```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract PagoJustoCaficultores {
    address public comprador;
    address public productor;
    uint public precioPorSaco;
    uint public cantidadEntregada;
    bool public entregaVerificada;

    constructor(address _productor, uint _precioPorSaco) {
        comprador = msg.sender; // Contrato creado por el comprador
        productor = _productor;
        precioPorSaco = _precioPorSaco;
    }

    function registrarEntrega(uint _cantidad) public {
        require(msg.sender == productor, "Solo el productor puede registrar
la entrega.");
        cantidadEntregada = _cantidad;
        entregaVerificada = false;
    }

    function verificarEntrega() public {
        require(msg.sender == comprador, "Solo el comprador puede verificar
la entrega.");
        require(cantidadEntregada > 0, "No hay entrega registrada.");
        entregaVerificada = true;
    }

    function liberarPago() public payable {
        require(entregaVerificada, "La entrega no ha sido verificada.");
        require(msg.value == cantidadEntregada * precioPorSaco, "El monto
del pago no es correcto.");
        payable(productor).transfer(msg.value);
    }
}
```

Con este contrato inteligente se garantiza el pago gusto a los caficultores a través de una blockchain asegura que los productores reciben el pago justo después de registrar y verificar la entrega.asegura que los productores reciben el pago justo después de registrar y verificar la entrega

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract TrazabilidadCafe {
    struct Etapa {
        string descripcion;
        string ubicacion;
        uint timestamp;
    }

    address public productor;
    Etapa[] public etapas;

    constructor() {
        productor = msg.sender; // Contrato creado por el productor
    }

    function agregarEtapa(string memory _descripcion, string memory
_ubicacion) public {
        require(msg.sender == productor, "Solo el productor puede agregar
etapas.");
        Etapa memory nuevaEtapa = Etapa({
            descripcion: _descripcion,
            ubicacion: _ubicacion,
            timestamp: block.timestamp
        });
        etapas.push(nuevaEtapa);
    }

    function obtenerEtapas() public view returns (Etapa[] memory) {
        return etapas;
    }
}
```

Con este contrato inteligente se garantiza la trazabilidad atravesé de una blockchain aumentando la seguridad e inmutabilidad proporciona la trazabilidad completa, que puede ser consultada por los compradores antes de liberar el pago.