In this prototype the accounts don't exist. Transaction have UTXOs at outputs, that have a public key and an amount value. The entry of the transaction is a list of UTXOs generated in another transaction, and a sign with its private key. The inputs UTXOs became invalid after the transaction. If an user A want transfer money to an user B, B must generate a public-private key pair and pass the public key to A. With this key, A must generate a transaction with one or more valid UTXOs, whose private key he/she know, and an UTXO as output with the public key received from B. Also, if the amount of the UTXOs used as input is superior to the amount A want to transfer, he/she can generate a public-private key pair and create another UTXO at output with the difference.

It has principally to advantages, the first is the transaction are less tracrable, because in the blockchain there are nothing that link an new UTXO to an user, neither it can be determinate if the "owner" of a new UTXO is the same that the UTXOs used in the input.
The second adventage is there is less metadata for maintein in memory. For the confirmate blocks, only it's necessary have the list of valid UTXOs in this instance.

In other side, it's require a more complicated wallet with it own communication protocol outside the blockchain

CHARACTERISTICS
- It's impossible the fast synchronization, each node must to know the previews blocks to validate a new node (or mining). It was designed in this way to prevent the nodes discard the confirmed blocks, witch will cause new nodes a way to obtain them.
It works defining the proof of work in this way: if there is two blocks consecutive B_n, B_{n+1}, and they hash are H_n and H_{n+1} respective, the node can verify the proof of work in this way:

- Take the first 8 bytes of H_n and use this number mod n to select a block in this branch.
- Take the second 8 bytes of H_n and concatenate it with the block selected
- Calculate the hash of the block plus the bytes. The X first bits of this should be equal to the first X bits of H_n.

It's more difficult create new block that pass the proof of work and that the 8 bytes of hash points to a node the miner want. So the most efficient miner will be the ones that know all the nodes

- A block is a list of transactions and an extra UTXO that can set the minner. If there is a difference between the sum of the values in the input and the output, this difference can be add to the this UTXO. The total value of this is this deference of all transactions in the block plus 5 units.

- For broadcast messages to all nodes, a node send a package with an id (a public key), a counter and the message, that can be a block, a transaction or a quest for a block.
The system verify the counter. If it already received this message it discart. If there are packages between the last received and the actual, it remember it for the future. If the package is the next to the last received it process this and resend it to the other nodes it knows. If in the cache there are packages consecutive to this new packages, it process and resend them also.

The exception is if it receive a quest for a block. In this case, the node don't resend the message, but only respond with the block (always respond with a block of the longest chain).

"TO DO" LIST
- It have not a discovery mechanism. A node only have a list of know IPs that must began with at least one IP and later it's feeding with tha IPs of the nodes that connect with it.
- There is not a mechanism that allow new nodes start later, because there are messages it doesn't know in what position the message counter is.

TEST

- The object Node is a test that set up three nodes and two wallets, and one of this wallets send money to another