



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Report "*TrustApp*"

A.A. 2024/25

Ivan De Simone - 0001069314

ivan.desimone@studio.unibo.it

Introduzione

L'applicazione TrustApp si propone di fornire un'interfaccia per richiedere e visualizzare dati di umidità del terreno provenienti dalla rete di oracoli ZONIA, interagendo con uno smart contract operante sulla testnet di Ethereum Sepolia.

L'applicazione è sviluppata interamente in Android nativo, utilizzando il linguaggio di programmazione Kotlin.

Le funzionalità principali offerte da TrustApp sono:

- Interfaccia grafica dotata di mappa per visualizzare i dettagli dei dati richiesti
- Form semplice ed intuitivo per effettuare richieste verso ZONIA
- Connessione con il wallet MetaMask, per firmare ed inviare sulla rete le transazioni
- Invio di notifiche per restare aggiornati sull'arrivo di dati dalla portata elevata

UI/UX

L'applicazione presenta un'activity principale, contenente una bottom navigation che permette lo spostamento fra tre differenti schermate, ognuna adibita ad uno specifico sfruttamento delle funzionalità dell'app.

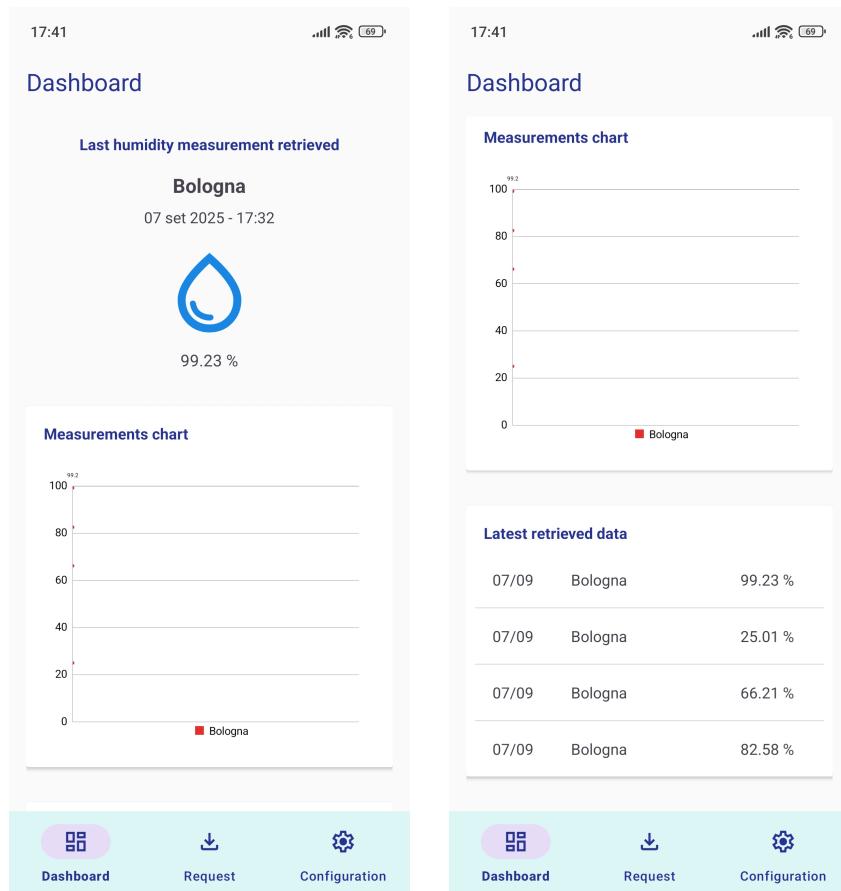
Dashboard

La prima schermata, visualizzabile all'apertura dell'applicazione, è la *Dashboard*. Qui è possibile notare a colpo d'occhio l'ultima misurazione registrata ottenuta dalla rete ZONIA.

L'immagine della goccia d'acqua consente di avere un impatto immediato sul valore misurato, in quanto si ingrandisce proporzionalmente all'umidità rilevata.

Subito sotto è disponibile un grafico a punti che raccoglie interamente le misurazioni salvate nell'app, suddividendole cromaticamente nelle varie località, fornendo informazioni sull'istante della misurazione e sul valore di essa.

Infine l'ultima sezione presenta un elenco delle ultime misurazioni registrate con data, luogo e valore. [1] [2]



Cliccando sulla card che mostra tale elenco, si accede ad una sotto-schermata in cui è possibile visualizzare la lista esaustiva delle misurazioni salvate. Questa lista è filtrabile per nome del luogo di rilevazione, rendendo pratica la ricerca di dati specifici. Le voci elencate mostrano l'istante di rilevazione con una precisione superiore. [3]

Se si vuole però avere informazioni dettagliate su una misurazione, si può cliccare sulla relativa voce nell'elenco, per venire indirizzati alla schermata di dettaglio. Qui sono presentate tutte le informazioni corrispondenti ad una determinata misurazione, quali nome del luogo, istante preciso di rilevazione, valore registrato e mappa, dotata di circoscrizione dell'area di interesse e coordinate del punto di rilevazione. Da questa schermata è inoltre possibile cancellare una misurazione dal database, previa conferma tramite dialog. [4]

17:44

Measurements list

← Measurements list

Search location

Bologna 07 set 2025 - 17:44	34.76 %
Bologna 07 set 2025 - 17:44	78.25 %
Bologna 07 set 2025 - 17:32	99.23 %
Bologna 07 set 2025 - 17:32	25.01 %
Bologna 07 set 2025 - 17:32	66.21 %
Bologna 07 set 2025 - 17:32	82.58 %

17:44

Measurement details

← Measurement details

Bologna

07 set 2025 - 17:44

34.76 %

Delete measurement

Dashboard Request Configuration

Request

La seconda schermata è il motore dell'applicazione: *Request data*. Questa schermata presenta una mappa per selezionare il punto di rilevazione di interesse, seguito da un form a riempimento semi-automatico per l'impostazione dei parametri. Sono presenti due bottoni per l'invio delle richieste: il primo per mandare la richiesta vera e propria verso ZONIA, il secondo per richiedere dati da un sorgente mock a fini di test. Al click sul pulsante di richiesta verso ZONIA viene impostata una connessione con MetaMask, che svolge la funzione di intermediario per le transazioni (approfondito in seguito). [5]

Durante lo svolgimento delle operazioni, un'area di log permette di seguire l'avanzamento del processo di richiesta, al cui termine verrà recapitata una notifica push che segnala il successo dell'operazione, con relativi data ottenuti, oppure il fallimento, con le debite motivazioni. [6]

17:41

Request data

Select your area of interest



Latitude Longitude

44.497880 11.352365

Location name

Bologna

Area's radius [m]

500

Number of measurements desired

1

Request data from ZONIA

Request mock data

Approve to spend sent...
Approve to spend confirmed...
Request sent...
Request not confirmed.

Dashboard Request Configuration

17:43

Transaction not confirmed! - now  The submitRequest transaction has...



Latitude Longitude

44.497880 11.352365

Location name

Bologna

Area's radius [m]

500

Number of measurements desired

1

Request data from ZONIA

Request mock data

Approve to spend sent...
Approve to spend confirmed...
Request sent...
Request not confirmed.

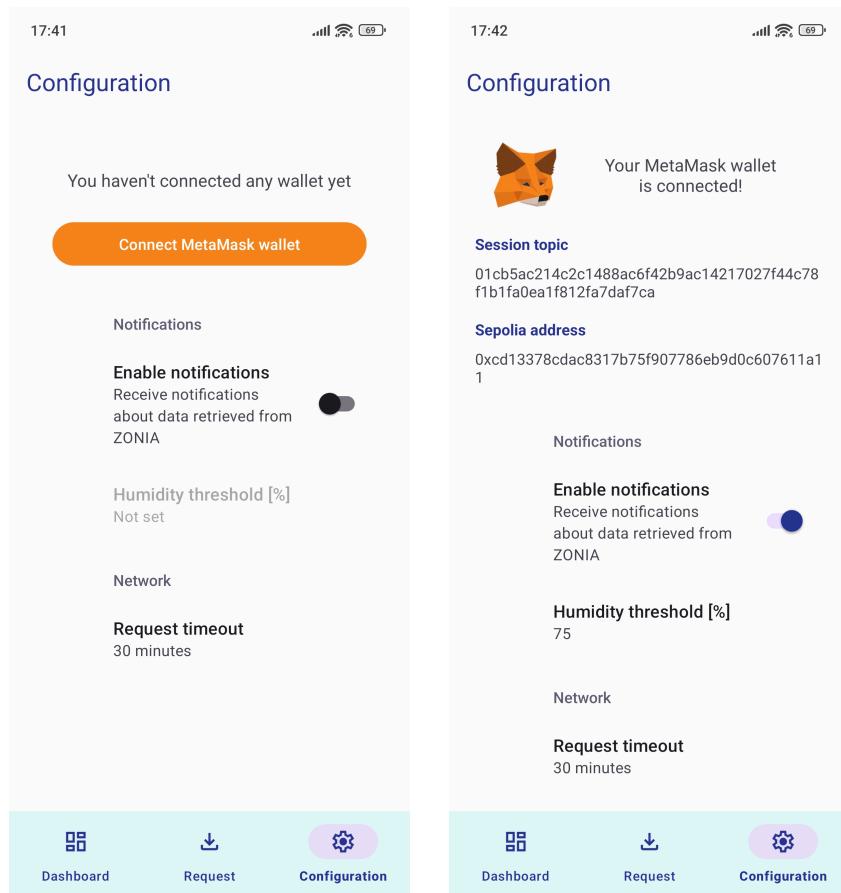
Dashboard Request Configuration

Configuration

La terza e ultima schermata permette di configurare la connessione al wallet MetaMask: *Configuration*. In un primo momento è presente un bottone per effettuare il collegamento, che una volta avvenuto sarà presentato come identificativo della sessione ed address dell'utente sulla rete Sepolia. Questa configurazione è necessaria per poter inviare le richieste verso ZONIA.

A seguire si trovano le impostazioni dell'app, come l'attivazione delle notifiche, previo permesso garantito dall'utente, e l'impostazione di soglia di notifica e timeout per le richieste.

[7] [8]



Flusso di esecuzione

Entrando nel dettaglio di come si svolgono queste operazioni, viene qui presentato il flusso ideale della richiesta verso ZONIA:

1. L'utente connette il proprio wallet di MetaMask, approvando la firma di transazioni sulla rete Ethereum Sepolia
2. L'utente seleziona il punto di interesse sulla mappa nella schermata *Request* ed imposta il raggio dell'area
3. L'utente clicca sul bottone per richiedere i dati a ZONIA
4. Appare una Snackbar che permette di aprire MetaMask per firmare l'approvazione alla spesa degli ZONIA Tokens
5. L'utente è re-direzionato alla TrustApp, dove apparirà un'altra Snackbar sempre per aprire MetaMask, questa volta per firmare la transazione della richiesta vera e propria dei dati
6. L'utente è re-direzionato nuovamente alla TrustApp, dove attenderà la conferma della transazione
7. Al termine dell'esecuzione della funzione sullo smart contract, i dati risultanti vengono ricevuti dalla TrustApp, la quale ne segnala l'arrivo tramite una notifica push

In caso di fallimenti o errori durante lo svolgimento di questo flusso, saranno prontamente inviate delle notifiche di segnalazione, oltre che la scrittura di tali eventi sul log temporaneo.

Dettagli implementativi

L'applicazione è stata sviluppata seguendo il pattern MVVM, suddividendo gli ambiti al fine di ridurre le dipendenze.

Il Model si fonda interamente ed unicamente sull'entità Measurement, che rappresenta una misurazione di umidità del terreno, associata a coordinate, nome del luogo, raggio dell'area, timestamp della rilevazione e valore effettivo. Questa entità viene gestita dal database Room, la cui istanza viene disposta sotto forma di singleton nella main activity.

Abbiamo poi il Repository, che si occupa di fare da intermediario tra il database, le fonti di dati remote ed i ViewModel. Tale repository mantiene due riferimenti a due classi distinte: una per effettuare le richieste alla sorgente di dati mock, ed una per effettuare le chiamate tramite la blockchain.

Sono presenti all'interno dell'app due ViewModel complementari, uno maggiormente legato ai dati da visualizzare a schermo, ed uno più orientato all'interazione con MetaMask ed in generale con la testnet Sepolia.

L'interazione con la blockchain Ethereum Sepolia si basa sull'utilizzo delle librerie WalletConnect e Web3j. La prima è fondamentale per poter stabilire la connessione con il wallet MetaMask, nonché per codificare le transazioni da firmare e successivamente decifrarne il risultato. La seconda si rende necessaria per connettersi ad un nodo RPC, in questo caso messo a disposizione da Alchemy, con lo scopo di interrogare la blockchain tramite funzioni che non ne cambiano lo stato (e quindi non richiedono il pagamento di fee).

Per poter inviare notifiche da frammenti di codice non dotati di contesto (repository,.viewmodel), si è ricorso all'utilizzo di un contratto tramite un'interfaccia, per dichiarare le funzioni su cui fare affidamento, che sarebbero poi state implementate concretamente da una classe separata, a cui il contesto viene passato tramite il costruttore.

Nella schermata per effettuare le richieste, oltre alla mappa viene fatto uso di Geocoding, traducendo le coordinate del punto selezionato dall'utente in un nome di luogo sufficientemente preciso da identificare la posizione.

Bug noti

A seguito della prima installazione dell'applicazione sul dispositivo, è stato rilevato in alcune occasioni un crash, molto probabilmente dovuto ad una race condition in fase di inizializzazione del client WalletConnect. Questo problema sembra non verificarsi più alle successive esecuzioni dell'app.

Nonostante la formattazione della query al Gate di ZONIA come da esempi forniti, la richiesta per i dati non conclude mai a buon fine, arrivato allo stato di timeout oppure di rifiuto dell'esecuzione, rendendo impossibile l'implementazione dell'inserimento nel db dei dati ricevuti da tale fonte.

È stato rilevato in rari casi un malfunzionamento della libreria per graficare i punti nella Dashboard, che porta ad un crash dell'applicazione. A questo problema non è stata trovata una soluzione, rendendo necessaria la disinstallazione e re-installazione dell'app per consentirne il funzionamento.

Sviluppi futuri

L'applicazione potrebbe subire numerosi aggiornamenti per migliorare l'esperienza utenti. Andando per gradi, si potrebbe consentire l'ordinamento della lista (oltre al filtraggio) per una migliore analisi dei dati. Sempre riguardo alla presentazione dei dati, si potrebbero aggiungere grafici aggiuntivi alla dashboard.

Una modifica corposa che si potrebbe effettuare, essendo che lo smart contract è attivo anche sulle blockchain BSC e Polygon, sarebbe consentire all'utente di selezionare la blockchain da cui preferisce ricevere i dati, magari fornendo delle stime dei costi di transazione in tempo reale, che possono fluttuare in base al congestionamento della rete.