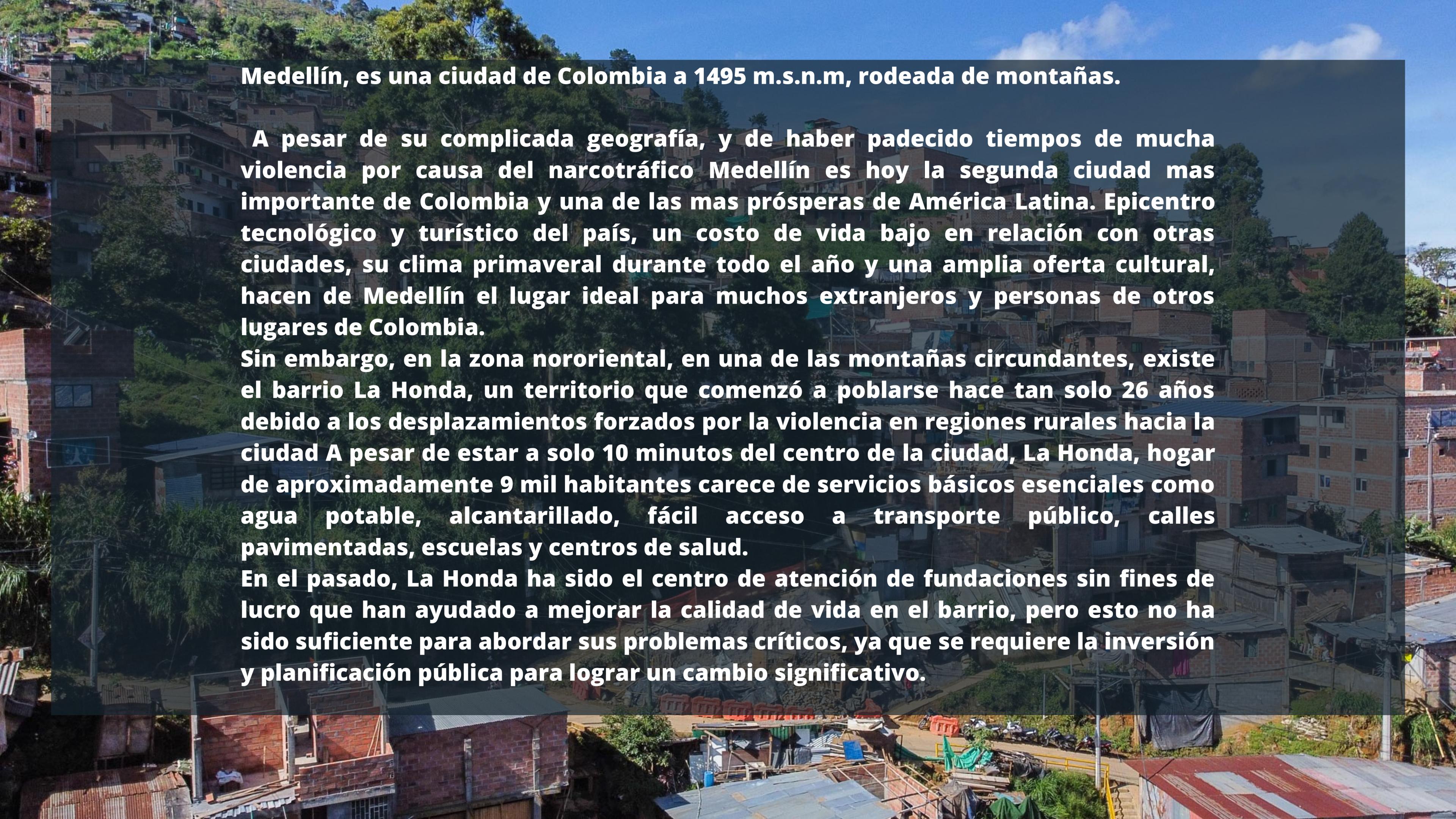


PROYECTO SOLIDITY

BARRIO LA HONDA: CAMBIANDO EL RUMBO DE SU HISTORIA CON LA BLOCKCHAIN

VICTORIA CODREANU
JENNIFER GIRALDO HERNÁNDEZ
VALESKA RIOS ZEGARRA

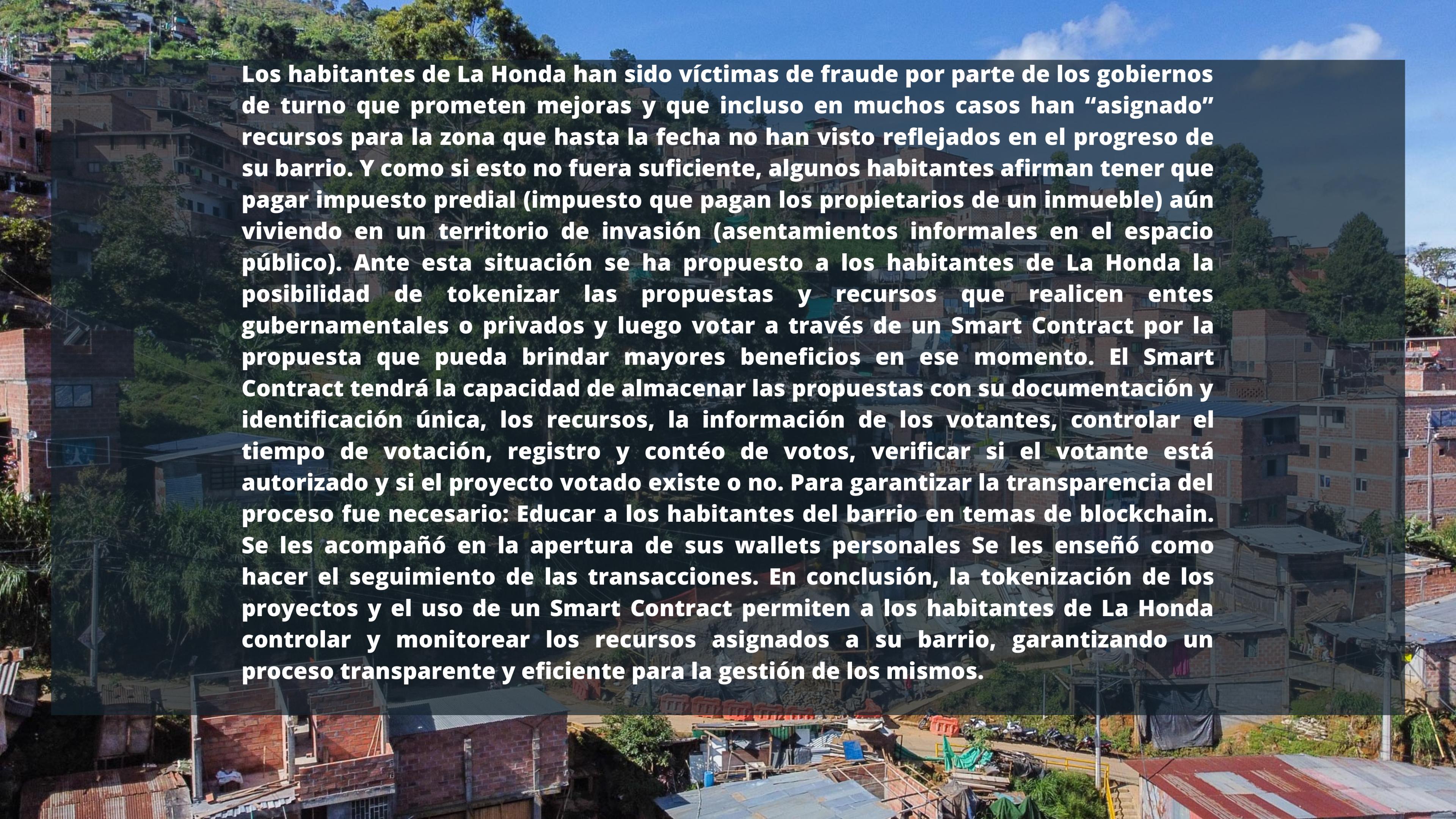


Medellín, es una ciudad de Colombia a 1495 m.s.n.m, rodeada de montañas.

A pesar de su complicada geografía, y de haber padecido tiempos de mucha violencia por causa del narcotráfico Medellín es hoy la segunda ciudad mas importante de Colombia y una de las mas prósperas de América Latina. Epicentro tecnológico y turístico del país, un costo de vida bajo en relación con otras ciudades, su clima primaveral durante todo el año y una amplia oferta cultural, hacen de Medellín el lugar ideal para muchos extranjeros y personas de otros lugares de Colombia.

Sin embargo, en la zona nororiental, en una de las montañas circundantes, existe el barrio La Honda, un territorio que comenzó a poblararse hace tan solo 26 años debido a los desplazamientos forzados por la violencia en regiones rurales hacia la ciudad A pesar de estar a solo 10 minutos del centro de la ciudad, La Honda, hogar de aproximadamente 9 mil habitantes carece de servicios básicos esenciales como agua potable, alcantarillado, fácil acceso a transporte público, calles pavimentadas, escuelas y centros de salud.

En el pasado, La Honda ha sido el centro de atención de fundaciones sin fines de lucro que han ayudado a mejorar la calidad de vida en el barrio, pero esto no ha sido suficiente para abordar sus problemas críticos, ya que se requiere la inversión y planificación pública para lograr un cambio significativo.



Los habitantes de La Honda han sido víctimas de fraude por parte de los gobiernos de turno que prometen mejoras y que incluso en muchos casos han "asignado" recursos para la zona que hasta la fecha no han visto reflejados en el progreso de su barrio. Y como si esto no fuera suficiente, algunos habitantes afirman tener que pagar impuesto predial (impuesto que pagan los propietarios de un inmueble) aún viviendo en un territorio de invasión (asentamientos informales en el espacio público). Ante esta situación se ha propuesto a los habitantes de La Honda la posibilidad de tokenizar las propuestas y recursos que realicen entes gubernamentales o privados y luego votar a través de un Smart Contract por la propuesta que pueda brindar mayores beneficios en ese momento. El Smart Contract tendrá la capacidad de almacenar las propuestas con su documentación y identificación única, los recursos, la información de los votantes, controlar el tiempo de votación, registro y conteo de votos, verificar si el votante está autorizado y si el proyecto votado existe o no. Para garantizar la transparencia del proceso fue necesario: Educar a los habitantes del barrio en temas de blockchain. Se les acompañó en la apertura de sus wallets personales Se les enseñó como hacer el seguimiento de las transacciones. En conclusión, la tokenización de los proyectos y el uso de un Smart Contract permiten a los habitantes de La Honda controlar y monitorear los recursos asignados a su barrio, garantizando un proceso transparente y eficiente para la gestión de los mismos.

EL REPOSITORIO GITHUB CONTIENE

- Presentación del proyecto
- El contrato inteligente
- Manual de usuario
- PDF entregable

```

5  contract Voting {
6      // Estructura para almacenar los recursos
7      struct Resource {
8          string name;
9          uint votes;
10     }
11 }
12
13     // Arreglo de recursos
14     Resource[] public resources;
15
16     // Diccionario para almacenar los votantes autorizados
17     mapping(address => bool) public authorizedVoters;
18
19     // Tiempo límite de la votación obteniendo el momento actual en segundos desde el epoch time (01/01/1970) + 5 días que durara la validez del contrato
20     uint public votingDeadline = block.timestamp + 5 days;
21
22     // Propietario del contrato
23     address public owner;
24
25     // Constructor
26     constructor() {
27         owner = msg.sender;
28     }
29
30     function authorizeVoter(address voter) public {
31         require(msg.sender == owner, "Solo el dueño del contrato puede autorizar a los votantes");
32     }
33
34     function registerResource(string memory name, uint id) public {
35         require(msg.sender == owner, "Solo el dueño del contrato puede registrar recursos");
36     }
37
38     function vote(uint resourceId) public {
39         require(authorizedVoters[msg.sender], "El votante no está autorizado");
40         require(block.timestamp <= votingDeadline, "La votación ha finalizado");
41
42         // Verificar si el recurso existe
43         uint resourceIndex;
44         for (uint i = 0; i < resources.length; i++) {
45             if (resources[i].name == resourceId) {
46                 resourceIndex = i;
47                 break;
48             }
49         }
50
51         if (resourceIndex != 0) {
52             resources[resourceIndex].votes++;
53         } else {
54             Resource memory newResource = Resource(name);
55             newResource.votes++;
56             resources.push(newResource);
57         }
58     }
59
60     function getWinner() public {
61         uint maxVotes = 0;
62         uint winningIndex = 0;
63
64         for (uint i = 0; i < resources.length; i++) {
65             if (resources[i].votes > maxVotes) {
66                 maxVotes = resources[i].votes;
67                 winningIndex = i;
68             }
69         }
70
71         return resources[winningIndex].name;
72     }

```

Este código es un contrato de Ethereum escrito en lenguaje de programación Solidity.

El contrato es llamado "Voting" y está diseñado para llevar a cabo una votación. Hay varios componentes principales en este contrato:

Estructura "Resource": esta estructura se utiliza para almacenar información sobre los recursos que se están votando. Incluye el nombre del recurso, su identificación única y el número de votos que ha recibido.

Arreglo "resources": este es un arreglo público que se utiliza para almacenar los recursos. Cada elemento en el arreglo es una instancia de la estructura "Resource".

Diccionario "authorizedVoters": este es un mapeo público que se utiliza para almacenar información sobre los votantes autorizados. La clave es la dirección de Ethereum de un votante y el valor es un valor booleano que indica si están autorizados o no.

```

22     // Propietario del contrato
23     address public owner;

```

Variable "votingDeadline": esta es una variable pública que se utiliza para almacenar el tiempo límite de la votación. Se establece en el momento actual en segundos desde el epoch time (01/01/1970) más 5 días.

Variable "owner": esta es una variable pública que se utiliza para almacenar la dirección Ethereum del propietario del contrato. El propietario del contrato es la dirección que lo desplegó y se almacena en la variable "owner".

Constructor: esta función es el constructor del contrato y se ejecuta cuando se crea el contrato. Establece la dirección Ethereum del remitente del mensaje como el propietario del contrato.

Función "authorizeVoter": esta función se utiliza para autorizar a los votantes. Solo el propietario del contrato puede ejecutar esta función.

Función "registerResource": esta función se utiliza para registrar un nuevo recurso para la votación. Solo el propietario del contrato puede ejecutar esta función.

Función "vote": esta función se utiliza para votar por un recurso específico. Verifica si el votante está autorizado y si la votación todavía está en curso antes de aumentar el número de votos para el recurso específico.

Función "getWinner": esta función se utiliza para obtener el recurso ganador de la votación. Recorre el arreglo de recursos y encuentra el recurso con el número más alto de votos.

```

44     require(authorizedVoters[msg.sender], "El votante no está autorizado");
45     require(block.timestamp <= votingDeadline, "La votación ha finalizado");
46
47     // Verificar si el recurso existe
48     uint resourceIndex;
49     for (uint i = 0; i < resources.length; i++) {
50         if (resources[i].name == resourceId) {
51             resourceIndex = i;
52             break;
53         }
54     }
55
56     if (resourceIndex != 0) {
57         resources[resourceIndex].votes++;
58     } else {
59         Resource memory newResource = Resource(name);
60         newResource.votes++;
61         resources.push(newResource);
62     }
63
64     return resources[winningIndex].name;
65 }

```

REPOSITORIOS GITHUB

VICTORIA CODEANU

https://github.com/vittoric/proyecto_solidity_victoria_codreanu

JENNIFER GIRALDO
HERNÁNDEZ

https://github.com/jgiralohl/proyecto_solidity_Jennifer_Giraldo

VALESKA RIOS
ZEGARRA

https://github.com/valmarkeketing/proyecto_solidity_Valeska_RZegarra

VIDEO PRESENTACIÓN

[HTTPS://YOUTU.BE/QOMMC34NX6A](https://youtu.be/qommC34Nx6A)