



Università degli Studi di Torino  
Facoltà di Scienze della Natura  
Corso di Laurea in Informatica

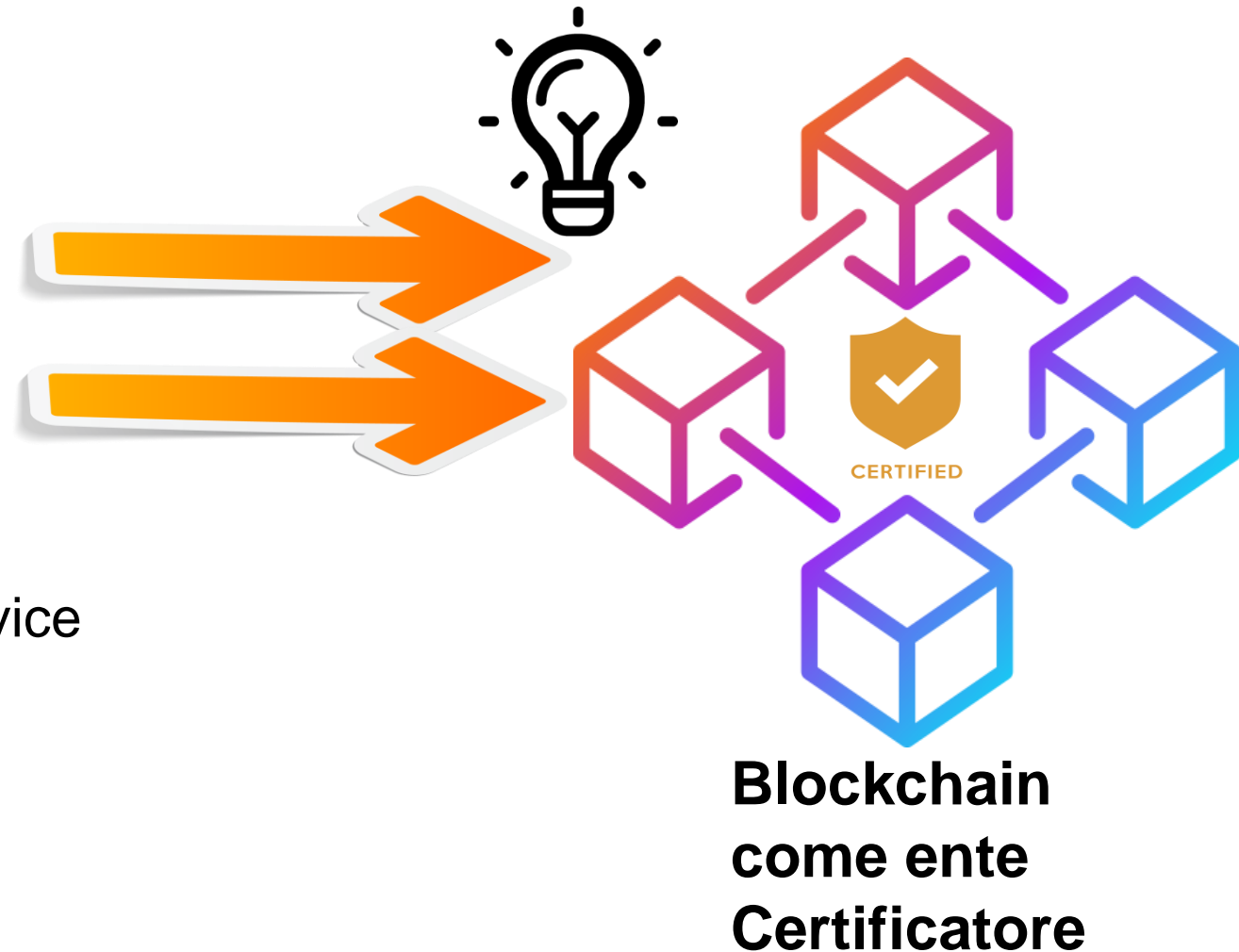
# Studio e realizzazione di un prototipo di un sistema basato su blockchain per il mobility as a service

Candidato: Giorgio Mecca  
Matricola: 880847

Relatore: Claudio Schifanella

# Obiettivi della Tesi

- Salvataggio di informazioni
- Certificazione di informazioni
- Costruzione di un prototipo di un applicativo utile al Mobility as a Service



# La Blockchain

DLT – Distributed Ledgers Technology

Struttura:

- Nodi
- Registro
- Blocchi
- Transazioni
- Hash Pointing

Proprietà:

- Immutabilità del registro
- Trasparenza
- Tracciabilità delle transazioni
- Sicurezza
- Condivisa, tra i nodi



# Scenario di partenza

Al giorno d'oggi molte delle informazioni riguardanti tratte e spostamenti sono salvati utilizzando dei fogli di calcolo



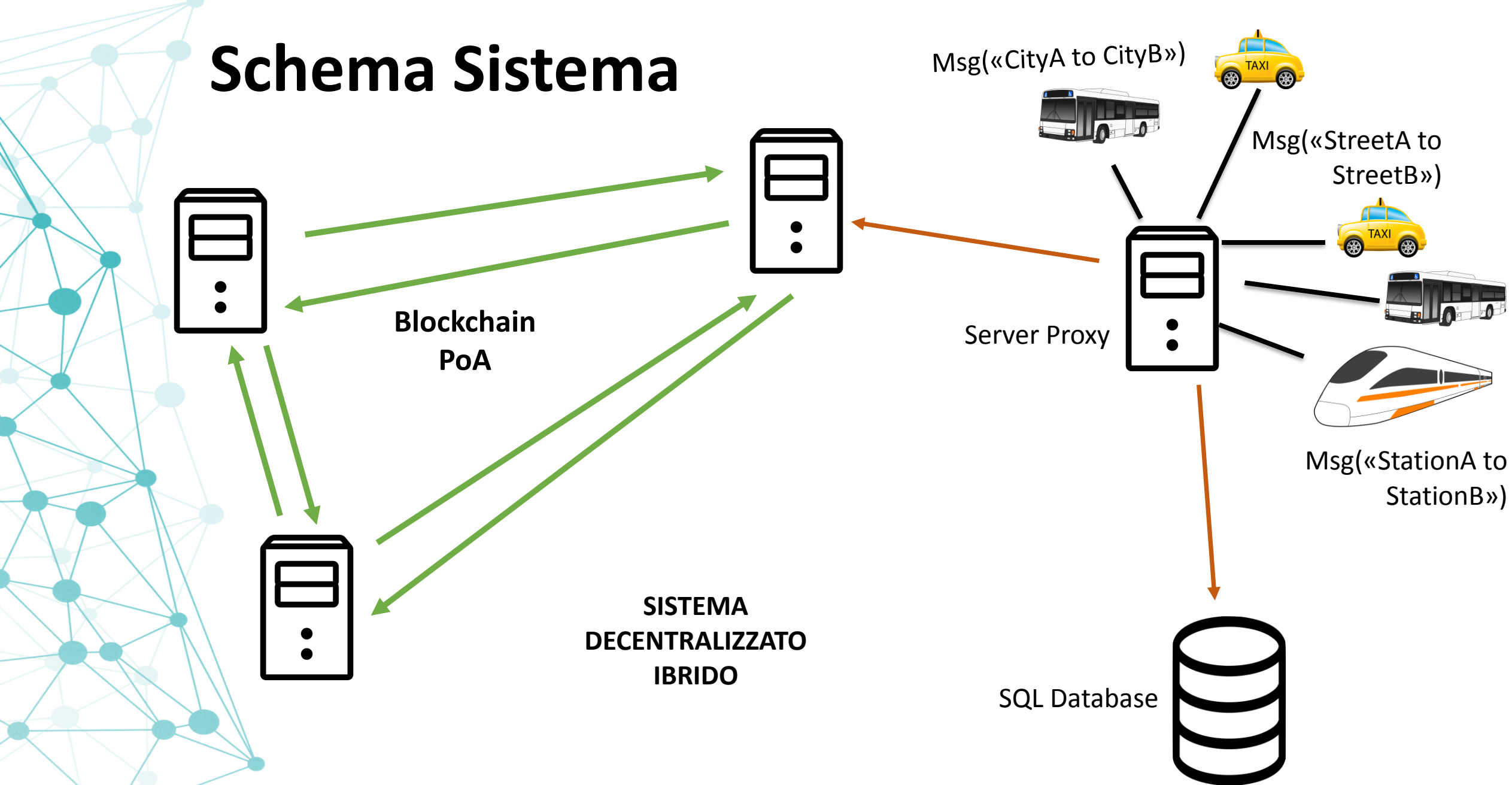
## Vantaggi

- Veloce
- Intuibile

## Svantaggi

- Assenza di sicurezza
- Difficoltà nelle interrogazioni
- Disorganizzazione nella memorizzazione

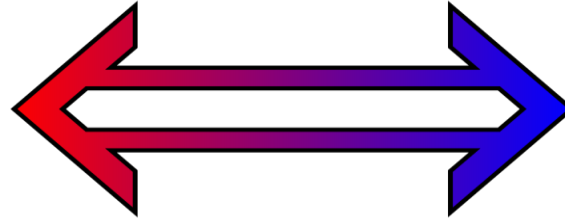
# Schema Sistema



# Attori

- attori nella Blockchain

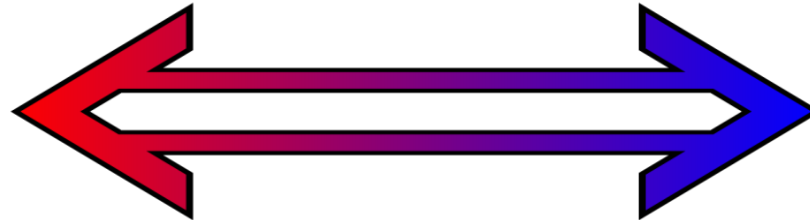
- Enti
- Owner/ Ente centrale



indirizzo: 0x67...(\*40)  
Chiave privata: 0x34...(\*64)

- attori nell'App

- User
- Admin



Username: Account2Ente1  
Password: \*\*\*\*

# Attori nell'app

User



Compiti:

Inserimento dei dati in tempo reale:

ID veicolo

Posizione iniziale

Possibili fermate

Posizione finale

A light blue rectangular box containing a form. At the top, the text 'ADD Vehicle ID' is centered. Below it is a white rectangular input field. At the bottom is a red rectangular button with the text 'Start Trace (Send Position)' in black.A light blue rectangular box representing a user interface. At the top, a rounded rectangle contains the text 'Benvenuto NormalUser01'. Below this, a larger rounded rectangle contains three red buttons stacked vertically. The buttons are labeled 'Save Stand (Send Position)', 'End Trace', and 'DeleteTrace' from top to bottom.

# Attori nell'app

Admin



Compiti:

- Possedere e utilizzare la private Key
- Interrogare il sistema

Benvenuto AdministratorUser

gg/mm/aaaa

Send

Private Key:

Send



# Attori nella Blockchain

Compiti:

Enti



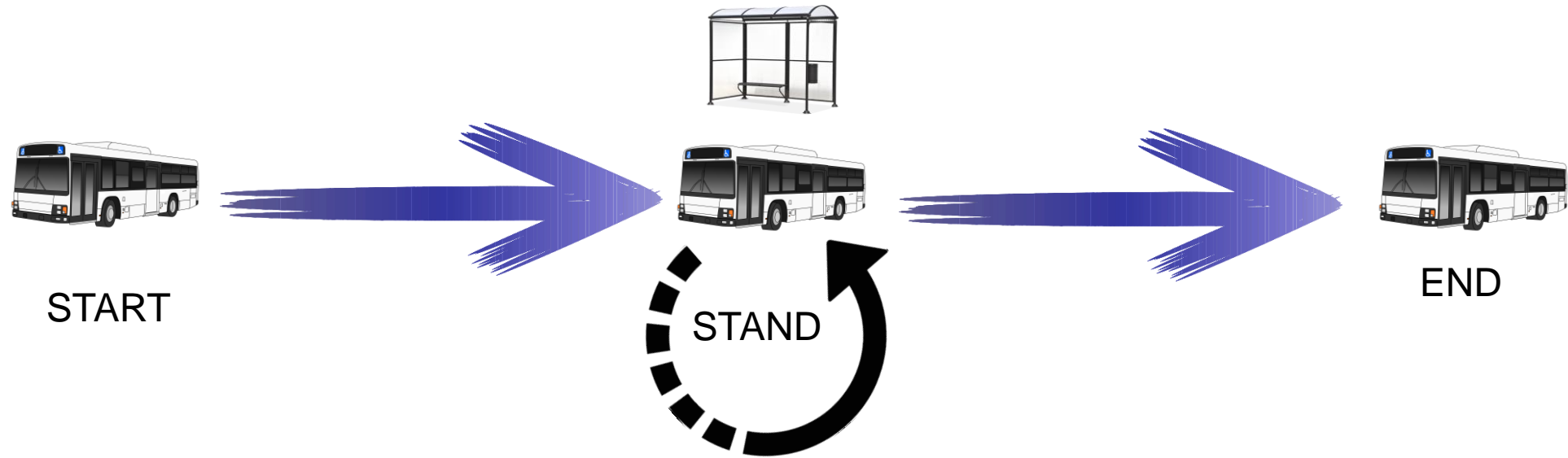
Effettuano transazioni  
per il salvataggio di  
informazioni nella  
blockchain

Owner



- Gestisce i permessi di utilizzo dei contratti
- Gestisce i nodi della blockchain

# Informazioni



I dati raccolti vengono salvati utilizzando la modalità centralizzata

I dati corrispondono ad una Tratta effettuata da:

- Un utente / dipendente
- Un determinato veicolo
- Varie Fermate / Stand

Più tratte sono raggruppate da un medesimo identificativo chiamato ID Hash

# Memorizzazione Centralizzata

Database SQL

Utente
Identificativo
Mail
Password

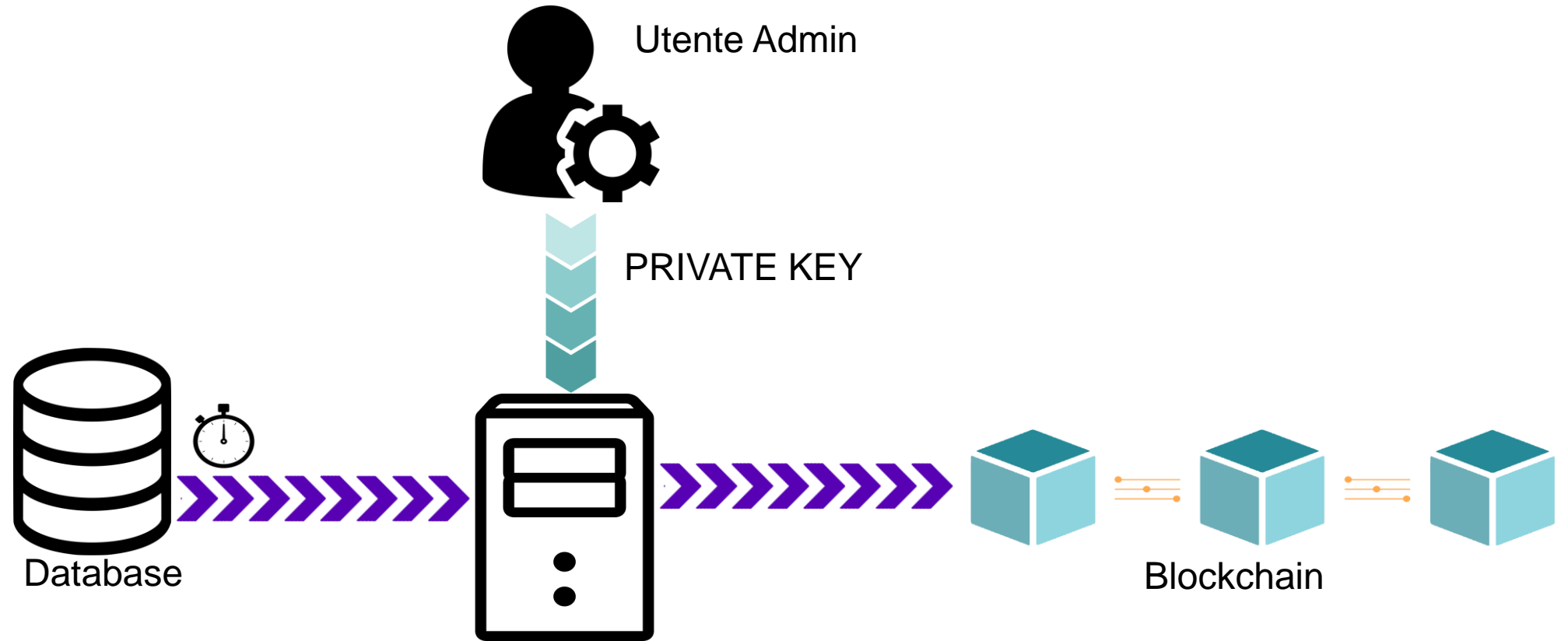
Tratta
Identificativo
Identificativo Veicolo
Identificativo Utente
Posizione Partenza
Posizione Arrivo
Data Partenza
Data Arrivo
ID Hash

Fermata
Identificativo Tratta
Posizione
Data

Fermata
Identificativo Tratta
Posizione
Data

IDHash
Address Ente
ID Hash
Transaction Hash
Data


# Utilizzo Ibrido




# Operazioni del server

Il server periodicamente :

1. Controlla per ogni ente se si sono salvate nuove tratte
2. (Se passo 1 positivo) "blocca" quest'ente



Gli utenti non possono più inserire nuove tratte finchè un utente Admin del medesimo Ente non inserisce la Private Key

- 
3. Raggruppa le tratte con ID Hash uguale e ne calcola il codice Hash
  4. Crea la transazione verso la blockchain e salva il codice hash contrassegnandolo con l'ID Hash

# Esempio di transazione

```
const contractInstance = new web3.eth.Contract(deployedContractAbi, deployedContractAddress);
const ABIMethod = await contractInstance.methods.add(ID_hash, hash_string.toString()).encodeABI();

const account = web3.eth.accounts.privateKeyToAccount(AccountPrivateKey);

const rawTxOptions = {
  from: account.address,
  to: deployedContractAddress, //address of contract
  value: '0',
  data: ABIMethod, //send value (ABI of method + params)
  gasPrice: '0x00', //ETH per unit of gas
  gas: '0x47b760', //max number of gas units the tx is allowed to use
};
```

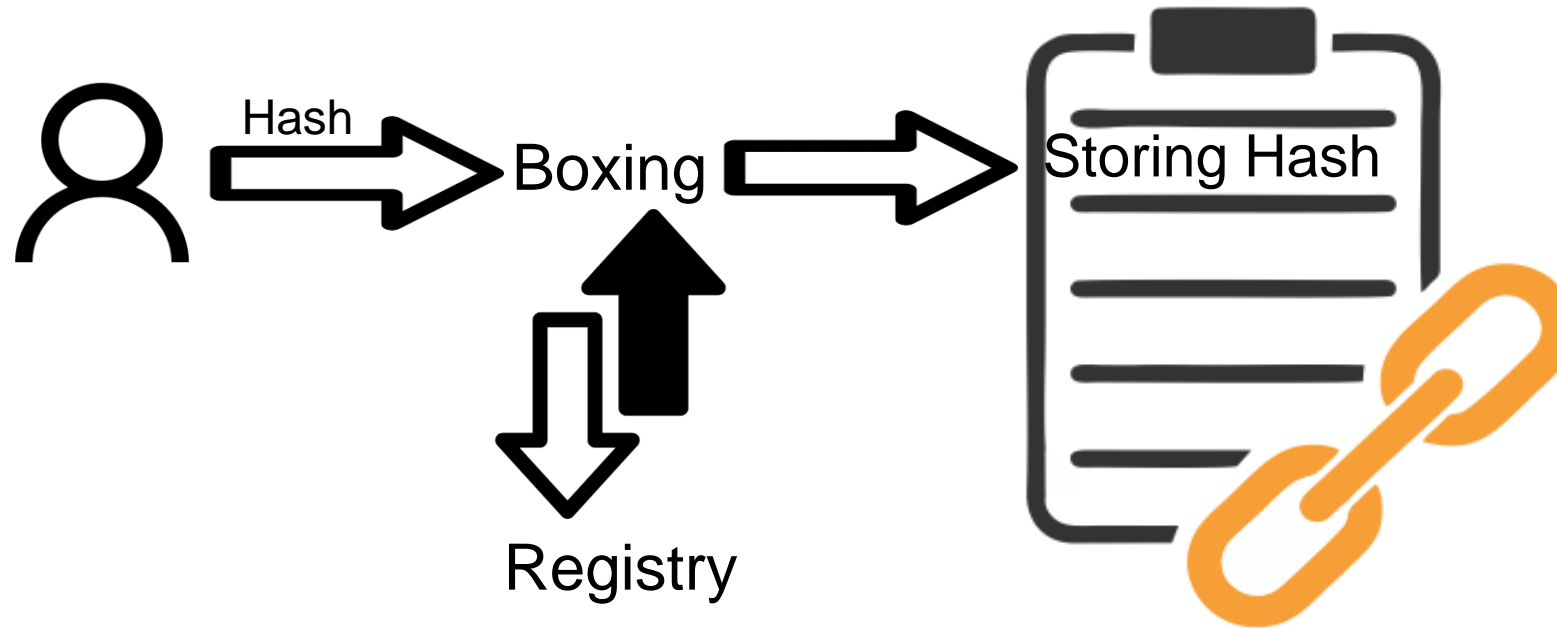
Viene creata una transazione grezza con destinazione l'address del contratto

```
var TransactionHash = web3.eth.accounts.signTransaction(rawTxOptions, AccountPrivateKey).then(async function(result){
  web3.eth.sendSignedTransaction(result.rawTransaction.slice(2),)
  .on('transactionHash', (hash) =>
    callback(null, hash)
  )
  .on('error', (error) => callback(error, '0'));
});
```

Viene firmata con la chiave privata ed infine inviata



# Smart Contract



Esistono 3 contratti:

- Boxing: interfaccia per i vari enti partecipanti al consorzio: "BusinessTravel.sol"
- Registry: contratto che funge da registro per gli utenti/Enti e le loro autorizzazioni: "BusinessRegistry.sol"
- Storing Hash: Utilizzato per il salvataggio dei vari hash identificati da un ID: "Travel.sol"



# Contratti

## BusinessTravel:

È il contratto con cui gli utenti della blockchain si interfacciano

Possiede come attributi 2 mapping:

1. Permette di assegnare un contratto Travel ad ogni Ente
2. Controlla se un ente è già inizializzato o meno

Questo contratto è un Boxing di Travel (non cambia la funzione del contratto e si pone come interfaccia)

```
contract BusinessTravel {  
  
    event TravelHash(address indexed _from, int ID,string travel_hash);  
  
    modifier AuthorizedAddress {  
        require(br.isAuthorized(msg.sender), "User not Authorized");  
        _;  
    }  
  
    mapping(address => Travel) travel;  
    mapping(address => bool) is_inialized;  
  
    BusinessRegistry br;  
  
    constructor (address _t) public {  
        br = BusinessRegistry(_t);  
    }  
  
    function add(int ID, string memory travel_hash) public AuthorizedAddress{  
    }  
  
    function get(int ID) public view returns (string memory) {  
    }  
}
```





## Travel:

È il contratto al centro del sistema.

Permette il salvataggio di vari codici hash.

Ogni codice è collegato tramite un mapping ad un intero che nel sistema corrisponde all'ID Hash

```
contract Travel {  
  
    mapping(int => string) travel;  
  
    function add(int ID, string memory travel_hash) public{  
        if(bytes(travel[ID]).length == 0) travel[ID] = travel_hash;  
    }  
  
    function get(int ID) public view returns (string memory){  
        return travel[ID];  
    }  
}
```



## Registry:

Contratto che funge da registro per gli utenti della blockchain

È modificabile solo dall'owner (proprietario del contratto)

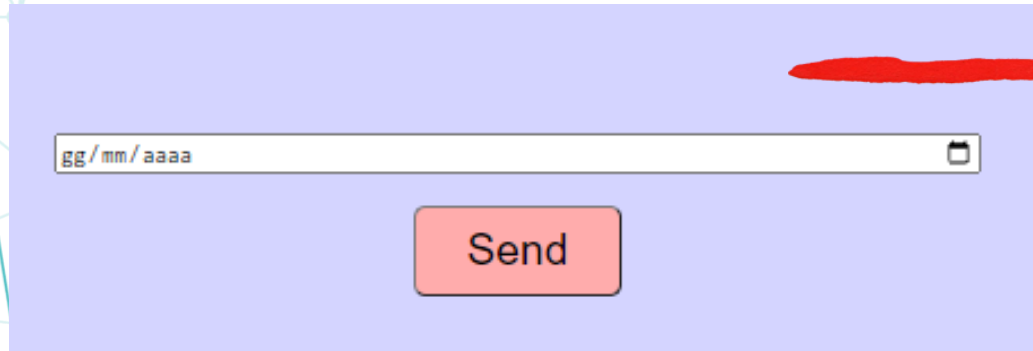
Fornisce la funzione isAuthorized per il controllo di un utente

```
contract BusinessRegistry {  
  
    modifier onlyOwner {  
        require(msg.sender == owner);  
        _;  
    }  
  
    address owner;  
    address[] AuthorizedUser;  
  
    constructor() public{  
        owner = msg.sender;  
    }  
  
    function add(address user) public onlyOwner { ...  
    }  
  
    function get() public view onlyOwner returns (address[] memory) { ...  
    }  
  
    function isAuthorized(address user) public view returns (bool) { ...  
    }  
  
    function remove(address user) public onlyOwner { ...  
    }  
  
    function checkIndex(address user) private view returns(uint){ ...  
    }  
}
```

# Certificazione e validazione

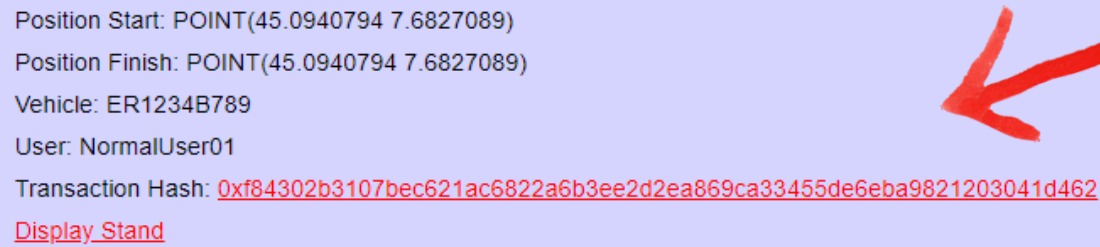
Un utente Admin effettua un'interrogazione

Server:



A user interface for querying the database. It features a light blue background. At the top, there is a text input field containing the placeholder text "gg/mm/aaaa" and a small calendar icon on the right. Below the input field is a red rectangular button with the text "Send" in black. A large red arrow points from the right side of the input field towards the list of steps on the right.

1. Interroga il DB, ottiene tutte le tratte salvate in quella data
2. Per ogni tratta:
  - Interroga il DB per ottenere tutte le tratte con lo stesso ID Hash, e ne calcola l'hash
  - Interroga la blockchain ottenendo per quel determinato ente l'hash corrispondente all'ID Hash
  - Compara i due codici hash



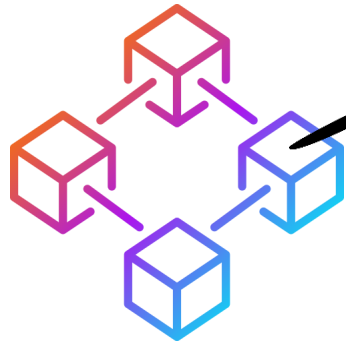
A user interface showing the results of a query. It has a light blue background. The results are displayed in a white box with a rounded top-left corner. The text is as follows:  
Position Start: POINT(45.0940794 7.6827089)  
Position Finish: POINT(45.0940794 7.6827089)  
Vehicle: ER1234B789  
User: NormalUser01  
Transaction Hash: [0xf84302b3107bec621ac6822a6b3ee2d2ea869ca33455de6eba9821203041d462](#)  
[Display\\_Stand](#)  
A large red arrow points from the right side of the results box towards the list of steps on the right.

# Conclusioni

- Costruzione di un prototipo di sistema
  - ➔ Costruzione di una blockchain PoA con BESU (Ethereum Client)
  - ➔ Costruzione di una WebApp SPA
- Testing interno del sistema
  - Salvataggio hash
  - Validazione delle tratte
  - Segnalazione possibili attacchi

# Sviluppi futuri

- Portare tutti i dati all'interno della blockchain privata
- Utilizzare una blockchain pubblica per validare la blockchain privata



Grazie per l'attenzione



**Giorgio Mecca**

**Università degli Studi di Torino**