**M6\_P2 DID METHOD**

1. **Introducción**

El objetivo de este ejercicio práctico fue implementar un DID Method utilizando un Smart Contract en Solidity. Un DID Method permite la creación y resolución de Identificadores Descentralizados (DIDs) en una red blockchain. Para lograr este objetivo, se desarrolló un Smart Contract que gestiona la creación, actualización, transferencia y resolución de DIDs, además de una interfaz de usuario (DApp) para interactuar con el contrato.

1. **Desarrollo del Smart Contract**

El Smart Contract ´DIDRegistry` fue implementado en Solidity y está compuesto de varias funciones:

1. **´createDID`**: Permite a los usuarios crear un nuevo DID proporcionando un identificador único y una descripción de datos asociada. Verifica que el DID no exista previamente y asigna el propietario y los datos asociados al DID.

2. **´updateDID`**: Permite al propietario de un DID actualizar la descripción de datos asociada a ese DID.

3. **´transferDID`**: Permite al propietario de un DID transferir la propiedad del DID a otra dirección.

4. **´resolveDID`**: Permite a cualquier usuario resolver un DID, es decir, obtener la descripción de datos asociada al DID.

El contrato también emite eventos (**´DIDCreated`**,**´DIDUpdated`**, **´DIDTransferred`**) para registrar las operaciones de las mismas.

1. **Desarrollo de la DApp**

Además del Smart Contract, se desarrolló una DApp utilizando la biblioteca React para proporcionar una interfaz de usuario amigable para interactuar con el contrato. La DApp se conecta a la red blockchain utilizando Web3.js, lo que permite a los usuarios:

- Crear nuevos DIDs proporcionando un identificador único y datos asociados.

- Actualizar la descripción de datos de un DID existente.

- Transferir la propiedad de un DID a otra dirección.

- Resolver un DID para recuperar su descripción de datos.

La DApp presenta formularios y botones que facilitan estas acciones y muestra los resultados en la interfaz del usuario

1. **Diagramas Utilizados**

Durante el desarrollo de este ejercicio, se utilizaron varios diagramas para comprender y visualizar el flujo de trabajo:

1. Diagrama de Secuencia de Creación de DID: Este diagrama ilustra el flujo de trabajo cuando un usuario crea un nuevo DID a través de la DApp, desde la interacción del usuario con la interfaz hasta la ejecución de la función `createDID` en el Smart Contract.

2. Diagrama de Secuencia de Resolución de DID: Este diagrama muestra cómo un usuario puede resolver un DID existente utilizando la DApp y el Smart Contract, recuperando la descripción de datos asociada.

3. Diagrama de Componentes de la DApp: Un diagrama de componentes utilizado para mostrar los componentes principales de la DApp, incluyendo la interfaz de usuario, la lógica de la aplicación y la conexión con Web3.js.

1. **Hipótesis consideradas para la realización de la prácica**
2. Solidity es un lenguaje adecuado para la implementación de contratos inteligentes:

Solidity es un lenguaje adecuado para desarrollar contratos inteligentes, ya que es ampliamente utilizado en la plataforma Ethereum.

1. Web3.js es una biblioteca adecuada para interactuar con contratos inteligentes:

Web3.js es una biblioteca adecuada para conectar la DApp con el contrato en la red Ethereum.

1. La funcionalidad del contrato y la DApp son suficientes para la gestión de DIDs:

Las operaciones de creación, actualización, transferencia y resolución de DIDs son suficientes para demostrar el concepto de un DID Method.

1. **Conclusiones**

El desarrollo de un DID Method utilizando un Smart Contract en Solidity y una DApp en React permitió demostrar la capacidad de crear sistemas de identidad descentralizada en una plataforma blockchain. El Smart Contract implementado permite la creación y gestión de DIDs, y la DApp proporciona una interfaz de usuario amigable para interactuar con el contrato.