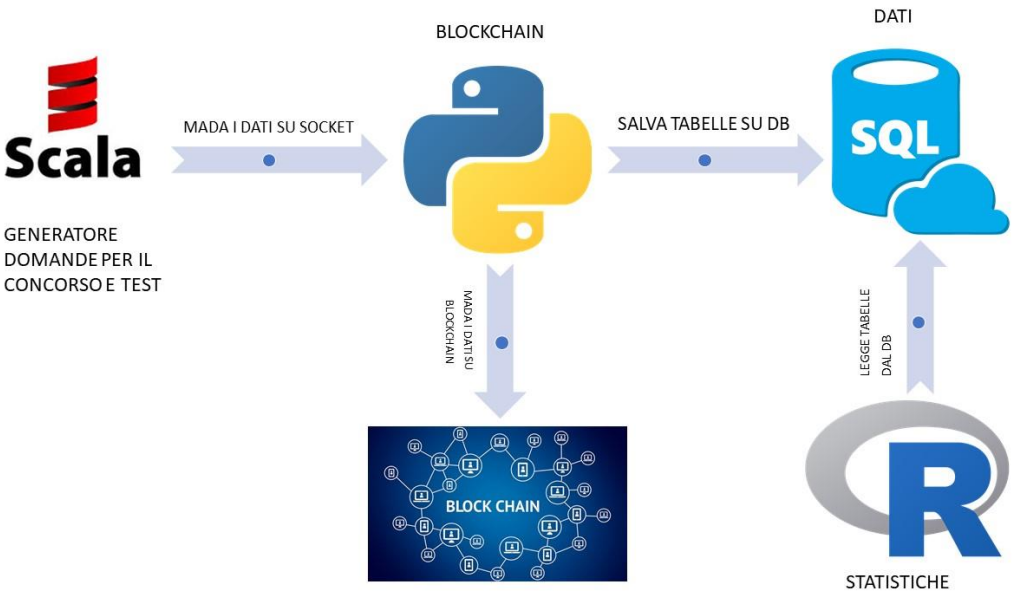
Progetto_APL

Giovanni Fausto - Alessia Rondinella

Blockchain per test

L'obiettivo di questo progetto è la creazione e gestione di test a risposta multipla, utilizzando la tecnologia blockchain.

L'immagine di seguito mostra la struttura del progetto e i linguaggi di programmazione utilizzati:



In particolare, nel progetto sono presenti i seguenti componenti:

lettoreDomande (Fausto):

Lettura di un file csv, contenente 700 domande che fanno riferimento a 7 categorie diverse, vengono generati casualmente dei test contenenti 70 domande a risposta multipla, cioè 10 domande per ogni tipologia. Inizialmente vengono lette le domande dal file e quindi messe in una lista contenente una HashMap che ha come key una stringa e come value un'altra stringa. Successivamente vengono generati 70 numeri random in modo tale da essere tutti diversi e in modo tale di essere raggruppati a 10 a 10, ad esempio i primi 10 sono scelti tra 1 e 100, gli altri da 101 a 200 e così via. Alla fine quindi avremo ancora una lista di HashMap con key stringa e value stringa.

Concorso (Fausto):

Una volta generato il test, viene gestita l'esecuzione vera e propria del test, in cui in particolare, è possibile scegliere due modalità di esecuzione del test:

- Simulazione: modalità in cui viene chiesto all'utente il numero di test di cui effettuare una simulazione.

- Svolgimento del test: modalità in cui l'utente partecipa attivamente al test, rispondendo alle varie domande.

In entrambe le modalità, si procede inserendo nome, cognome e codice del partecipante, e successivamente si procede allo svolgimento del test, scegliendo per ogni domanda, una tra le possibili 4 opzioni, prese anch'esse dal file cvs e sono mescolate tra di loro perchè altrimenti la prima risposta sarebbe sempre quella corretta.

Le risposte vengono valutate utilizzando il seguente criterio:

- +1 per ogni risposta corretta;
- 0 per ogni risposta errata in fase di simulazione, altrimenti - 0.25;
- +0.25 se non viene fornita una risposta in fase di simulazione, altrimenti 0;

Alla fine del test, i dati del partecipante ed il test, comprensivo delle domande uscite e punteggi ottenuti per ogni domanda, vengono inviati al client py utilizzando le Socket.

Blockchain e Block (Rondinella):

Creazione della classe Block per l'archiviazione delle transazioni e della Blockchain che permette di collegare gli oggetti block creati, attraverso la definizione di metodi che permettono la gestione della stessa.

In particolare, ogni blocco contiene gli stessi attributi. Gli oggetti Block creati sono collegati insieme all'interno della struttura dati che costituirà la Blockchain. Per garantire l'immutabilità, ogni blocco al suo interno contiene l'hash del blocco precedente, quindi, in caso di modifica di un blocco, tutti gli altri conterranno hash che non sono corretti, invalidando la catena.

Gli attributi che caratterizzano i blocco sono: *index*, che è l'indice del blocco; *transazioni*, che è una lista che memorizza le transazioni, i test dei partecipanti con le loro informazioni; timestamp, che indica il momento della creazione; *nonce*, che è un numero, inizialmente zero, incrementato finché l'hash non soddisfa il vincolo di difficoltà imposto per la creazione del blocco; *hash*, che è l'hash del blocco che soddisfa il Pow.

La classe Blockchain è responsabile dell'aggiunta di nuovi blocchi e di archiviare all'interno di essi le transazioni, che coincidono con il test effettuato da un singolo partecipante.

Server (Fausto/Rondinella):

Realizzazione di un server contenente gli endpoint per interagire con essa. Il progetto è implementato su un'architettura RESTful, che utilizza il formato standard JSON per i messaggi per la comunicazione C/S. In particolare, viene utilizzato il framework Flask per implementare le chiamate RESTful. E' possibile utilizzare un Web client qualsiasi per interagire con gli endpoint RESTful, come Postman o curl. E' anche possibile utilizzare una piccola interfaccia web in html che è stata creata appositamente per fare le get più utili, come ad esempio quella per visionare la blockchain.

Gli endpoint disponibili sono:

- /nuovaTransazione - POST - per la creazione di una nuova transazione in un blocco;
- /pending - GET - per verificare se ci sono altre transazioni non ancora confermate;

- /mine - GET - per l'estrazione del nuovo blocco;
- /chain - GET - permette di visualizzare l'intera blockchain;
- /partecipanti - GET - restituisce l'elenco dei partecipanti al test e il punteggio totale ottenuto da ognuno.
- /partecipanti/<codice> - GET - dato il codice di un partecipante, restituisce il test effettuato dal partecipante e le sue informazioni.

Il server, inoltre, esegue il salvataggio della bc, per poterla ricaricare in seguito, per non effettuare la creazione ogni volta. Il salvataggio viene fatto utilizzando il modulo Pickle per la serializzazione della struttura dell'oggetto bc. Si occupa anche del salvataggio dei vari risultati ottenuti su un server MySQL che serviranno per poi essere attenzionati da R.

Client (Rondinella):

Viene realizzato un client per la ricezione Socket del test effettuato, si occupa successivamente di effettuare la richiesta GET di creazione della transazione da inserire in BC e la richiesta POST per effettuare il mine del blocco. Ogni volta che viene aggiunto un novo blocco, e quindi che viene fatto il mine, viene aggiunto un nuovo valore anche alla dataframe, questa a sua volta viene salvata su un DataBase MYSql sotto forma di tabella, in modo tale che sia poi semplice usare questi dati per fare delle statistiche su R.

MySql:

La persistenza dei dati è implementata attraverso un database MySql, a tal fine viene creato un database apl, che conterrà due tabelle. Una contenente i punteggi di tutti i candidati, e l'altra contenente sempre punteggi ma relativi alle varie categorie.

R (Fausto/Rondinella):

Le statistiche sono state implementate in R in quanto è il più portato dei tre linguaggi per questo tipo di operazioni.

In particolare, abbiamo pensato di calcolare 5 differenti statistiche:

- La media dei punteggi relativi alle varie categorie, principalmente per capire qualche categoria è andata peggio o meglio, e quindi in futuro andare a cambiare la tipologia di domande;
- I punteggi totali di ogni candidato, andando a metterli in ordine decrescente in modo tale da rendersi subito conto di chi ha fatto un punteggio più alto;
- Abbiamo anche riportato la media dei punteggi delle categorie su un grafico a torta in modo tale da renderlo più comprensibile;
- I punteggi del peggiore e migliore candidato;
- La distribuzione dei punteggi, per capire in media come sono andati i test;
- Infine, un grafico riassuntivo dei precedenti.