Applicazione multi piattaforma per la condivisione di dati biometrici sfruttando l'infrastruttura IPFS

Nicandro Potena, Francesco Filippi

Febbraio 2023

1 Introduzione

Il nostro progetto mira a sviluppare la proposta numero 14 descritta sul sito del corso. Tale proposta richiede la realizzazione di un sistema che permetta di immagazzinare dati criptati in uno storage decentralizzato. I dati che raccogliamo sono dati biometrici che vengono salvati in una infrastruttura IPFS, per fare ciò abbiamo deciso di utilizzare il servizio gratuito di Pinata e le Api che fornisce. Tramite un'applicazione sviluppata in Ionic, che permette di essere eseguita su tutti i tipi di device (IOS, Android, Windows), offriamo all'utente un'interfaccia adatta per gestire sia il caricamento sia il recupero dei file andando a comunicare con IPFS.

2 Casi d'uso

Abbiamo sviluppato l'applicazione pensando a due possibili casi d'uso:

1. All'interno di un nucleo familiare si vuole monitorare lo stato di salute/dati biometrici di un membro della famiglia. Con la nostra applicazione è possibile farlo andando ad abilitare l'utente da monitorare come utente che pubblica i dati, e i familiari come utenti che visualizzeranno i dati.

2. Un medico vuole monitorare i dati di diversi pazienti, ogni paziente andrà a pubblicare i dati con un ID diverso, in modo da renderli riconoscibili, mentre il medico potrà andare a visualizzarli.

3 Installazione

Per effettuare l'installazione dell'applicazione è necessario seguire i seguenti step:

- Scaricare ed installare NodeJs.

- Scaricare ed installare Ionic tramite CLI, eseguendo 'npm install -g @ionic/cli'
- Scaricare ed installare Git
- Scaricare ed installare Android Studio
- Clonare il progetto da Git tramite CLI usando 'git clone https://github.com/NicandroP/Blockchain-project.git'
- Posizionarsi nella cartella del progetto prima di eseguire i comandi successivi
- Installare le dependencies di npm e di axios, utilizzando i seguenti comandi su CLI: 'npm install' e 'npm install axios'
- Per lanciare l'applicazione desktop, eseguire tramite CLI, il comando 'ionic serve'
- Per installarlo su Android, eseguire i seguenti comandi: 'ionic cap sync', 'ionic cap open android', 'ionic cap run android'

4 Tecnologie utilizzate

Le tecnologie utilizzate sono:

1. IPFS (InterPlanetary File System) [3] è un protocollo di comunicazione e una rete peer-to-peer per l'archiviazione e la condivisione di dati in un file system distribuito. IPFS utilizza l'indirizzamento basato sul contenuto in modo da identificare in modo univoco ogni file sulla rete.
2. Ionic [2], framework open-source che permette la creazione di applicazioni multipiattaforma sfruttando la logica delle pagine web e quindi il linguaggio di meta-markup HTML, il linguaggio di programmazione ad oggetti javascript o typescript, in base alla scelta del developer, e CSS, inoltre permette l'integrazione dei framework più popolari, in questo caso è stato scelto Angular.
3. Angular [1] è un framework open-source, scritto in typescript, di sviluppo per applicazioni web supportato da Ionic e sviluppato principalmente da Google. Questo framework fornisce la possibilità di dividere in moduli le applicazioni più complesse, questo garantisce di poter avere una riutilizzabilità del codice e inoltre permette tramite lazy-loading di velocizzare i vari caricamenti durante la navigazione all'interno dell'applicazione andando a caricare soltanto i moduli necessari in quel momento, e solo quando necessario, gli altri moduli.
4. Pinata [4] è un servizio di cloud storage che garantisce la resilienza dei dati e ti permette di usufruire di tutti i vantaggi di uno nodo IPFS tramite l'utilizzo delle API che fornisce.

5 Funzionalità

5.1 Login

Al lancio dell'applicazione verrà chiesto all'utente di selezionare una modalità tra writer e reader. Il writer sarà l'utente che avrà la possibilità di pubblicare i dati, l'applicazione chiederà se registrarsi come nuovo nodo oppure se abbiamo un device già registrato come writer. Se viene selezionata l'opzione per creare un nuovo nodo, verrà generata una nuova chiave di Pinata, tramite l'apposita API, e una password generata casualmente, per il criptaggio dei dati, che verranno visualizzate dall'utente, come mostrato in figura 5.1 e serviranno per il recupero dei dati da parte dei reader. Nel caso in cui venga scelta l'altra opzione, all'utente verrà richiesto di inserire public key, private key, e password dell'altro device già registrato.
Il reader sarà colui che potrà solo visualizzare i dati, una volta selezionata questa modalità, verrà chiesto all'utente di inserire il valore di public key e password dell'utente di cui vogliamo visualizzare i dati. Tutti i reader avranno la stessa chiave pinata, ma andranno a recuperare i dati in base alla public key inserita durante la fase iniziale, i writer invece avranno una chiave pinata differente, questo permetterà il riconoscimento all'interno del nodo IPFS.

5.2 Caricamento dei dati

Cliccando sulla tab 'Uploading', è possibile tramite il form, mostrato in figura 2, caricare dati biometrici relativi alla persona. In questo form i valori richiesti sono: height, weigth, age, min pressure, max pressure, blood glucose. Andando a cliccare sul bottone 'upload file' viene generato un file con formato Json, viene criptato con l'algoritmo 'AES', per permettere solo agli utenti che conoscono la password di vederne il contenuto. Successivamente viene fatta la chiamata alla API Pinata, per caricarlo su IPFS. All'interno dei metadati del file caricato viene inserita la public key del writer, per poter essere identificato.
Nel caso in cui l'utente sia un reader, questa pagina non sarà visualizzata.

5.3 Recupero e visualizzazione dei dati

Cliccando sulla tab 'Your data' l'utente potrà visualizzare tramite una lista i files che sono stati caricati su Pinata. Ogni volta che l'utente entrerà in questa schermata, sarà effettuata una chiamata API a Pinata per chiedere la lista dei cids dei file che sono stati caricati solo dall'utente scrittore di cui abbiamo inserito la public key, e, per ogni file, sarà effettuata una chiamata di tipo 'GET', per andare a recuperarlo, de-crittografarlo utilizzando la password salvata in memoria, e successivamente visualizzarlo sotto forma di card all'interno di una card list.
I file che sono stati scaricati verranno salvati in memoria locale, in modo tale che, se l'utente dovesse rientrare in questa schermata, sarà eseguita nuovamente la

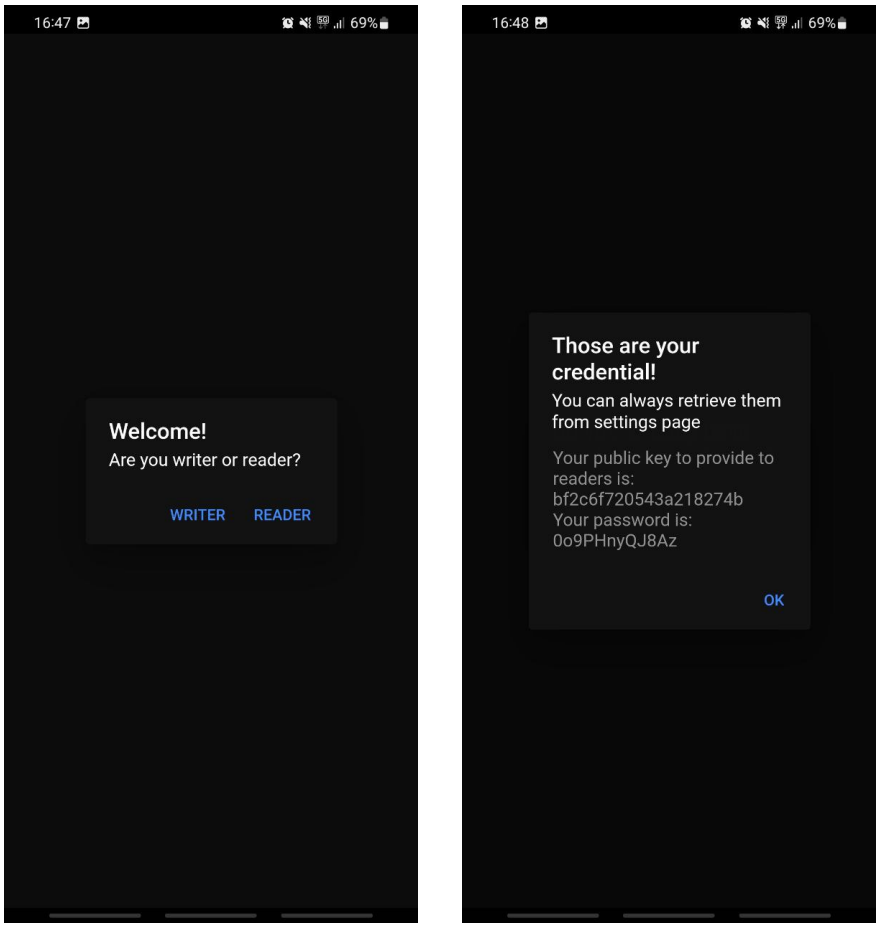

Figure 1: Schermata di login e di nodo registrato come writer

chiamata Api per richiedere la lista dei cids, e, se tale lista combacia con la lista già presente in memoria, significa che non ci sono aggiornamenti, di conseguenza non saranno effettuate le successive chiamate 'GET'. Questo passaggio ha lo scopo di non sovraccaricare il sistema e di non effettuare troppe chiamate a Pinata, per evitare malfunzionamenti del servizio. Cliccando su ciascuna card, sarà possibile visualizzare un alert che contiene informazioni aggiuntive sui dati inseriti da un utente come mostrato in figura 3.

5.4 Filtraggio dati

Sempre all'interno della tab di visualizzazione dei dati è possibile filtrare i dati che vogliamo visualizzare sulla base di quando sono stati caricati. Tramite l'apposito menù a tendina è infatti possibile scegliere di visualizzare i dati

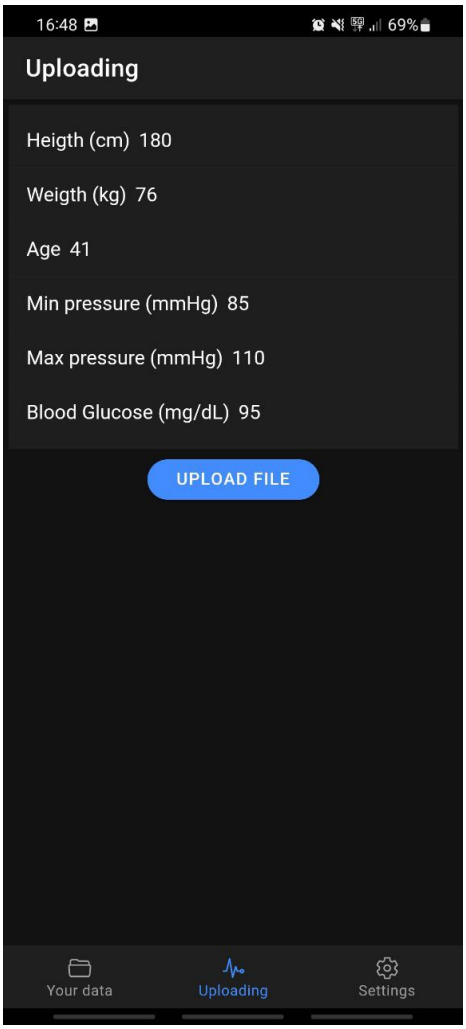

Figure 2: Caricamento dati

caricati l'ultimo giorno, durante l'ultima settimana o durante l'ultimo mese. L'applicazione dei filtri è mostrata in figura 5.4

5.5 Configurazione

Cliccando sulla tab 'Settings' verranno visualizzati tutti i dati relativi all'utenza. Nel caso di un nodo writer verranno visualizzate: la propria public key, la propria private key e la propria password di criptaggio/decriptaggio. Nel caso di un nodo reader invece verranno visualizzate la public key e la password di criptaggio/decriptaggio del writer di cui vogliamo leggere i dati. Inoltre viene

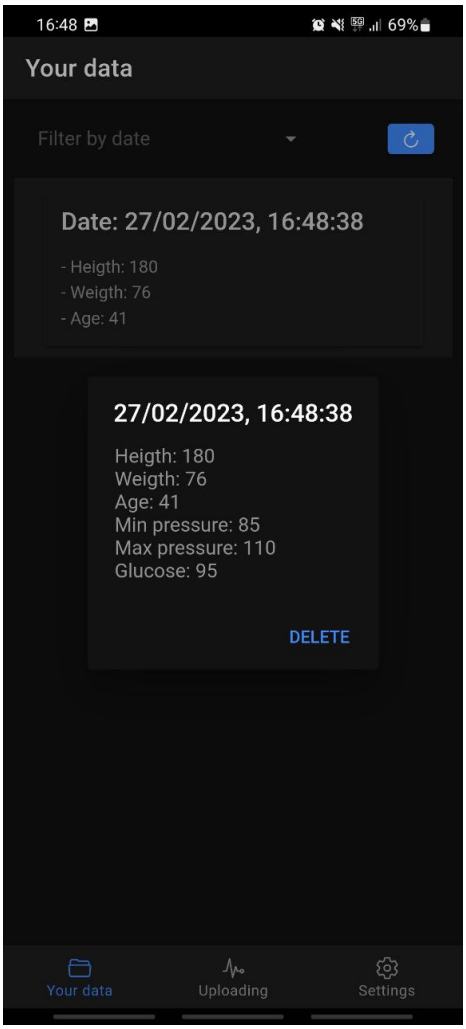

Figure 3: Visualizzazione dei dati

visualizzato un bottone che permette di effettuare il logout, questo cancella tutte le variabili di configurazione e permette un nuovo avvio da zero dell'applicazione per poter cambiare nodo da visualizzare o passare da writer a reader e viceversa.

6 Conclusioni e sviluppi futuri

La gestione di dati sensibili è un tema molto delicato, tramite la nostra applicazione abbiamo reso possibile la condivisione di questi in modo sicuro tramite il criptaggio, nonostante siano pubblici sulla rete IPFS. Questo permette anche

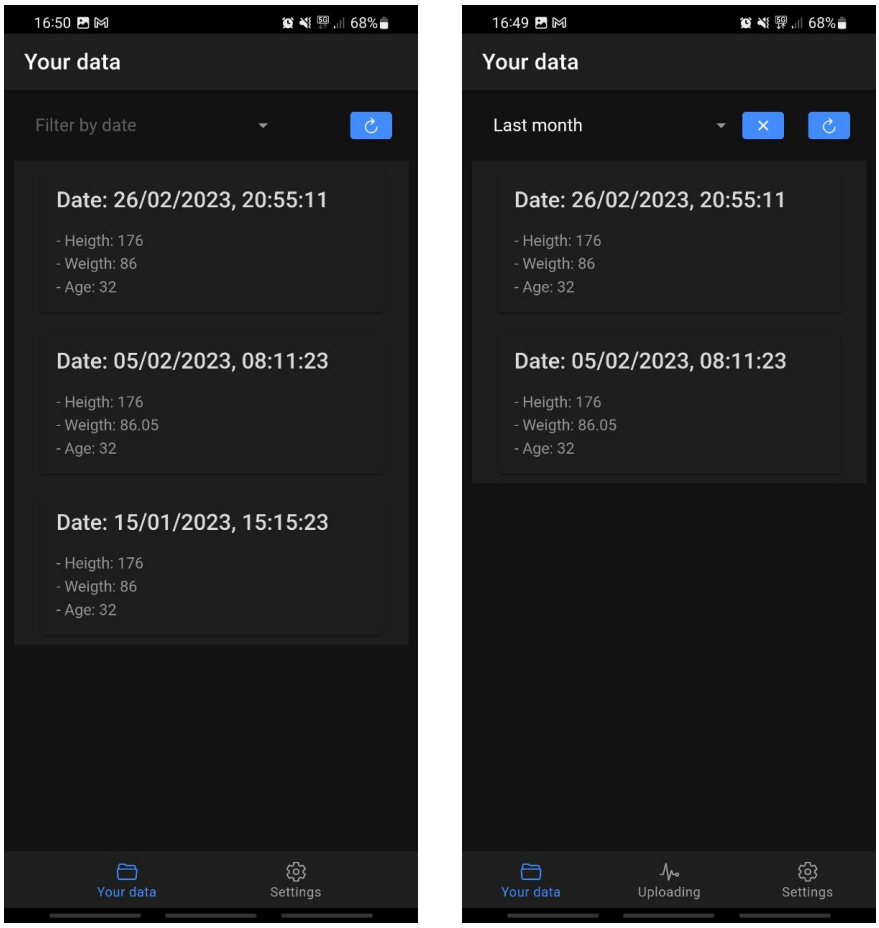

Figure 4: Schermata senza filtro e con filtro applicato

una facile reperibilità da parte di tutti coloro che vogliono visualizzare i dati e li possono decriptare.
Non è possibile effettuare test di scalabilità ed efficienza in quanto la versione gratuita di Pinata ha grosse limitazioni, infatti è possibile mantenere pinnati contemporaneamente solo 100 file con la dimensione massima di 1 GigaByte in totale, e per ognuno di essi il numero massimo di richieste è 15 al minuto. Inoltre mancano varie funzionalità che offrono i servizi a pagamento.
Come sviluppi futuri si potrebbero implementare nuove funzionalità, tra cui:

- Aggiunta di nuove variabili e diversi tipi di filtraggio.
- Visualizzazione dell'andamento nel tempo delle variabili attraverso grafici.
- Aggiunta della lettura automatica tramite sensori di dispositivi wereable, o tramite telefono. Per fare ciò si potrebbero utilizzare alcune librerie di

Ionic e Capacitor per andare a leggere i valori dei sensori, ma è consigliabile aggiungere queste funzionalità nel codice nativo (xCode e Android Studio), così facendo non potremmo però utilizzare l'applicazione in versione Desktop. Un'altra idea sarebbe quella di sfruttare la tecnologia del web of things [5] per poter comunicare con tutti i tipi di dispositivi, andando a introdurre un layer intermedio di raccolta di dati, garantendo così la possibilità di utilizzare anche la versione Desktop.

References

[1] Angular. https://ionicframework.com/.

[2] Ionic. https://angular.io/.

[3] Ipfs. https://ipfs.tech/.

[4] Pinata. https://www.pinata.cloud/.

[5] Web of things w3c. https://www.w3.org/WoT/.