



**TECNOLOGICO NACIONAL DE MÉXICO**  
**INSTITUTO TECNOLÓGICO DE VERACRUZ**



# **JBJ TOKEN**

**ALUMNO**  
**GONZALEZ CASTRO BRYAN JOSIMAR E17021607**

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


**MATERIA:**  
**TRANSACCIONES COMPUTACIONALES**  
**CON BLOCKCHAIN 3:00-4:00PM**

**GRUPO: 9J8B**

## PROPUESTA:

La intención de esta dapp es desarrollar una aplicación que pueda otorgar a un usuario puntos(tokens) en el sitio de venta de artículos JosShop, cada vez que ingrese al sitio se le otorgaría un token por la visita al sitio web e ir acumulando progresivamente más tokens que eventualmente podrá intercambiar por su valor equivalente en pesos por la cantidad de puntos acumulados y así obtener un descuento en el total de su compra.

## INTERFAZ

 Estado: Conectado.

### Tiendas JosShop

#### Información de cuenta principal

Ver información

TotalSupply: 0  
Nombre:  
Símbolo:  
Decimales: 0

#### Consulta los puntos de tu cuenta

Cuenta:  Consultar

Puntos Totales:

#### Transferencia de puntos

Cuenta Destino:

Cantidad de puntos:  Enviar puntos

#### Tranferir puntos a otras cuentas:

#### Tranferir puntos a otras cuentas:

Cuenta Origen:

Cuenta Destino:

Cantidad de puntos:  Transferir

#### Aprobacion de puntos:

Cuenta a aprobar:

Cantidad:  Aprobar

#### Puntos aprobados:

Propietario:

Cuenta Aprobada:  Consultar Aprobación

Puntos aprobados: 0

#### Gastar puntos en mi compra

Cantidad de puntos a gastar:  Gastar

Transaction Hash Block number From

## METODOS

- **Información de cuenta principal:** sirve para consultar los siguientes datos:

Nombre.

Símbolo.

Posiciones decimales que determinan la divisibilidad del token.

Cantidad total de tokens disponibles.

- **Consulta los puntos de tu cuenta:** consultar puntos(tokens) en una cuenta, usa la función `balanceOf()` proporciona el número de tokens que posee una dirección determinada. Cualquiera puede verificar el saldo de cualquier dirección, al igual que todos los datos son públicos en la cadena de bloques.

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

- **Transferencia de puntos:** poder transferir tokens JBJ de la cuenta principal a otra, usa la función `transfer()` que puede transferir algunos tokens directamente del remitente del mensaje a 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.

En otras palabras, puedes autorizar a alguien –o a otro contrato– para que transfiera fondos en tu nombre. Un caso de uso potencial involucra el pago por servicios basados en suscripciones, cuando no deseas realizar manualmente dicho pago cada día/semana/mes. En su lugar, simplemente permites que el programa se ocupe de ello. Esta función activa el mismo evento que `transfer`.

```
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;
}
```

- **Aprobación de puntos:** Con dicha función, puedes limitar el número de tokens que un smart contract puede retirar de tu balance. Sin ella, corres el riesgo de un mal funcionamiento del contrato (o de que sea explotado) y robe todos tus fondos.

```
function approve(address spender, uint tokens) public override returns
(bool success) {
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    return true;
}
```

- **Puntos aprobados:** La función `allowance()` proporciona el número de tokens permitidos para ser extraídos de una dirección dada a otra dirección dada. Cualquiera puede verificar el saldo de cualquier dirección, así como toda la información es pública en la cadena de bloques. Cuando le has otorgado permiso a un contrato para que gestione tus tokens, puedes utilizar esta función para comprobar cuántos de ellos puede retirar todavía.

```
function allowance(address tokenOwner, address spender) public override  
view returns (uint remaining) {  
    return allowed[tokenOwner][spender];  
}
```

- **Gastar puntos en mi compra:** devolverá a la cuenta principal los tokens que el usuario desea intercambiar por un descuento al equivalente de los puntos en dinero real al total de realizar su compra.

**ADDRESS DEL CONTRATO:**

0x77120391be854260940c38d28f55c07Df4DbF1c4

**INFURA ID:**

wss://ropsten.infura.io/ws/v3/344840a50e9a410f9d2c3182b85b77c5

**CUENTAS PRUEBA:**

**CUENTA 1**

0x80A5B39B1e87E4bf20706D61Cb42fE90e3d15E58

**CUENTA 2**

0x0CFC81A413d01f8b4c033B3c70fBbCf32CC67849

**CUENTA 3**

0x9dEE164c87A696221f07F5aE8B0529A6cc5eB33E

## 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 JBJToken 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 = "JBJ";
        name = "JBJToken";
        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:

### Informacion de cuenta principal

[Ver informacion](#)

TotalSupply: 1000000

Nombre: JBJToken

Simbolo: JBJ

Decimales: 0

### Consulta los puntos de tu cuenta

Cuenta:  [Consultar](#)

Puntos Totales: 999994

Envío de puntos a la cuenta 2, primero veremos cuantos puntos tiene.

### Consulta los puntos de tu cuenta

Cuenta:  [Consultar](#)

Puntos Totales: 0

### Transferencia de puntos

Cuenta Destino:

Cantidad de puntos:  [Enviar puntos](#)

### Consulta los puntos de tu cuenta

Cuenta:  [Consultar](#)

Puntos Totales: 5

Envió de puntos de la cuenta 2 a la 3, primero veremos cuantos puntos tiene la cuenta 3.

#### Consulta los puntos de tu cuenta

Cuenta: 0x9dEE164c87A696221f07 [Consultar](#)

Puntos Totales: 6

#### Tranferir puntos a otras cuentas:

Cuenta Origen: 0x0CFC81A413d01f8b4c03

Cuenta Destino: 0x9dEE164c87A696221f07

Cantidad de puntos: 4 [Transferir](#)

Consultaremos los puntos de la cuenta 2 y 3

Cuenta 2

#### Consulta los puntos de tu cuenta

Cuenta: 0x0CFC81A413d01f8b4c03 [Consultar](#)

Puntos Totales: 1

Cuenta 3

#### Consulta los puntos de tu cuenta

Cuenta: 0x9dEE164c87A696221f07 [Consultar](#)

Puntos Totales: 10

## Aprobación de puntos a la cuenta 2 por 100 tokens

Cuenta Destino: 0x9dEE164c87A09622107  
Cantidad de puntos: 4 [Transferir](#)  
Transacción exitosa

**Aprobacion de puntos:**

Cuenta a aprobar: 0x0CFC81A413d01f8b4c03  
Cantidad: 100 [Aprobar](#)

**Puntos aprobados:**

Propietario:   
Cuenta Aprobada: [Consultar Aprobación](#)  
Puntos aprobados: 0

Give permission to access your JBJ?

By granting permission, you are allowing the following cuenta to access your funds

0x0cfc...7849 [Editar permiso](#)

**Cuota de transacción** [Editar](#)

Esta solicitud tiene asociada una cuota. **\$0.21**  
0.000068 ETH

[View full transaction details](#)

[Rechazar](#) [Confirmar](#)

### Aprobacion de puntos:

Cuenta a aprobar: 0x0CFC81A413d01f8b4c03

Cantidad: 100

[Aprobar](#)

Aprobacion exitosa

### Puntos aprobados:

Propietario: 0x80A5B39B1e87E4bf207C

Cuenta Aprobada: 0x0CFC81A413d01f8b4c03

[Consultar Aprobación](#)

Puntos aprobados: 100

### Gastar puntos de la cuenta 3

#### Consulta los puntos de tu cuenta

Cuenta: 0x9dEE164c87A696221f07 [Consultar](#)

Puntos Totales: 10

#### Gastar puntos en mi compra

Cantidad de puntos a gastar:  [Gastar](#)

#### Consulta los puntos de tu cuenta

Cuenta: 0x9dEE164c87A696221f07 [Consultar](#)

Puntos Totales: 5