Cypress component testing was used to validate individual UI components in isolation. This included form inputs, modals, dropdowns, and navigation elements. Each component was tested for rendering correctness, input validation behavior, and user interactions (e.g., submitting, closing, toggling states).

Component tests helped ensure UI consistency across modules and reduced bugs early in the development cycle.

6.2. End-to-End Testing (Post-Deployment)

After deployment, Cypress end-to-end (E2E) tests were executed to simulate real user behavior throughout the platform. These tests covered complete user flows such as:

- User registration and login
- Field and crop creation
- Treatment and input registration
- Report generation and filtering
- Navigation between modules
- Public access via QR code

The E2E tests helped validate that all critical workflows functioned as expected in the deployed environment, with real API calls and database interactions.

All tests were executed in a staging environment prior to final release, helping to catch integration bugs and regression issues introduced during development iterations.

6.3. User Acceptance Testing

Informal user acceptance testing was also performed by interacting with the system. This helped uncover usability issues, such as unclear button labels or the need for additional validation messages, which were addressed in subsequent design updates.

7. Security Considerations

7.1. Password Hashing and User Authentication

To protect user credentials, bcrypt was used on the front end to hash passwords before sending them to the database. This ensures that plaintext passwords are never transmitted or stored, reducing the risk of credential leaks in the event of a data breach. The hashed passwords are then stored in the MySQL database hosted on Amazon RDS.

During login, the entered password is hashed and compared with the stored hash, allowing for secure authentication without ever exposing raw credentials.

7.2. Session Handling and Access Control

User sessions are managed with a custom authentication mechanism. Once logged in, the user's session is stored securely in local storage and used to control access to protected routes within the app. Access to crop management, field registration, reports, and other producer-facing modules is restricted to authenticated users.

8. Deployment

AgroTrace 360 is deployed using a serverless architecture on Amazon Web Services (AWS).

8.1. Hosting

The frontend, built with React and TypeScript, is hosted as a static site on Amazon S3 and is publicly accessible at: http://agrotrace360.s3-website-us-east-1.amazonaws.com

The backend consists of AWS Lambda functions written in JavaScript, exposed via API Gateway, and connected to a MySQL database hosted on Amazon RDS.

Public crop detail pages (accessed via QR codes) are also hosted on S3 as static HTML files.

8.2. Deployment Process

- The frontend is built with `npm run build` and deployed to S3 via the AWS CLI.
- Backend functions are uploaded to Lambda manually through the AWS Console.
- API routes are configured in API Gateway, and database access is securely managed via IAM and security groups.

- After deployment, Cypress is used for automated end-to-end testing on the live environment.

This deployment approach ensures scalability, reliability, and a clean separation between the frontend, backend, and data layers.

9. Documentation

A complete user manual for AgroTrace 360 is available, providing step-by-step guidance on registration, field and crop management, treatment tracking, report generation, and QR code usage. It includes screenshots and examples to support easy onboarding and is included with the project deliverables.

10. Future Improvements

AgroTrace 360 includes the core functionality needed for crop management and traceability, but several enhancements are planned to improve usability, scalability, and accessibility:

- Responsive design to ensure full usability across mobile and tablet devices.
- Role-based access control for administrator and consumer accounts.
- Offline data entry support for use in low-connectivity rural areas.
- Automated reminders for key dates like treatments, seeding, and harvest.
- Multi-language support, beginning with Spanish and English.

11. Lessons Learned

Developing AgroTrace 360 was a valuable learning experience that reinforced the importance of planning, focus, and iteration. Choosing to follow a Minimum Viable Product (MVP) approach helped me prioritize essential features—such as field and crop registration, tracking, and QR-based traceability—while keeping the project scope manageable. This allowed me to deliver a functional platform within the timeline and build a solid foundation for future improvements.

Early database design was crucial, as having a clear data model streamlined development and reduced the need for structural changes later. I also learned that even in an MVP, investing in intuitive UI design and clear error handling greatly improves usability and feedback.

Testing throughout the process—using Cypress for both component and end-to-end tests—ensured stability and confidence in the core features. Incremental development and early user feedback were key in refining the app and confirming it met real user needs.

Overall, the project taught me the value of starting small, testing often, and keeping users at the center of the process. These lessons will continue to guide my work in future development projects.

12. Conclusion

AgroTrace 360 demonstrates how technology can empower small and medium-sized agricultural producers by simplifying traceability, improving data organization, and building consumer trust. Through its modular design, QR-code integration, and user-friendly interface, the platform offers a practical solution for digitizing agricultural practices in a context where accessibility and ease of use are key.

This project has not only addressed a real-world challenge but also provided a valuable opportunity to apply software development, cloud deployment, and user-centered design skills in a meaningful way. The current version establishes a strong foundation, with a clear roadmap for future enhancements aimed at broader adoption and long-term sustainability.

13. References

The development of AgroTrace 360 was supported by resources and documentation from the following platforms:

- AWS Documentation – for guidance on deploying Lambda functions, API Gateway, and S3 hosting
- Google Maps API – for field geolocation features
- Cypress – for component and end-to-end testing
- Tailwind CSS and Recharts – for building responsive UI components and dashboards
- UI layout and feature ideas were also inspired by best practices observed in platforms like LiteFarm and FarmOS.