TÉCNICO LISBOA

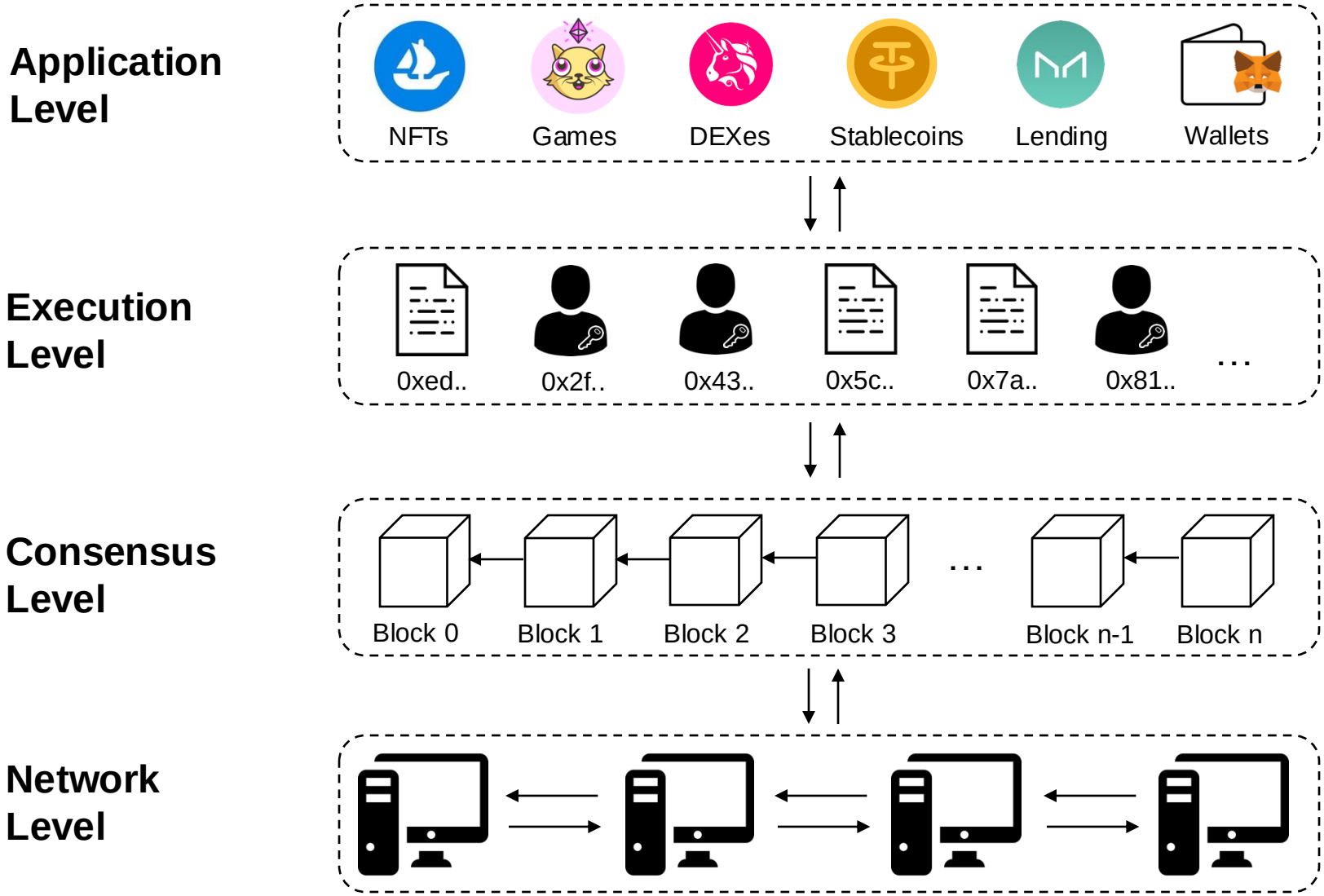
Smart Contracts

Highly Dependable Systems

Blockchains

- Blockchains aim to **decentralize control** over a particular asset by substituting trusted central entities
- **Bitcoin** decentralizes control over **payments** by substituting banks
- **Ethereum** decentralizes control over **computation** by introducing smart contracts

Blockchain Model



Smart Contracts

- First introduced by **Nick Szabo** in 1997
- "A trustless system consisting of self-executing computer programs that facilitates the enforcement of legal contracts"

- Remained out of reach…
- …until Ethereum came around the corner in 2015

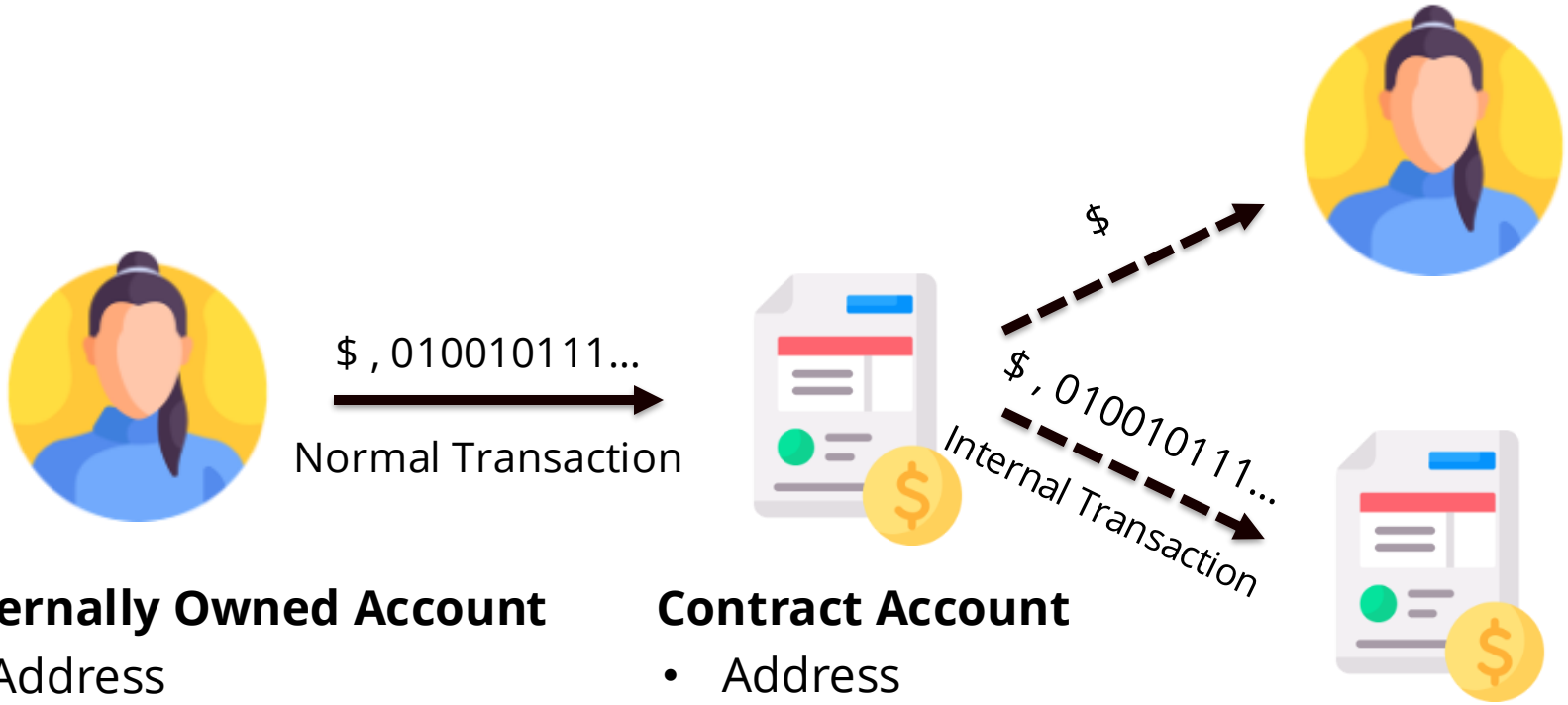
- Smart contracts are **neither smart nor contracts**

- **Programs stored and executed across a blockchain**

Yet Different…

- **Immutable** (i.e., code cannot be changed once deployed)
- You **pay for execution** (i.e., transaction/gas fees)
- No access to **storage outside the blockchain**
- No ability to performs **calls outside the blockchain**

Ethereum Accounts



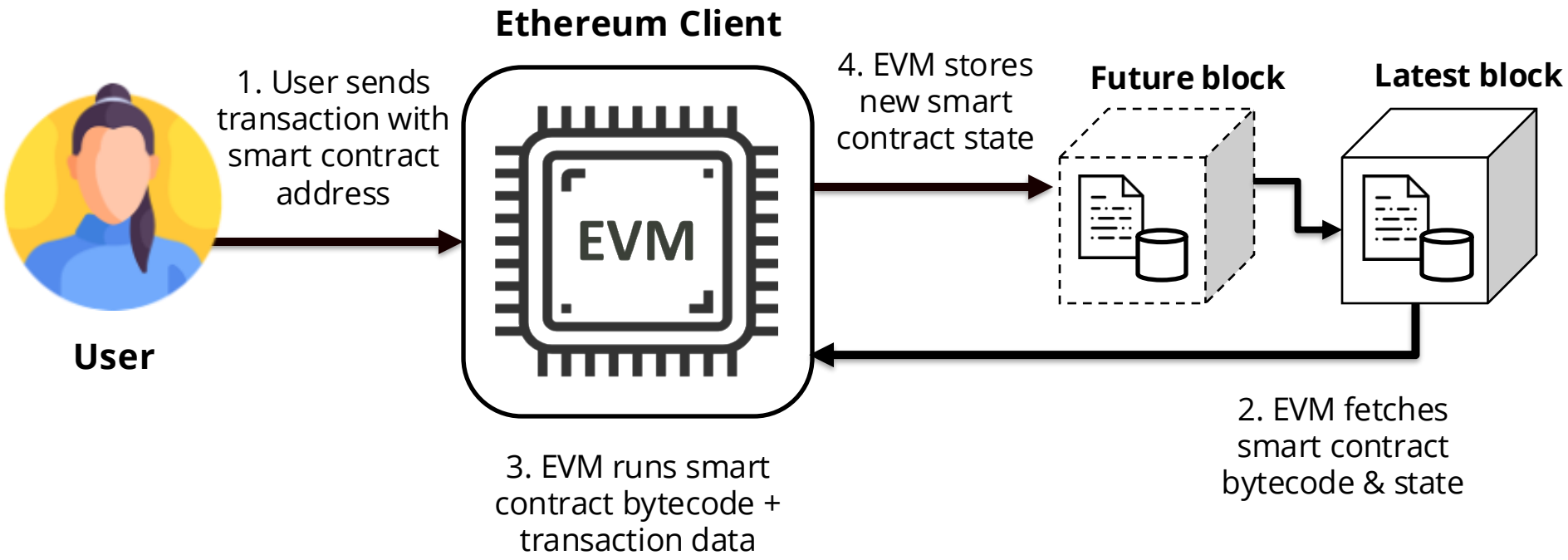
Externally Owned Account

- Address
- Balance

Contract Account

- Address
- Balance
- Code
- Storage

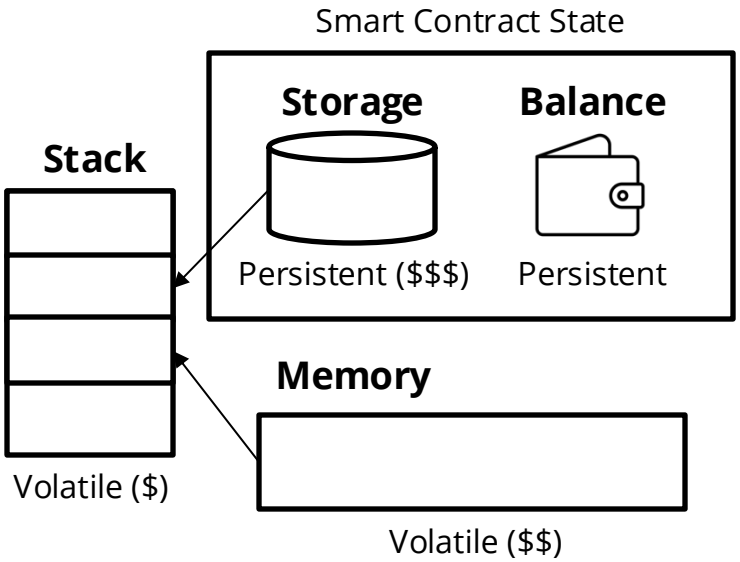
Ethereum Execution Model



Ethereum Virtual Machine

- Turing-complete
- 256-bit stack-based architecture (**no registers**)
- Memory model is composed of **stack**, **memory**, and **storage**
- Termination is guaranteed (**gas mechanism**)

Instruction	Gas Cost
JUMPI	10
PUSH32	3
MSTORE	9
SSTORE	20000
…	…



Solidity

- High-level programming language
- Designed for Ethereum smart contracts
- Similarities with JavaScript / C++
- Statically-typed language (types known at compile time)
- Supports inheritance and interfaces

Deploying Solidity Smart Contracts

```
pragma solidity ^0.5.11;

contract HelloWorld {

  function hello() public pure returns(string memory) {
      return 'Hello World!';
  }
```

Source Code

```
6080604052348015600f
57600080fd5b50600436
106028576000356 0e01c
806319ff1d2114602d575
B600080fd5b603360a…
```

EVM Bytecode

Developer **Solidity Compiler** **Smart Contract**

Solidity Peculiarities

Mappings

Key type

```
mapping ( string => address ) public registry;
```

Value type

- Key value storage / hash table
- Every key is initially mapped to zero
- No built-in way to query the length of a mapping
- Not possible to iterate over current elements

Solidity Peculiarities

Fallback functions

```
    fallback() external payable { x = 1; y = msg.value; }

    // This function is called for plain Ether transfers, i.e.
    // for every call with empty calldata.
    receive() external payable { x = 2; y = msg.value; }
```

- Executed when contract is called, and no functions match the given function signature
- Contract may contain only one fallback function
- Dedicated fallback function named receive for empty calldata

Solidity Peculiarities

Access to native currency

```
function acceptExactlyTwoEther() public payable returns(uint) {
    require(msg.value >= 2.0 ether);

    uint refund = msg.value - 2.0 ether;
    payable(msg.sender).transfer(refund);

    return address(this).balance;
}
```

- Functions receiving native currency need to be labeled with payable
- Supports native currency units (e.g., ether, wei, etc.)
- Native currency can be transferred via call, send, and transfer
- Balance represents native currency owned by the smart contract

Solidity Peculiarities

Selfdestruct

```
function closeContract() public {

        selfdestruct(msg.sender);
}
```

- Smart contracts can be removed via selfdestruct function
- Selfdestruct function transfers contract balance to provided address

Tokens

- Smart contracts that function like digital assets
- Tokenization is the process of converting an asset into a token
- Difference between fungible and non-fungible tokens
  - **Non-fungible**: asset has a distinct ID (unique)
  - **Fungible**: asset is interchangeable
- Using standard interfaces for tokens helps enable interoperability

ERC20 Standard (Fungible)

https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

Basic functionality:

```
function totalSupply() constant returns (uint256 totalSupply)

function balanceOf(address _owner) constant returns (uint256 balance)

function transfer(address _to, uint256 _value) returns (bool success)
```

Delegating control:

```
function transferFrom(address _from, address _to, uint256 _value) returns (bool success)

function approve(address _spender, uint256 _value) returns (bool success)

function allowance(address _owner, address _spender) constant returns (uint256 remaining)
```

ERC20 vs ERC721 vs ERC1155

ERC20

0xabde ⟶ 20 COIN

0xefgh ⟶ 30 COIN

0xhifjk ⟶ 10 COIN

Fungible Tokens

ERC721

Kitty #1 ⟶ 0xabde

Kitty #2 ⟶ 0xefgh

Kitty #3 ⟶ 0xhifjk

Non-Fungible Tokens

ERC1155

Swords ⟶ 0xabde ⟶ 20 SWORD

0xefgh ⟶ 30 SWORD

Shields ⟶ 0xabde ⟶ 5 SHIELD

Fungible & Non-Fungible Tokens