
TECNOLOGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLOGICO CAMPUS VERACRUZ


# RentiPelis


**ALUMNA:**
**VALDES MORALES KARLA LIZBETH E17021563**

**PROFESOR:**
**LOPEZ MENDEZ GENARO**

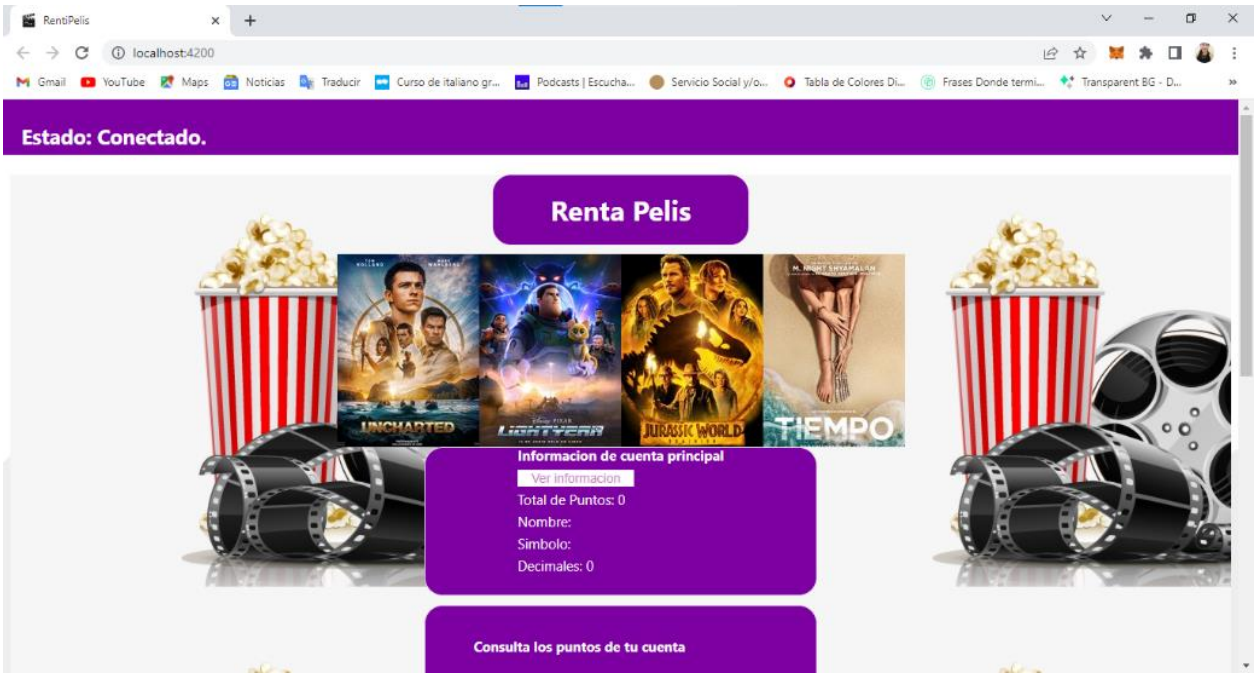

**MATERIA:**
**TRANSACCIONES COMPUTACIONALES CON BLOCKCHAIN**

**FECHA:**
**20/06/2022**

## PROPUESTA:

La intención es desarrollar una aplicación que pueda otorgar a un usuario puntos (tokens) por renta de películas, cada vez que rente una película con dinero real (pesos mexicanos) se le otorgaría 10 tokens así podría ir acumulándolos hasta que eventualmente podrá canjearlos por su valor equivalente en pesos, consiguiendo la renta de una película totalmente gratis, además de que podrían igualmente canjearse para completar el costo de la renta, sin límite de tokens.

## INTERFAZ:




## METODOS

- **Información de cuenta principal:**
  Sirve para consultar los siguientes datos:
  - Total de puntos:
  - Nombre:
  - Símbolo:
  - Decimales:

- **Consulta los puntos de tu cuenta:**
  Se pueden consultar los puntos (tokens) en una cuenta, para eso se usa la función balanceOf () que proporciona el número de tokens que posee una dirección determinada.
  Se pueden verificar los puntos de cualquier dirección.

```
  function balanceOf(address tokenOwner) public override view returns (uint
balance) {
        return balances[tokenOwner];
    }
```

- **Transferencia de puntos:**
  Con esta función podrás transferir tokens de la cuenta principal a otra, para esto se usa la función transfer () que puede transferir algunos tokens directamente del remitente del mensaje a cualquier otra dirección.

```
    function transfer(address receiver, uint tokens) public override returns
(bool success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        balances[receiver] = safeAdd(balances[receiver], tokens);
        emit Transfer(msg.sender, receiver, tokens);
        return true;
    }
```

- **Transferir puntos a otras cuentas:**
  La función transferFrom es una alternativa cómoda a transfer que permite un poco más de programabilidad en las aplicaciones descentralizadas. Igual que transfer, se emplea para mover tokens, pero éstos no han de pertenecer necesariamente a la persona que llama al contrato, por lo tanto, puedes autorizar a alguien – o a otro contrato – para que transfiera fondos en tu nombre.

```
  function transferFrom(address sender, address receiver, uint tokens)
public override returns (bool success) {
        balances[sender] = safeSub(balances[sender], tokens);
        allowed[sender][msg.sender] = safeSub(allowed[sender][msg.sender],
tokens);
        balances[receiver] = safeAdd(balances[receiver], tokens);
        emit Transfer(sender, receiver, tokens);
        return true;
    }
```

- **Canjear puntos en mi compra:**

Esta función devolverá a la cuenta principal los tokens que el usuario desea intercambiar por un descuento o producto (película rentada) al equivalente de los puntos en dinero real al total de realizar su compra.

**ADDRESS DEL CONTRATO:**
0xb9492086d3db6768a398e0eb87dd653ad6c02817

**CUENTAS PRUEBA:**

**CUENTA 1**
0x4D179C8808466ED88A70F790A23A9031C1bF6D80

**CUENTA 2**
0x0CFC81A413d01f8b4c033B3c70fBbCf32CC67849

## CONTRATO:

```
// SPDX-License-Identifier: unlicensed
pragma solidity 0.8.4;
// ----------------------------------------------------------------------------
---
// Safe maths
// ----------------------------------------------------------------------------
---
contract SafeMath {
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
}
// ----------------------------------------------------------------------------
---
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
// ----------------------------------------------------------------------------
---
abstract contract ERC20Interface {
    function totalSupply() virtual public view returns (uint);
```

```
    function balanceOf(address tokenOwner) virtual public view returns (uint
balance);
    function allowance(address tokenOwner, address spender) virtual public
view returns (uint remaining);
    function transfer(address to, uint tokens) virtual public returns (bool
success);
    function approve(address spender, uint tokens) virtual public returns
(bool success);
    function transferFrom(address from, address to, uint tokens) virtual
public returns (bool success);
    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint
tokens);
}
// ----------------------------------------------------------------------------
---
// ERC20 Token, with the addition of symbol, name and decimals
// assisted token transfers
// ----------------------------------------------------------------------------
---
contract KarlaV is ERC20Interface, SafeMath {
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;
    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;
    // ------------------------------------------------------------------------
---
    // Constructor
    // ------------------------------------------------------------------------
---
    constructor() {
        symbol = "KLVM";
        name = "KLVMTOKEN";
        decimals = 0;
        _totalSupply = 1000000;
        balances[msg.sender] = _totalSupply;
        emit Transfer(address(0), msg.sender, _totalSupply);
    }
    // ------------------------------------------------------------------------
---
    // Total supply
    // ------------------------------------------------------------------------
---
```

```
    function totalSupply() public override view returns (uint) {
        return _totalSupply - balances[address(0)];
    }
    // ------------------------------------------------------------------------
---
    // Get the token balance for account tokenOwner
    // ------------------------------------------------------------------------
---
    function balanceOf(address tokenOwner) public override view returns
(uint balance) {
        return balances[tokenOwner];
    }
    // ------------------------------------------------------------------------
---
    // Transfer the balance from token owner's account to receiver account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
---
    function transfer(address receiver, uint tokens) public override returns
(bool success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        balances[receiver] = safeAdd(balances[receiver], tokens);
        emit Transfer(msg.sender, receiver, tokens);
        return true;
    }
    // ------------------------------------------------------------------------
---
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
    // recommends that there are no checks for the approval double-spend
attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
---
    function approve(address spender, uint tokens) public override returns
(bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }
    // ------------------------------------------------------------------------
---
```

```
    // Transfer tokens from sender account to receiver account
    //
    // The calling account must already have sufficient tokens approve(...)-
d
    // for spending from sender account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // 0 value transfers are allowed
    // ------------------------------------------------------------------------
---
    function transferFrom(address sender, address receiver, uint tokens)
public override returns (bool success) {
        balances[sender] = safeSub(balances[sender], tokens);
        allowed[sender][msg.sender] = safeSub(allowed[sender][msg.sender],
tokens);
        balances[receiver] = safeAdd(balances[receiver], tokens);
        emit Transfer(sender, receiver, tokens);
        return true;
    }
    // ------------------------------------------------------------------------
---
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
---
    function allowance(address tokenOwner, address spender) public override
view returns (uint remaining) {
        return allowed[tokenOwner][spender];
    }
}
```

## PRUEBAS Y VISTA DE METODOS: