

Technical Project Report - Android Module

Agrisense - Smart Agriculture Monitoring System

Subject: Universidade de Aveiro - Computação Móvel

Date: 13/01/2025

Students: 103690: Sebastian Duque González
107378: Daniel Nascimento Pedrinho

Project **Smart Agriculture Monitoring System**

abstract: This project focuses on developing a mobile application for monitoring environmental conditions in agricultural fields. Using simulated sensors, the app provides real-time data of some key parameters through an interactive map interface. Built with a modern architecture (MVVM), the app integrates Firebase Firestore for backend support, ensuring seamless synchronization and scalability. This solution aims to enhance precision agriculture by making environmental monitoring accessible and intuitive.

Report contents:

[1 Application concept](#)

[Overview](#)

[Essential journey map](#)

[2 Implemented solution](#)

[Architecture overview \(technical design\)](#)

[Implemented interactions](#)

[Sub interactions](#)

[Project Limitations](#)

[New features & changes after the project presentation](#)

[3 Conclusions and supporting resources](#)

[Lessons learned](#)

[Work distribution within the team](#)

[Project resources](#)

[Reference materials](#)

1 Application concept

Overview



The Smart Agriculture Monitoring System is a mobile application developed to help users in tracking essential environmental conditions in agri-cultural terrains. This app enables real-time monitoring of various sensor data, including temperature, atmospheric pressure, luminosity and humidity. This app provides instant access to these metrics and it aims to support agronomists, and agricultural enthusiasts in making informed decisions and monitoring plants and harvest.

This monitoring system aims at the growing need for precision in agriculture, where data insights and monitoring can significantly impact productivity and sustainability. With our user-friendly interface and robust backend integration, the application ensures reliable data transmission and visualization. This allows users to respond to environmental changes and unpredictable situations.

Essential journey map:

Phase 1: Login/Authentication

The objective of this phase is to access the application by entering valid credentials for authentication.

- Touchpoint

- The user enters the credentials for authentication, **email** and **password**.
- The user can click on the eye icon to **view** or **hide** the password.

- **Feedback da App:** If the user enters the data **correctly**, a welcome message is displayed and they are redirected to the main tab of the app. If the password is **invalid** (empty or less than 6 characters), the following message is displayed: "Password must be at least 6 characters".

Phase 2: Navegação pela Interface Principal

After successfully logging in, the user is redirected to the **main interface** of the app, which features **tabs** for different functionalities.

- Touchpoint

- **Main tabs available:**
 - **Home:** Displays all simulated sensor data and includes a Logout button.
 - **Map:** Shows an interactive map with pins (markers) representing sensors that when clicked show their current sensor value.
 - **Profile:** Displays the authenticated user's unique ID and profile info.

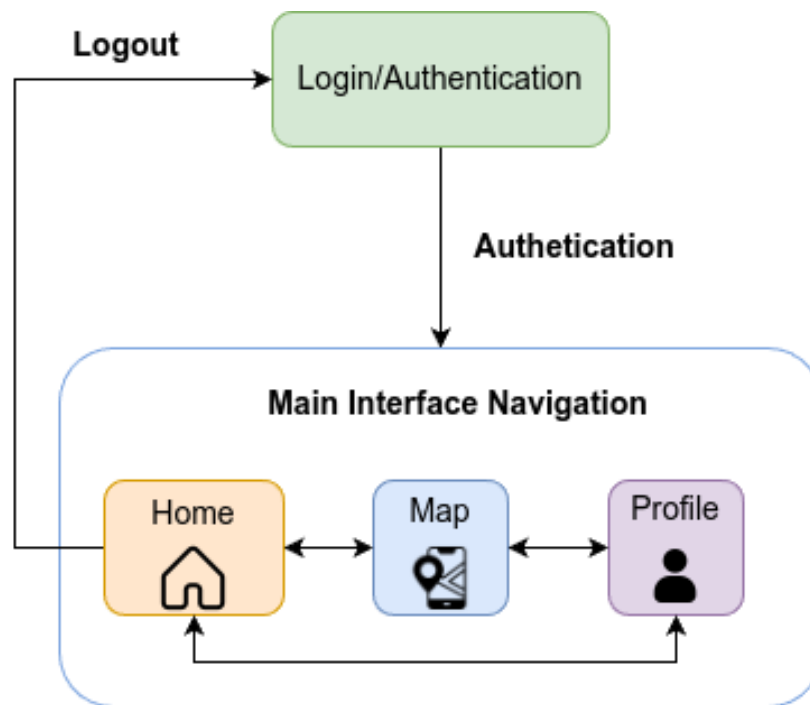


Fig.1 Essential journey map diagram

2 Implemented solution

Architecture overview (technical design)

Model-View-ViewModel (MVVM) Architecture

The app follows the MVVM architecture, separating concerns into **Model**, **View**, and **ViewModel** layers:

- **Model**: Represents the data layer of the app, fetching sensor data from **Firebase** and dynamically simulates sensor readings.
- **View**: Consists of UI components such as fragments (**SensorDetailsBottomSheet**, **MapFragment**, **ProfileFragment**) and layouts. The views observe the ViewModel for real-time updates.
- **ViewModel**: Classes like **MapViewModel** encapsulates all the logic and serve as an intermediary between the view and model, providing data via **LiveData**.

Firestore

- All of the information relevant to the users is stored in a Firestore, maintaining data persistence throughout the entire app. Updates are performed on a “on-demand” basis, meaning data is only stored and retrieved when certain actions that demand the information are performed, like authenticating to load, and creation of new pins for storing

- The Firestore itself contains 2 collections, one with user information and one with pin and sensor information.

-The **user collection** contains documents identified with the UID, and each document contains relevant information for that user

```
{
  "YwfXhFfGa7gzjMWBZtC2sGGPM9n2": {
    "color": "green",
    "dob": "2003-08-15T00:00:00Z"
  },
  "3wtizcG0rBhg6khNODpAJ7zPIW2": {
    "color": "blue",
    "dob": "2000-05-22T00:00:00Z"
  }
}
```

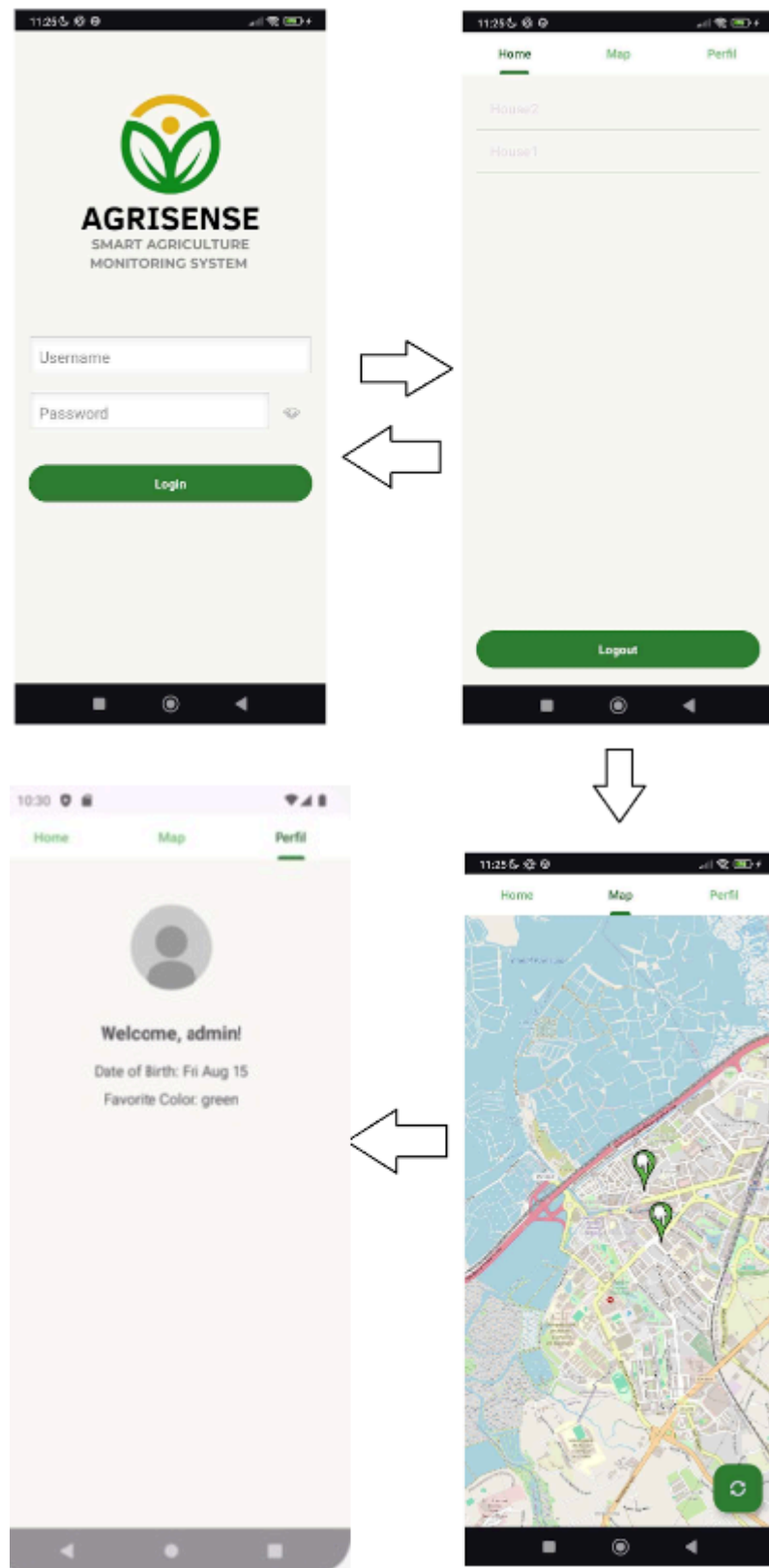
User Information

- The **pin collection** contains all of the pins, with sensors associated with each pin, and its location on the map, longitude and latitude wise

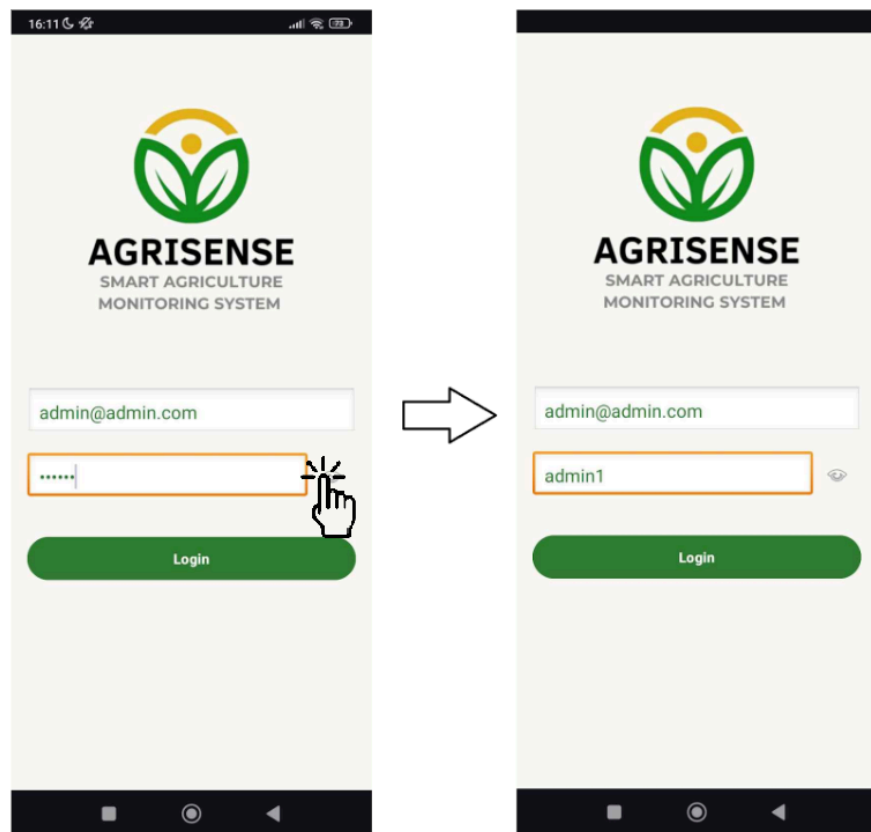
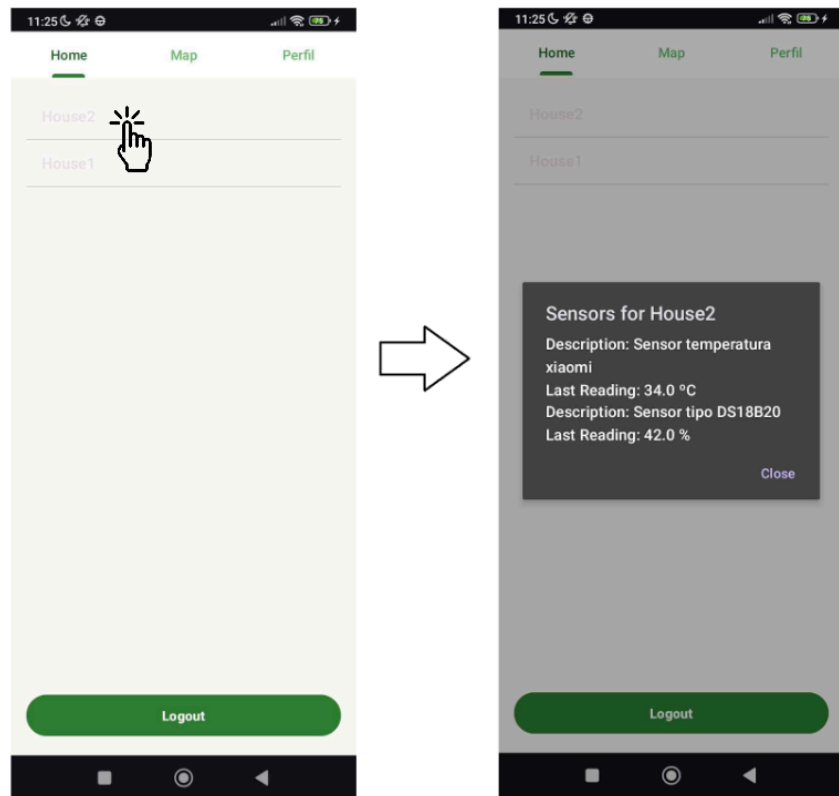
```
{
  "pins": {
    "2PckZeLnuHYyElh3pxob": {
      "latitude": 40.642704117473436,
      "longitude": -8.651454448699951,
      "pinName": "House2",
      "sensors": [
        {
          "description": "Sensor temperatura Xiaomi",
          "typeSensor": "temperature",
          "unit": "°C",
          "value": [22.5, 23.0, 24.1],
          "timestamp": [1672531200000, 1672534800000, 1672538400000]
        },
        {
          "description": "Sensor de Humidade",
          "typeSensor": "humidity",
          "unit": "%",
          "value": [45.0, 47.5, 50.2],
          "timestamp": [1672531200000, 1672534800000, 1672538400000]
        }
      ]
    }
  },
}
```

Pins Information with all the sensor

Implemented interactions



Sub Interactions



Project Limitations

- Currently, no actual sensors are used, although the entire pipeline and workflow developed support them with minimal changes
- No pictures are currently being used in the application, as no suitable solution was found
- Since firebase's base plan does not allow for image storage, we considered hosting the images on external services, like imgur, but our end-goal of using the device camera to register plants would become not viable, since we would need to somehow upload and link that image during runtime.
- Another consideration was local storage. But since these images are user-only, and should be visible on any device, we decided against it

3 Conclusions and supporting resources

Lessons learned


- Implementation of firebase and firestore was challenging at first, specially figuring out how to actually interconnect the database with the app, since it was our first time using firebase
- During development of our project, we realized the power of the ViewModel in Android. Initially, managing dynamic data updates and ensuring a smooth user experience felt challenging. However, the ViewModel simplified this by retaining data during configuration changes and integrating it with LiveData. It improved our code structure by decoupling UI and business logic, making what seemed complex much more manageable and organised.

Work distribution within the team

Taking into consideration the overall development of the project, the contribution of each team member was distributed as follows: Sebastian did 60% of the work, and Daniel contributed with 40%.

Reference materials



1. Libraries:

-  ViewModels & Configuration Changes - Android Basics 2023
- Firebase (Firestore & Auth): <https://firebase.google.com>
- OSMDroid: <https://github.com/osmdroid/osmdroid>

2. Android Components:

- MVVM: <https://developer.android.com/topic/architecture?hl=pt-br>
- ViewPager2 & TabLayout:
<https://developer.android.com/jetpack/androidx/releases/viewpager2>

3. Tutorials:

-  ViewModels & Configuration Changes - Android Basics 2023
-  How to connect Firebase with Android Studio Project [2024]
- <https://github.com/osmdroid/osmdroid/wiki>

Project resources

Resource:	Available from:
Code repository:	https://github.com/ElSebasdg/Android_Project_CM/tree/main
Ready-to-deploy APK:	https://github.com/ElSebasdg/Android_Project_CM/blob/main/app-debug.apk
Demo video:	<optional, but recommended: link to a video demonstration of the app. Consider make a video demo if your project requires specific/complex setup, not easily replicable.>