

TECHNICAL DOCUMENTATION

SMART IRRIGATION MONITORING SYSTEM



**REALIZED BY: GHARBI RANIA
HAMHOUM WISSAL**

GROUP: INDP3-AIM

TABLE OF CONTENTS

I-Introduction.....	2
II-Project Description.....	2
III-Project Description.....	2
IV-Client side.....	4
V-Middleware.....	6
VI-IoT.....	8

I-Introduction

This document is the official technical documentation of the Smart irrigation system application suite. It is divided into 3 parts:

- The client technical documentation: the mobile application.
- The technical documentation of the server: certificate, APIs, virtual machine.
- The technical documentation of the hardware: raspberry pi 4 raspberry pi operating system and DHT22 sensor

II-Project description

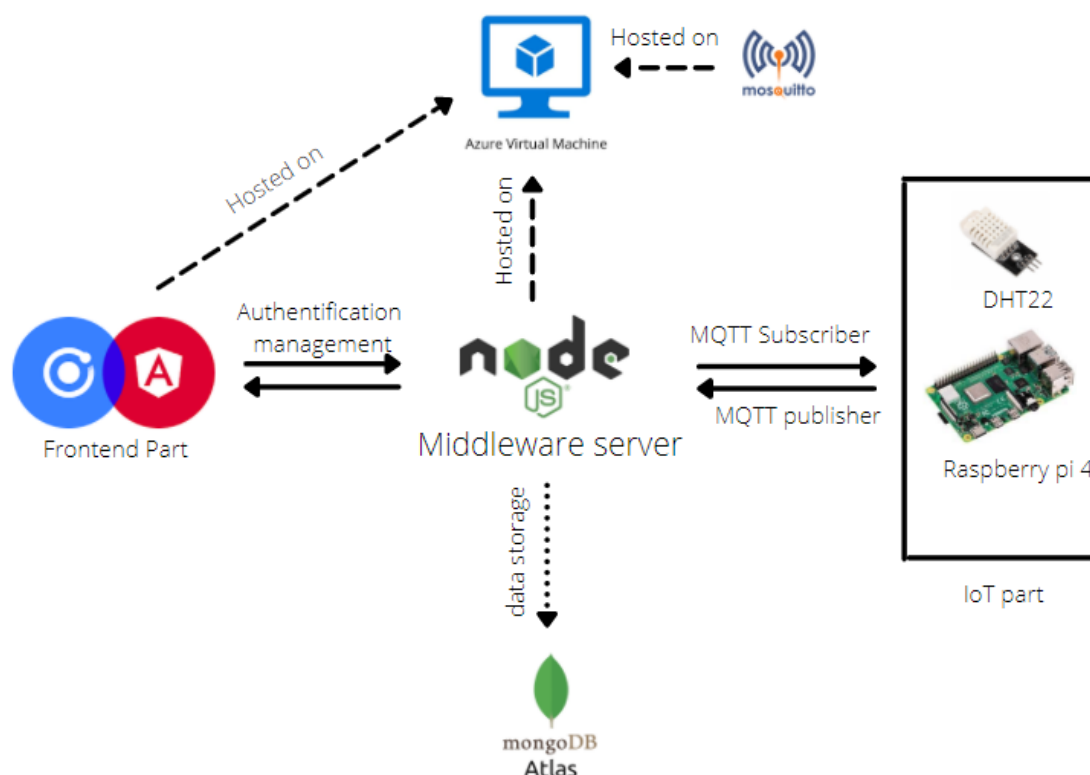
To properly manage the field irrigation we propose a solution based on sensors that measures the soil moisture and temperature, those measurements will help decide if the field needs to be watered.

A mobile application is also a part of the solution.

The application presents the following functionalities:

- Registration / authentication
- Locate the field on the map and precise the different parcels
- Displays information retrieved via an agriculture api on the parcel's characteristics.
- Displays the sensors measurements statistics
- Displays a notification in case the moisture level goes beneath a certain threshold.
- Display the itinerary to a specific parcel

III-Project global architecture



- First we implement the IoT part by connecting sensor to the raspberry pi 4 and we start the Mosquitto and we run the script that is written in C language to connect to the broker and display the temperature and humidity values.
- At the same time we run the subscriber in the Node JS server, hosted in Azur virtual machine, so the values published by the sensor.
- By running the Backend of the application and the Frontend we can use our app.

IV- Client-side

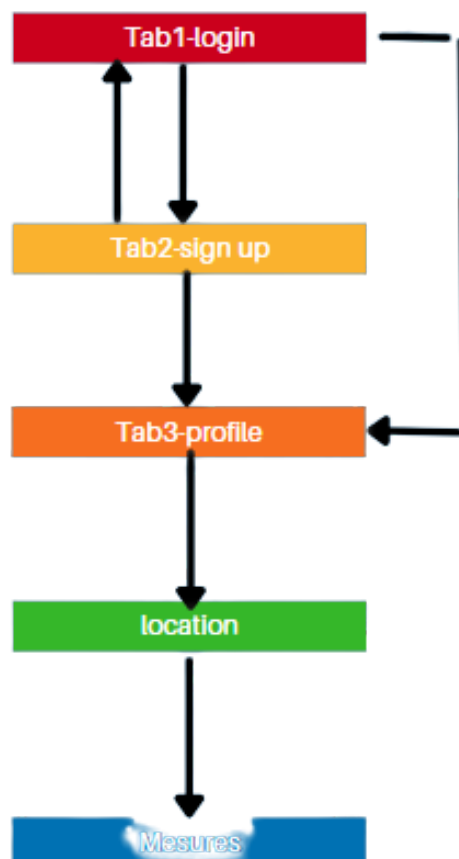
1-Used technologies

- Ionic with angular
- Capacitor

2-Main used libraries

- Leaflet: this library is used to show the map and apply the geolocation and reverse geocoding. We used it mainly because it is free and easy to implement.
- HttpClient: this library is used to build the REST api that connect the frontend to the NodeJS backend.

3-Frontend architecture

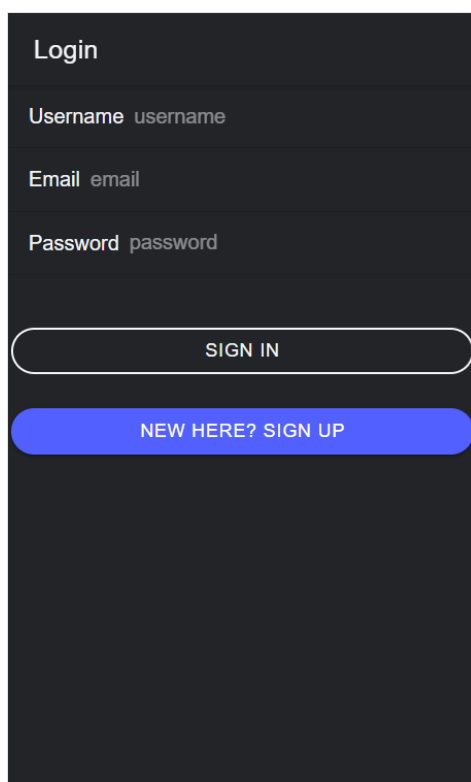


4- Interfaces

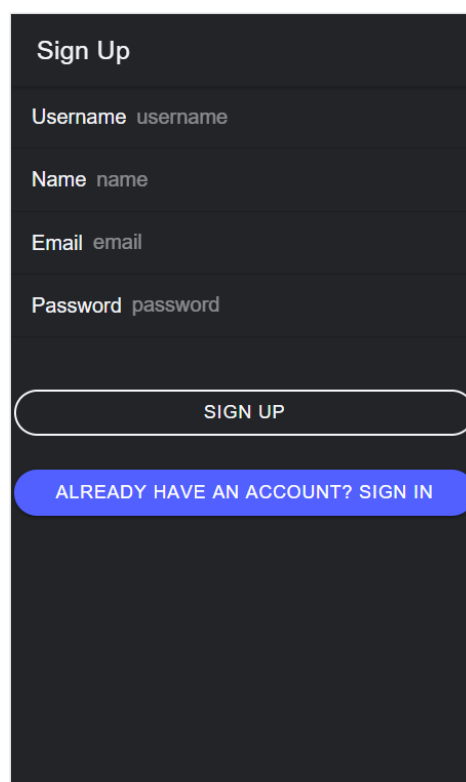
Our application is composed of mainly 4 interfaces:

- **Sign In**: the user must enter his username, email and password if he already has an account. Otherwise, he must click on the button "New here?" to create new account.

- **Sign Up** : the user must enter his username, name, email and password. Otherwise, he must click on the button "Already have an account?" to log in.



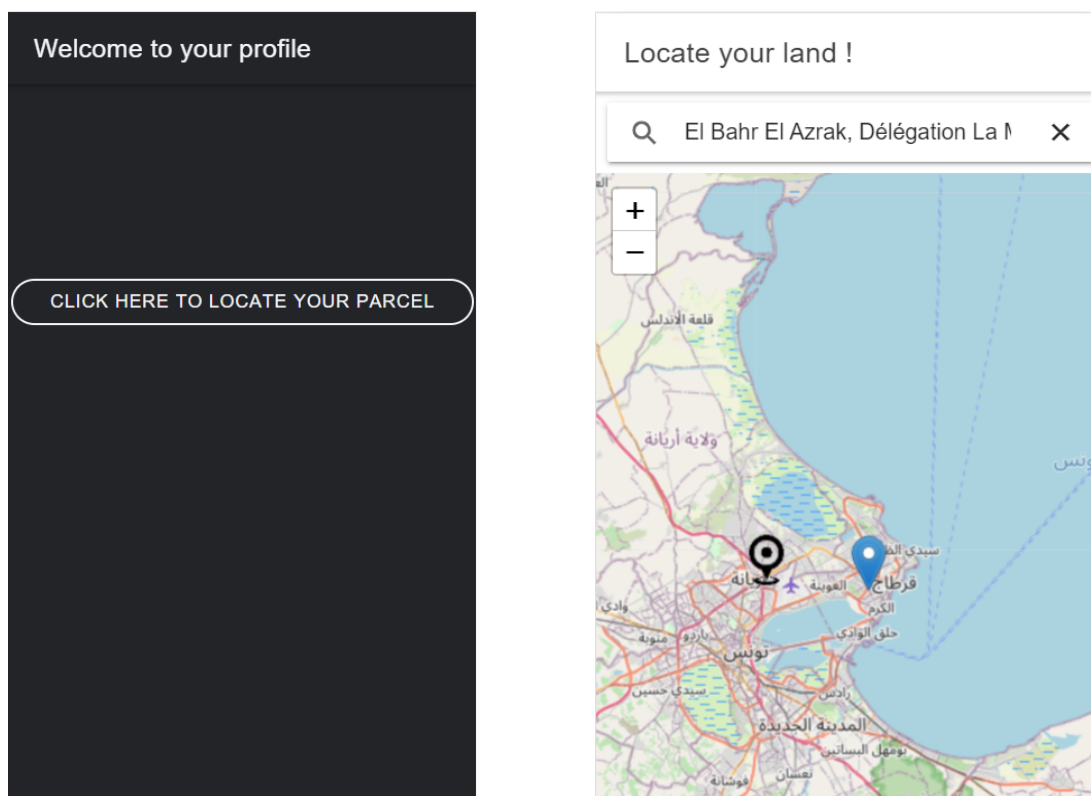
The Login interface is displayed on a dark background. It features a title 'Login' at the top. Below the title are three input fields: 'Username' with placeholder text 'username', 'Email' with placeholder text 'email', and 'Password' with placeholder text 'password'. At the bottom, there are two buttons: a white button with the text 'SIGN IN' and a blue button with the text 'NEW HERE? SIGN UP'.



The Sign Up interface is displayed on a dark background. It features a title 'Sign Up' at the top. Below the title are four input fields: 'Username' with placeholder text 'username', 'Name' with placeholder text 'name', 'Email' with placeholder text 'email', and 'Password' with placeholder text 'password'. At the bottom, there are two buttons: a white button with the text 'SIGN UP' and a blue button with the text 'ALREADY HAVE AN ACCOUNT? SIGN IN'.

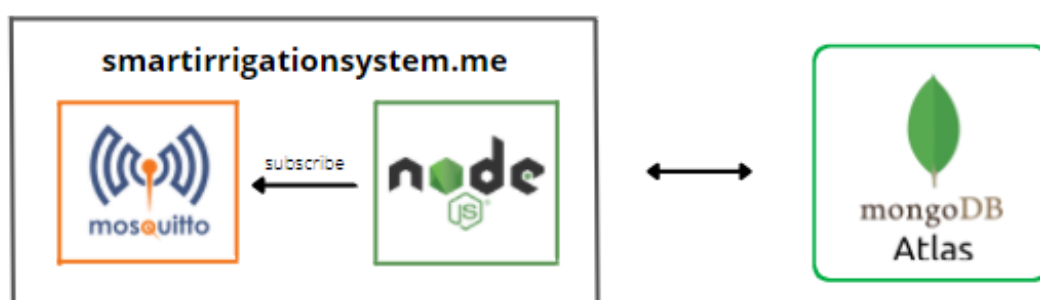
- **Profile**: once the user is signed in, he will find this interface, where he should click on the button "click here to locate your parcel".

-Field search: in this interface the user will find the world map. In the search bar he should type the address of his land and the map will show the itinerary using two markers .



V- Middleware






1- Architecture:



The middleware server is a nodejs server with an Express, it is hosted on an Azure virtual machine along side with the mqtt broker under the domain name "Smartirrigationsystem.me"

2- Domain name : smartirrigationsystem.me

The domain name is obtained from **Namecheap** Which is an ICANN-accredited domain name registrar providing domain name registration and web hosting based in Phoenix, Arizona, US. The domain name is available free of charge for a year thanks to github student package.

<input type="checkbox"/> Type	Host	Value	TTL	
<input type="checkbox"/> A Record	@	20.74.240.100	30 min	
<input type="checkbox"/> A Record	api	20.74.240.100	Automatic	
<input type="checkbox"/> A Record	mqtt	20.74.240.100	Automatic	
<input type="checkbox"/> CNAME Record	www	smartirrigationsystem.me.	30 min	
<input type="checkbox"/> TXT Record	_acme-challenge	DoNOu1VU15uv0IO0gJPENuUW-sn3BL7T0lw5hjs8psU	Automatic	

3- SSL certificates:

In order to secure communications from and to our middleware server, a wildcard certificate was generated by letsencrypt certbot. The certificates were created on the azure virtual machine hosting our server.

4- API description:

The api fonctionnalités:

1. Authorization and authentication
2. Crud operations on users (stored in the table farmer)
3. Crud on fields (each new field is associated to a user by adding the user id as a field in the data base)
4. Crud on parcels: each parcel is associated to a field the same way a field is associated to user
5. Sign in and Sign up methods.

5- Azure virtual Machine:

Specifications:

1. D2S V3 vm hosted in the middle east server
2. 2 Vcpus
3. 8 GiB RAM
4. Static Ip address: 20.74.240.100

VI-IoT

1- Equipments

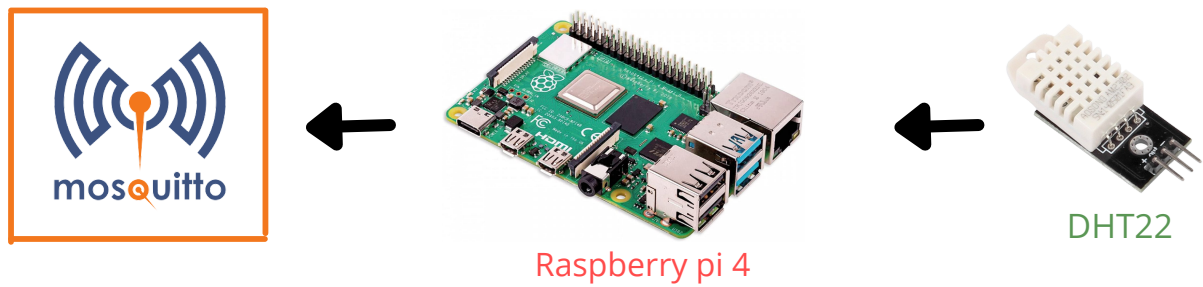
-Raspberry pi 4

-DHT22: temperature and humidity sensor

2- Software

- PuTTY
- VNC
- Raspbian
- Genee programming editor

3- Architecture:



The role of the IoT side is to measure the temperature and the humidity levels of the soil and to communicate these information to the node JS middleware in order to be stored in the database and later displayed on the client mobile application. To do so the raspberry pi 4 card runs a script written in the C programming language, the script formats the pieces of information received from the dht22 sensor and then sends it to the mosquitto broker hosted on an azure virtual machine. The mosquitto, then delivers the data to the nodejs server.