Ejercicio 5: Sistema de Licitaciones Públicas en Ethereum

Tecnologías de Registro Distribuido y Blockchain (BC)

Integrantes:

Jesús Alfredo González Salcedo
Manuel Martín Louredo Pérez

Octubre 2025

Índice

1. Introducción

Este documento presenta un contrato inteligente diseñado para administrar procesos de licitación pública de manera transparente, verificable y resistente a manipulaciones. El sistema permite publicar licitaciones, recibir ofertas, evaluarlas bajo criterios objetivos y seleccionar automáticamente al ganador en base a una métrica ponderada de precio y calidad. El uso de blockchain asegura que cada fase sea trazable, auditable e inmutable.

2. Análisis del Escenario

2.1. Motivación y problema

Las licitaciones públicas tradicionales presentan diversos problemas:

- **Falta de transparencia**: Los procesos de evaluación no son públicamente verificables.
- **Posibilidad de manipulación**: Las ofertas pueden ser alteradas antes del cierre.
- **Corrupción**: Favoritismos y adjudicaciones no objetivas
- **Desconfianza ciudadana**: Imposibilidad de auditar el proceso completo.
- **Costes administrativos**: Burocracia; lenta y costosa.

2.2. Por qué blockchain

El uso de una blockchain pública como Ethereum elimina estos problemas al ofrecer:

- **Inmutabilidad**: Una vez publicada la oferta, no se puede alterar.
- **Trazabilidad**: Cada fase del proyecto queda registrada con timestamps verificables.
- **Transparencia**: Todo el proceso es públicamente auditable.
- **Automatización**: Las reglas de evaluación y adjudicación son ejecutadas por el contrato sin intermediarios.
- **Descentralización**: No depende de una entidad central.

Por tanto la elección de blockchain pública es clave, ya que el objetivo es garantizar verificabilidad independiente y descentralización del proceso.

3. Diseño

3.1. Actores del sistema

- **Administrador (owner)**: Crea licitaciones, cierra el periodo de ofertas, marca la evaluación como completada y calcula el ganador.
- **Evaluadores**: Asignan puntaciones de calidad a las ofertas admitidas.
- **Proveedores**: Envían ofertas con precio y documentación.
- **Ciudadanos/Auditores**: Consultan el estado del proceso y los resultados finales.

3.2. Caso de uso principal (flujo)

1. El administrador publica una licitación con criterios de evaluación.
2. Los proveedores envían sus ofertas (con precio y documentación) antes de la fecha límite.
3. Al vencer el plazo, el administrador cierra el periodo de recepción de ofertas.
4. Los evaluadores puntúan la calidad técnica de cada oferta.
5. El administrador confirma que todas las ofertas están evaluadas.
6. Acto seguido, ejecuta el algoritmo de selección del ganador.
7. Por último, el resultado queda público y permanente en la blockchain. Cualquiera puede ver el resultado.

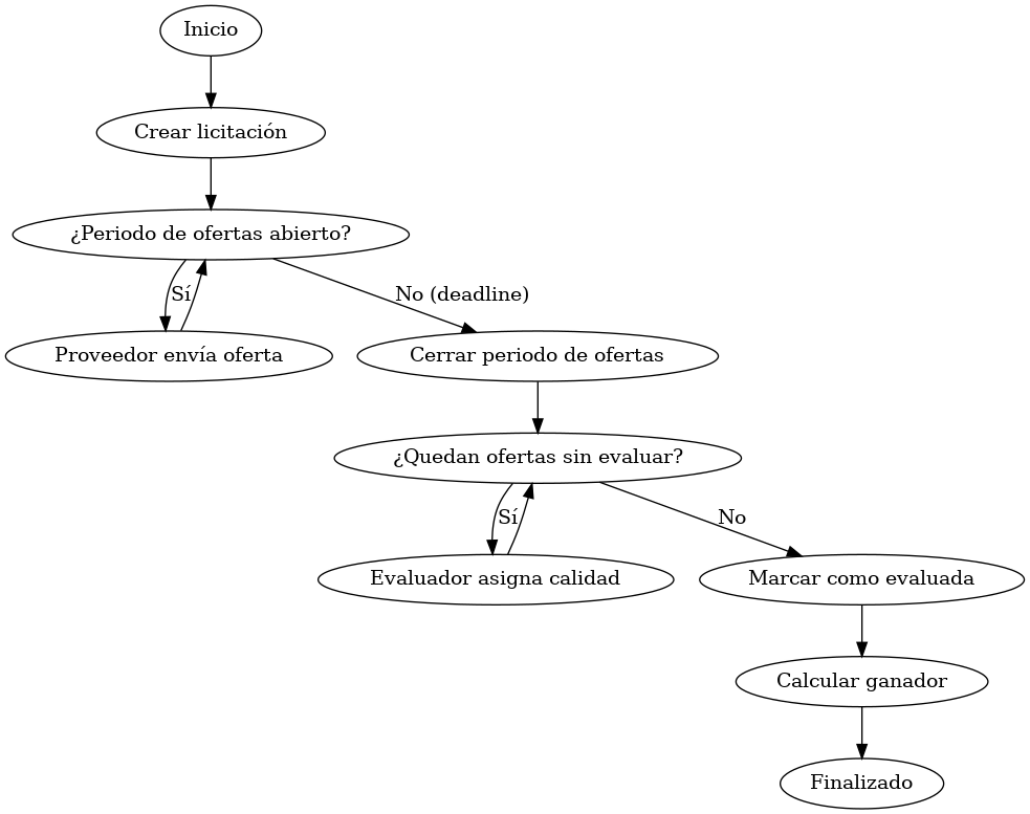

Figura 1: Diagrama de flujo del proceso de licitación pública

3.3. Contenido del Smart Contract

3.3.1. Estructuras de datos

Tender (Licitación)

Listing 1: Estructura de datos Tender

```
struct Tender {
    address creator;       //Administrador que creo la licitacion
    string description;    //Descripcion del proyecto
    uint256 maxPrice;      //Precio maximo aceptable
    uint256 deadline;      //Timestamp limite para ofertas
    uint8 weightPrice;     //Peso del precio en evaluacion (0-100)
    uint8 weightQuality;   //Peso de la calidad en evaluacion (0-100)
    Status status;         //Estado actual de la licitacion
    address winner;        //Direccion del ganador
}
```

Offer (Oferta)

Listing 2: Estructura de datos Offer

```
struct Offer {
    address provider;        // Direccion del proveedor
    uint256 price;           // Precio ofertado
    string documentation;    // Hash IPFS de documentacion tecnica
    uint8 qualityScore;      // Puntuacion de calidad (0-100)
    bool evaluated;          // Si ha sido evaluada
    bool exist;              // Si la oferta existe
}
```

Status (Estados)

Listing 3: Estructura de datos Status

```
enum Status {
    Open,        // Aceptando ofertas
    Closed,      // Cerrada, en evaluacion
    Evaluated,   // Todas las ofertas evaluadas
    Finalized    // Ganador calculado
}
```

3.3.2. Mappings y variables

Listing 4: Mappings y variables

```
// ID --> Licitacion
mapping(uint256 => Tender) public tenders;
// ID --> Proveedor --> Oferta
mapping(uint256 => mapping(address => Offer)) public offers;
// ID --> Lista participantes
mapping(uint256 => address[]) public participants;
// Evaluadores autorizados
mapping(address => bool) public evaluators;
// Contador de licitaciones
uint256 public tenderCount;
```

3.3.3. Funciones principales

Gestión de evaluadores

- `addEvaluator(address)` : Añadir evaluador autorizado.
- `removeEvaluator(address)` : Remover evaluador.
- `isEvaluator(address)` : Verificar si es evaluador.

Gestión de licitaciones

- `addTender(...)` : Crear una nueva licitación.
- `closeOfferPeriod(uint256)` : Cerrar periodo de ofertas.
- `markAsEvaluated(uint256)` : Marcar como lista (ready) para realizar cálculo.

Gestión de ofertas

- `submitOffer(uint256, uint256, string)` : Enviar oferta.
- `evaluateOffer(uint256, address, uint8)` : Evaluar calidad.
- `calculateWinner(uint256)` : Calcular y asignar el ganador.

Consultas

- `getTenderInfo(uint256)` : Devuelve información completa de la licitación.
- `getOffers(uint256)` : Devuelve todas las ofertas con puntaciones.
- `getOffer(uint256, address)` : Devuelve una oferta específica.
- `getParticipants(uint256)` : Devuelve la lista de participantes.

3.3.4. Modificadores

Listing 5: Modificadores presentes en el contrato

```
onlyOwner          // Solo el administrador del contrato
onlyEvaluator      // Solo evaluadores autorizados
```

3.4. Reglas y restricciones principales

- Solo se admiten ofertas mientras la licitación esté abierta y antes del deadline.
- Una misma dirección no puede enviar varias ofertas a la misma licitación.
- La calidad solo puede ser evaluada por cuentas autorizadas (onlyEvaluator)
- No se puede calcular el ganador hasta que todas las ofertas han sido evaluadas.

4. Implementación

El contrato se implementó en Solidity empleando `Ownable` de OpenZppelin para la gestión de privilegios. Todas las funciones críticas incluyen validaciones mediante `require` para asegurar la consistencia del contrato en cada transición.

El cálculo del ganador se realiza mediante una combinación ponderada de dos factores:

- **PrecioScore** = min(100, (maxPrice * 100 / precioOferta)
- **FinalScore** = (PrecioScore * pesoPrecio + Calidad * pesoCalidad) / 100

Es importante decir que no se maneja dinero dentro del contrato en esta versión, por lo tanto, no es vulnerable a reentracy. La seguridad se centra en control de acceso, validación de estado y consistencia de datos.

5. Pruebas

Las pruebas se realizaron con la ayuda de la herramienta Remix IDE:

- Despliegue y comprobación de `owner`.
- Creación de una licitación y lectura de su estado inicial.
- Envío de ofertas desde distintas cuentas para la misma dirección.
- Cierre del periodo de ofertas una vez superado el deadline.
- Evaluación de todas las ofertas por evaluadores autorizados.
- Confirmación de que todas las ofertas han sido evaluadas (`markAsEvaluated`)
- Cálculo automático del ganador (`calculateWinner`).

Además de pruebas anteriores realizadas en Remix, se desarrolló una interfaz web mínima utilizando Web3.js y MetaMask para interactuar con el contrato desde el navegador. Esta interfaz permitió comprobar el funcionamiento del flujo sin depender de Remix, simulando un entorno más cercano a un uso real.

6. Líneas futuras

Entre las posibles mejoras previstas:

- Ocultación de ofertas hasta el cierre mediante esquema commit-reveal.
- Inclusión de depósitos económicos para evitar ofertas no serias.
- Migración a capa 2 (Polygon, Optimism) para reducir costes de gas.
- Extensión de criterios de evaluación mediante oráculos externos.

7. Conclusión

El sistema de licitaciones desarrollado mediante smart contracts en Ethereum proporciona una solución robusta a los problemas de transparencia, verificación y manipulación. El contrato representa un caso de uso real y aplicable de blockchain pública, demostrando cómo la tecnología puede mejorar ciertos procesos gubernamentales críticos manteniendo los principios fundamentales de la ciberseguridad: confidencialidad, integridad y disponibilidad.