Condividere informazioni in modo sicuro combinando Git e Blockchain

Paolo Speziali

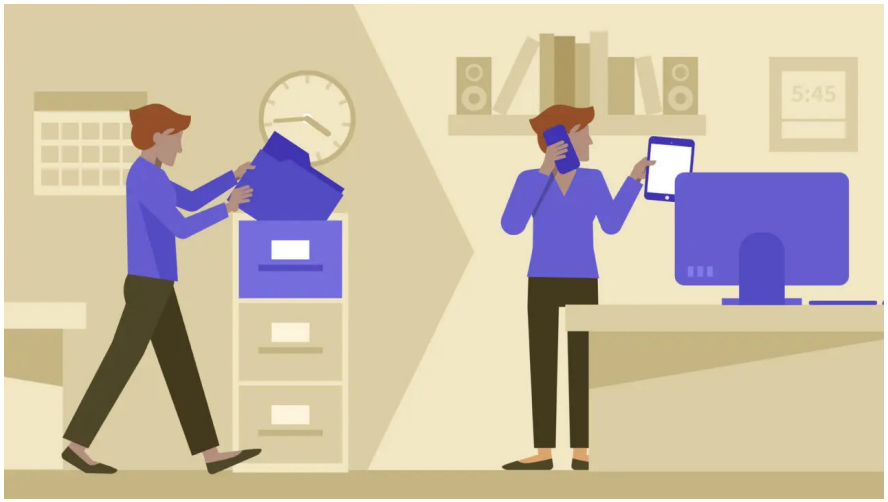
Università degli Studi di Perugia - Dipartimento di Ingegneria

A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

A.A. 2020/2021

La digitalizzazione

È in atto, negli ultimi anni, un piano di **digitalizzazione** delle PA. Esso mira all'evoluzione tecnologica di tutte le sue mansioni e alla creazione di portali web per il cittadino. L'esigenza di questa trasformazione si è fatta sentire anche da parte dell'Unione Europea, che con il Recovery Fund ci sta fornendo i fondi per attuarla.

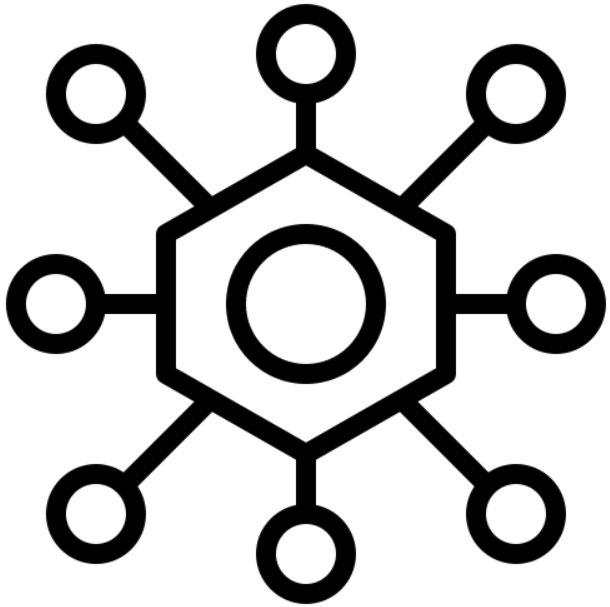


Il problema della burocrazia

Tuttavia, anche avendo i fondi necessari, sono molti i problemi che non permettono una digitalizzazione totale delle PA, tra cui la lenta e farraginosa macchina della burocrazia. Sembra necessario un processo di **sburocratizzazione** grazie a degli strumenti digitali che permettano di salvare, validare e condividere documenti in maniera sicura.

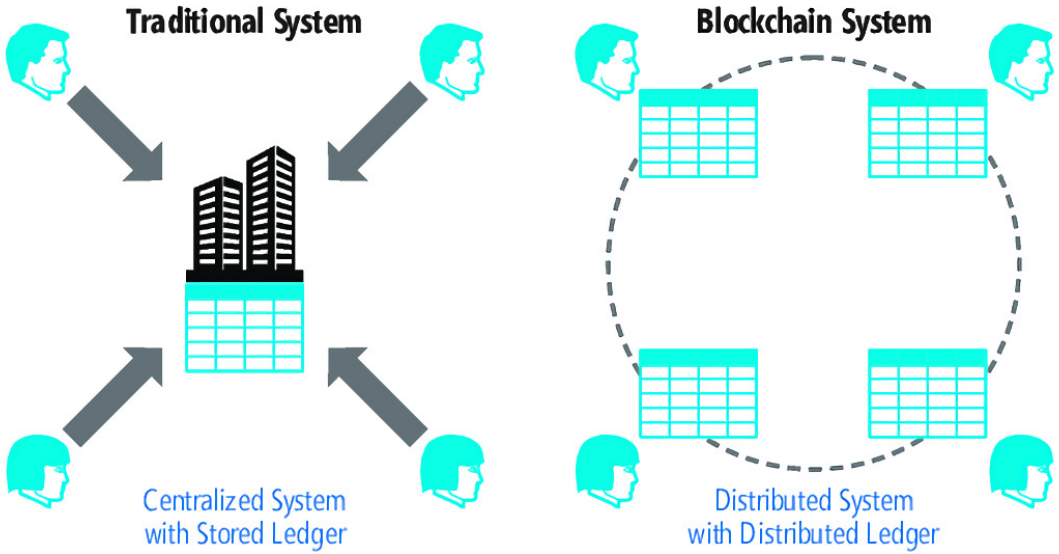
Gli strumenti attuali

Gli strumenti attualmente in utilizzo hanno un'architettura centralizzata: un'entità centrale si occupa dell'immagazzinamento e della verifica dei dati degli utenti. Ciò è potenzialmente rischioso, sia perché potrebbero verificarsi attacchi alle unità centrali, sia perché mettiamo in mano di un'azienda esterna i nostri dati.



Strumenti decentralizzati

Usando invece strumenti decentralizzati, sia per la gestione dei file, per cui utilizzeremo **Git**, sia per la verifica delle informazioni, per cui useremo la **blockchain**, saremo in grado costruire uno strumento che può affidarsi alla parola di una moltitudine di entità, rendendo molto più complicati e rilevabili attacchi e manomissioni.



Fase 2: Specifica dei requisiti

Partendo da questa idea lo step successivo è stato la definizione dei requisiti del progetto. Abbiamo perciò concordato che l'attività potesse orientarsi verso la creazione di un tool universale per la registrazione dell'impronta digitale (tramite funzioni di hashing) di insiemi di file, da noi battezzati Storage Unit, su Blockchain, utilizzando gli strumenti messi a disposizione da Git per gestirli meglio.
Il nome scelto per l'applicazione è stato **PineSU**.

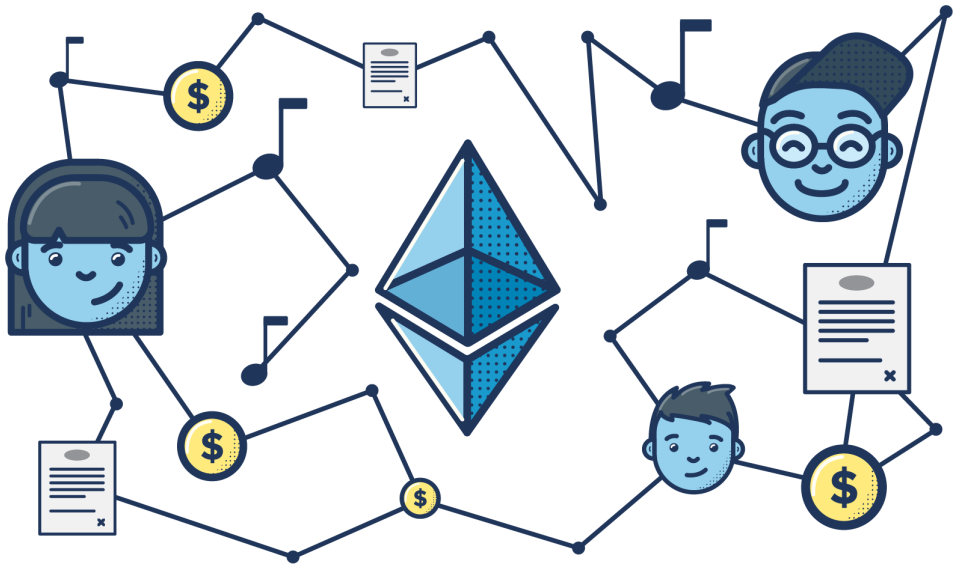
Fase 2: Specifica dei requisiti

L'applicazione avrebbe dovuto quindi fornire all'utilizzatore le funzionalità del software Git corredando il tutto con la possibilità di registrare e, successivamente, verificare l'integrità delle Storage Unit. Una volta creata una Storage Unit essa deve essere immutabile, dando tuttavia la possibilità di esportarne singoli file dotandoli sempre di meccanismi di controllo.

Figura: Il logo dell'applicazione

Excursus su Ethereum e Blockchain

La tecnologia di Ethereum permette la creazione di una rete distribuita e decentralizzata finanziata e incentivata ad operare e rimanere attiva mediante la creazione, lo scambio e l'utilizzo dell'omonima criptovaluta.

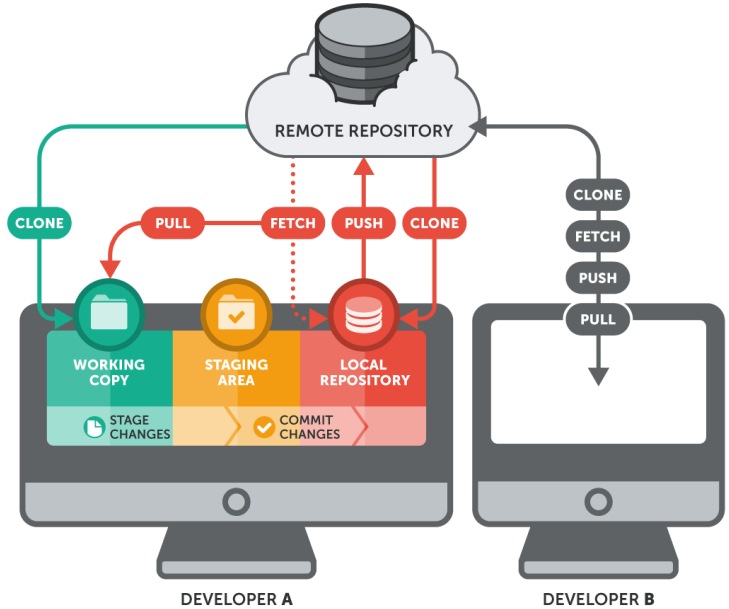


Excursus sulle Applicazioni Distribuite

Su tale rete è possibile mettere a disposizione delle applicazioni che chiunque può utilizzare dietro pagamento di una piccola commissione. Queste applicazioni sono realizzabili dagli sviluppatori interessati tramite vari framework e suite di applicativi, il più celebre è **Truffle**.

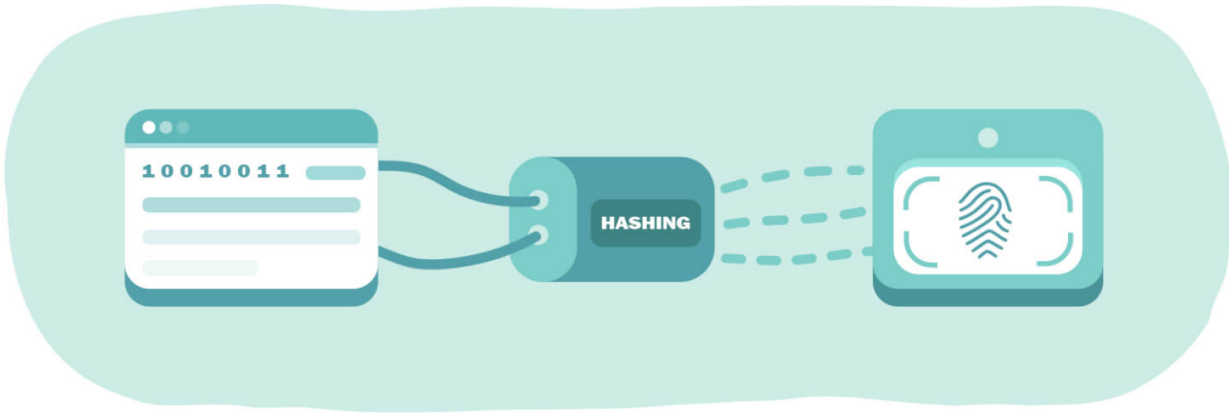
Excursus su Git

Git è un software che permette in maniera semplice di gestire insiemi di file tramite un sistema di controllo di versione, una grande risorsa per gli sviluppatori che devono contribuire in maniera condivisa ad uno stesso progetto o che devono tenere sotto controllo i vari cambiamenti che sono stati apportati ai vari file e documenti.



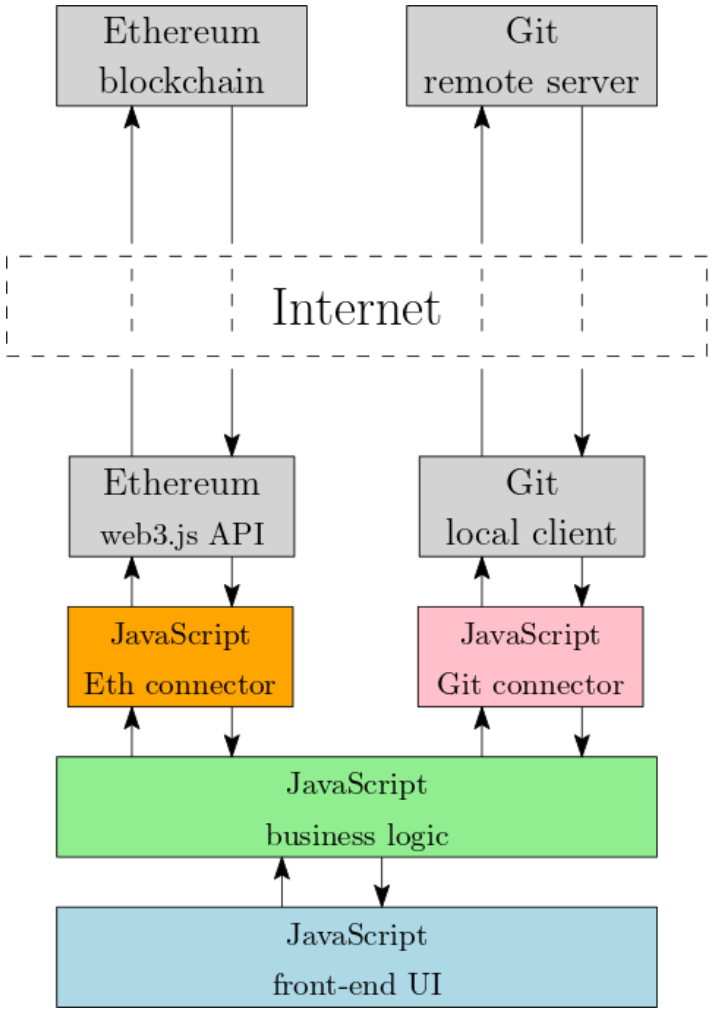
Excursus sulle funzioni di hashing

Le funzioni di hashing sono funzioni non invertibili che permettono di associare in maniera univoca (o quasi) stringhe di caratteri (e quindi anche documenti di varia natura tradotti in stringhe) a delle stringhe alfanumeriche di lunghezza fissa.

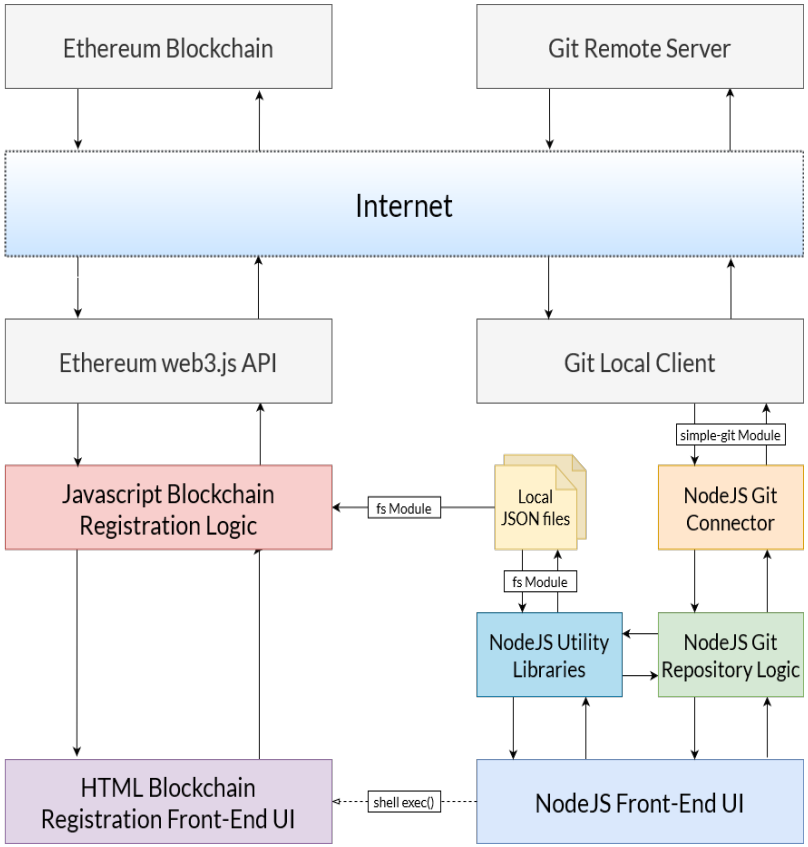


Fase 3: Progettazione - Prima stesura

La stesura iniziale è stata svolta dal professore che mi ha quindi fornito una linea guida da cui poter prendere spunto per poter realizzare il progetto dopo aver concordato sulle sue funzionalità, tale architettura è stata tuttavia modificata abbastanza in quanto le tecnologie utilizzate sono ancora troppo sperimentali e non offrono gli strumenti per potere essere adattati ad una struttura del genere.



Fase 3: Progettazione - Seconda stesura



Il Front-End in NodeJS che comunica con una parte logica è stato mantenuto, sono stati però introdotti alcuni moduli Utility che vengono utilizzati da entrambe le componenti.
La connessione alla Blockchain avviene attraverso l'avvio di un Web Server apposito e l'apertura di una scheda del browser, l'interazione da parte dell'utente deve avvenire tramite l'add-on **Metamask**.

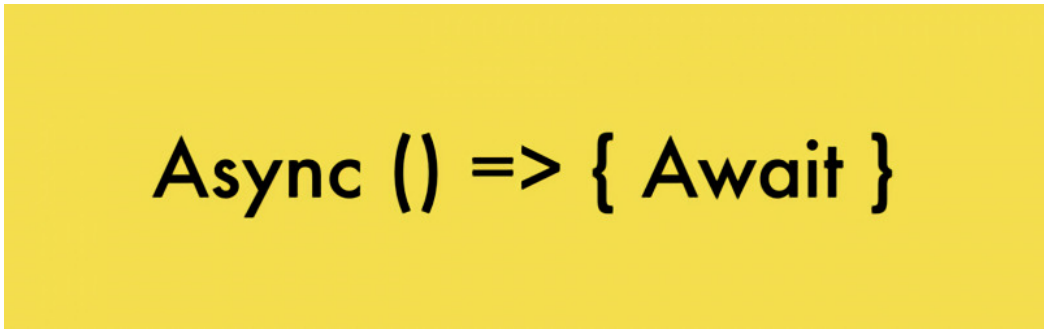
Fase 4: Apprendimento

In concomitanza con la fase di progettazione è stato necessario l'apprendimento di alcune tecnologie in modo da poter imparare le loro modalità di utilizzo e poter effettuare la seconda stesura dello schema progettuale.
Oltre ai già citati Truffle Suite e Metamask, abbastanza intuitivi nell'utilizzo, le tecnologie che seguono sono quelle che hanno occupato la maggior parte di questa fase.

Fase 4: Apprendimento - Solidity

Per l'apprendimento della sintassi e delle peculiarità del linguaggio da utilizzare per scrivere DAPP per la Blockchain Ethereum mi sono avvalso della documentazione ufficiale e del tutorial interattivo **CryptoZombies**.

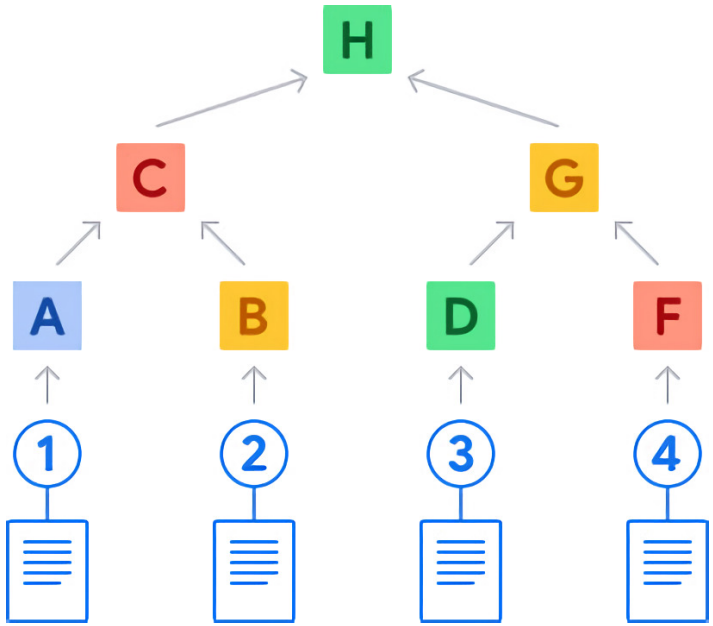
Fase 4: Apprendimento - Javascript Asynchronous Programming



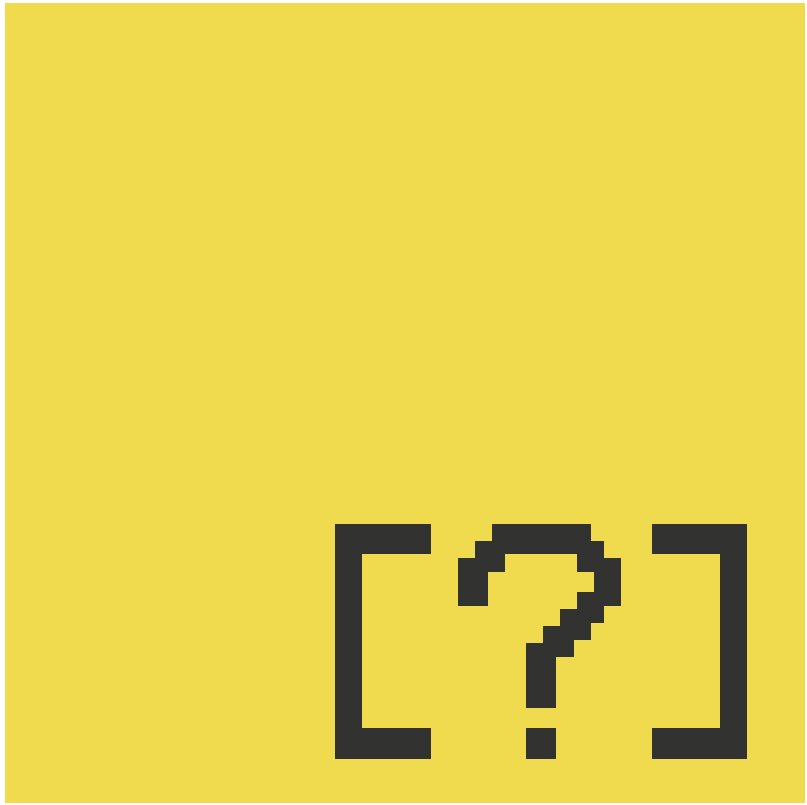
L'utilizzo di alcuni moduli all'interno dell'applicazione ha richiesto che io spendessi diverso tempo ad imparare le tecniche di programmazione asincrona di Javascript in quanto lo scorretto utilizzo delle keyword async e await sono state fonte di svariati problemi nella prima fase della stesura del codice.

Fase 4: Apprendimento - Modulo Merkle Tree

La necessità di dover calcolare una singola stringa Hash per una moltitudine di file ha portato il professore a proporre l'utilizzo di questa struttura, è stato quindi necessario da parte mia non solo comprenderne bene il funzionamento ma anche essere in grado di poter utilizzare al meglio i moduli che fornivano metodi per lavorare con questi particolari alberi.



Fase 4: Apprendimento - Modulo InquirerJS



Anziché realizzare un Front-End dotato di GUI, ho preferito optare per una interfaccia testuale. Tuttavia, non volendo rinunciare all'immediatezza e la semplicità che un approccio grafico e user-friendly poteva portare al progetto ho imparato ad utilizzare il modulo di InquirerJS, il quale consente di realizzare menù a scelta singola, multipla e libera in tutta comodità.

Fase 5: Implementazione

La fase di implementazione è divisibile in tre macro-sezioni:

1. Creazione delle librerie di utility e dei moduli di interfacciamento con Git
2. Creazione della CLI e del flow di interazione con le librerie e la logica di Git
3. Creazione del modulo di interrogazione e registrazione per la Blockchain

Creazione delle librerie di utility e dei moduli di interfacciamento con Git

La prima stesura di codice è avvenuta nella creazione della classe “connettore” per Git con il modulo “simple-git” e del relativo modulo Logic il quale richiama le sue funzioni a seconda della necessità della CLI.

Creazione delle librerie di utility e dei moduli di interfacciamento con Git

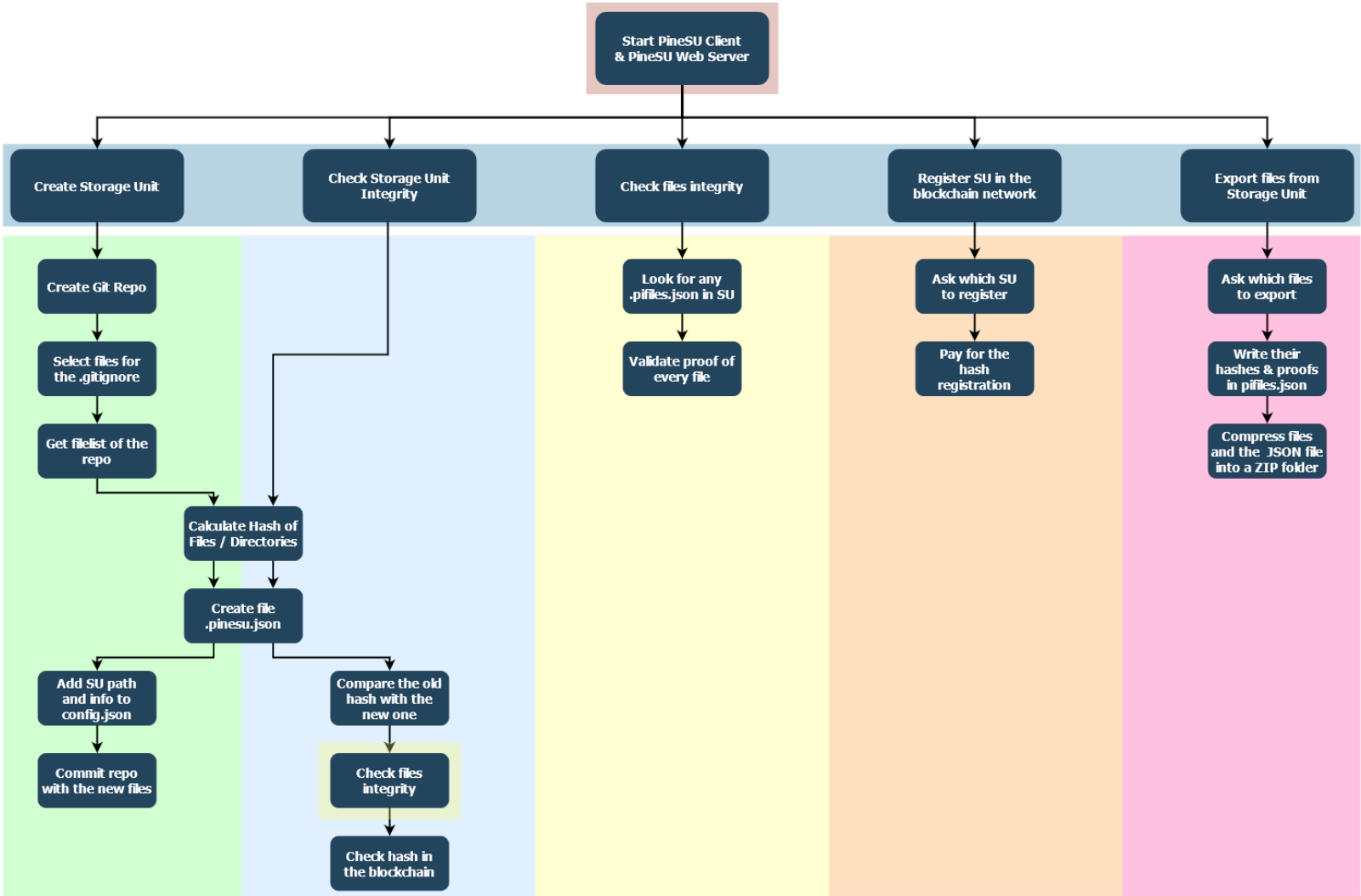
In concomitanza ho scritto i moduli del package "lib":

- files: Lettura e scrittura di file JSON in cui conservare le informazioni riguardanti la Storage Unit o l'utente che sta utilizzando l'applicativo;
- inquirer: Contiene tutte le scelte che vengono poi presentate all'utente nella CLI;
- treelist: Si occupa di effettuare tutte le operazioni riguardanti l'hashing di file, l'assegnazione di hash alle subdirectories e la creazione e gestione di Merkle Tree.

Creazione della CLI e del flow di interazione con le librerie e la logica di Git

Ho proseguito andando a creare l'effettivo workflow del programma richiamando le scelte dal modulo inquirer e funzioni differenti in base alle selezioni dell'utente.

Creazione della CLI e del flow di interazione con le librerie e la logica di Git - Workflow



Creazione del modulo di interrogazione e registrazione per la Blockchain

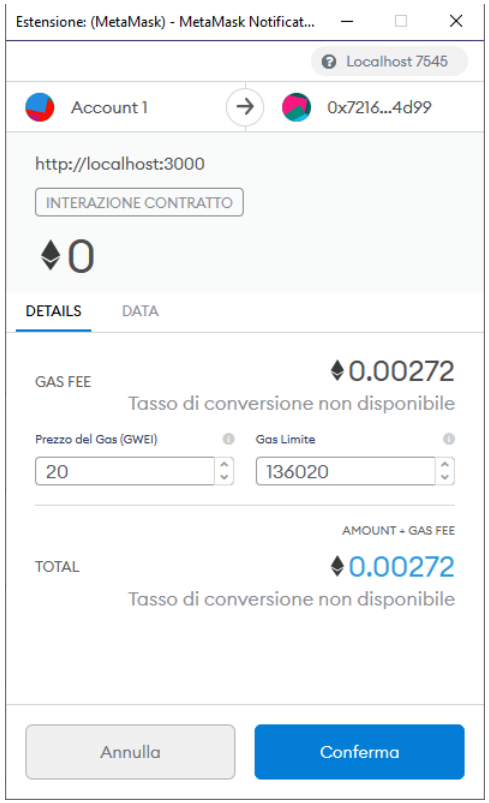


Figura: Transazione per registrare un hash nella blockchain

L'ultima parte della fase di implementazione è stata la realizzazione del Web Server locale che permette all'utente di registrare gli hash delle proprie Storage Unit nella blockchain, il risultato è stato ottenuto con una versione pesantemente modificata dell'applicazione *sample* fornita dal sito della suite Truffle.

Resoconto

- Data di inizio tirocinio: 15/2/2021
- Data di fine tirocinio: 28/5/2021
- Totale Ore: 150 (25 ore · 6 CFU)
- Professore Tutor: Luca Grilli
- Studente Tirocinante: Paolo Speziali
- Anno Accademico: 2020/2021

Fonti

- Strumenti Ethereum per sviluppatori
- Tutorial Truffle DAPPs - Pet Shop
- Build a JavaScript CLI with Node.js
- Tutorial di Mozilla su async / await
- Immagini reperite dai siti ufficiali degli strumenti eccetto per alcune scaricate da queste pagine web:
  - Funzioni di Hashing
  - Concorso pubblico
  - Ethereum Blockchain
  - Git repository
  - async / await
  - Merkle Tree
- Strumento di upscaling delle immagini