1. Introduction

The web3 solution, BlockMed, is designed to model the prescription and dispensing circuit of medications. It implements the W3C's Self-Sovereign Identity (SSI) paradigm, using Decentralized Identifiers (DID) and Verifiable Credentials (VC) to ensure the authenticity, privacy, and security of interactions among system actors: Patients, Doctors, Pharmacies, and Health Insurance Providers (HIP).

2. System Architecture

2.1. **Frontend**

- **Layout:** Figma
- **Framework:** Web application based on React using https://ionicframework.com/.
- **Additional Libraries:** Cryptographic generators, QuarkID/KMS, QRScanner, Sound.

2.2. **Backend**

- **Development:** Implementation of the W3C SSI, DID and VC standards.
- **Language:** Typescript.
- **Database:** MongoDB for non-transactional data storage and off-chain storage.
- **API:** API-Zksync for publishing DIDs and credentials.
- **Verifiable Credentials (Prescriptions):** JWT (JSON Web Tokens).

2.3. **Blockchain and SSI**

- **Network:** Zksync.
- **Smart Contracts:** "RecetasW3" for managing medical prescription transactions (creation and dispensing) and "SimpleSidetreeAnchor" for managing identity creation (DIDs). Both contracts are written in Solidity.
- **DID Method:** Declaration of the "recetasbc" method and its behavior in the context of the SSI protocol DID representation scheme. Example: did:recetasbc:EiAIIUBzGj0fH7ZImdvDsCqu4tj6IwfxTSr_gdkXGe-fZw where:
  - "did" is the scheme,
  - "recetasbc" is the method,
  - "EiAIIUBzGj0fH7ZImdvDsCqu4tj6IwfxTSr_gdkXGe-fZw" is the specific method identifier.
- **Verifiable Credentials (Prescriptions):**
  - Prescription schema (issued by the doctor),
  - Dispensation schema (medication dispensing by the pharmacy),
  - Reception schema (confirmation of reception by the patient).

All these schemes are nested through a unique prescription identifier uniquely associated with the prescription and all subsequent states.

2.4. **Web3 Integration**

- **Decentralized Web Node (DWN):** Allows peer-to-peer information exchange according to the W3C SSI protocol.
- **IPFS:** Decentralized system for DID Documents associated with each generated DID.
- **Digital Wallets:** Two accounts are used, one for creating DIDs and another for recording prescription transactions.
- **SmartContracts:** RecetasW3 y SimpleSidetreeAnchor

3. Main Functionalities

3.1. **Identity Management**

- **DID Creation:** Each system actor (Patient, Doctor, Pharmacy, HIP) is identified by a DID.
- **Verifiable Credential Management:** The issuance and validation of Verifiable Credentials comply with W3C standards. Credentials are sent peer-to-peer; no data is stored in any database. Only dispensing information (medication and batch) is stored on the blockchain.

3.2. **Prescription and Medication Dispensing**

- **Medication Prescription:** Doctors issue prescriptions in the form of Verifiable Credentials. Doctors sign a transaction that is recorded on the blockchain (prescription schema). If the patient declares a HIP, the prescription is also sent to the HIP in the same format.
- **Prescription Sending:** Patients send their prescriptions to pharmacies in the form of Verifiable Credentials.
- **Medication Dispensing:** Pharmacies dispense medications to patients in the form of verifiable credentials (dispensation schema).
- **Dispensation Validation:** Patients validate the reception of medication at the pharmacy in the form of Verifiable Credentials (reception schema). Patients sign a transaction that is recorded on the blockchain.
- **Prescription History:** Activity can be tracked from the issuance and dispensing records (emitReceta and dispenseMedicamento functions of the SmartContract RecetasW3).

3.3. **Authentication and Authorization**

- **Registration and Login:** Since this is a POC, registration and authentication are not required.

- **Role Management:** Selectors are provided to set roles as
    - Patient,
    - Doctor,
    - Pharmacy.

The HIP role is preconfigured to facilitate the POC user experience.

4. Health Insurance Providers view

To provide HIP information in the POC, a service was developed to receive the information generated within the network and make it available to each HIP's infrastructure.
A collector associated with each HIP's DID that periodically receives prescriptions sent to their DID, in the context of the patient's health coverage.

Technology:

- Typescript service for decoding encrypted messages
- MongoDB for storage

5. Deployment and Maintenance

5.1. **Deployment on Production Environment:**

- Deployment of Zksync proxy on cloud servers (AWS).
- Deployment of DWN on cloud servers (AWS).
- Deployment of MongoDB (for temporary storage of encrypted credentials) on cloud servers (AWS).

Project Environments

- URL to open source code: https://gitlab.com/groups/recetasbc
- URL to live demo: https://app.recetasbc.com.ar
- Smart Contract RecetasW3: https://sepolia.explorer.zksync.io/address/0x330E512dDB94d2dd17D53816422Af7245BcC1fD1
- DIDs Transactions in ZkSync network: https://explorer.zksync.io/address/0x232e65C20af532344E4eA79cB0CdB15A9B5F995B

References

- Definition of DID, W3C: https://www.w3.org/TR/did-core/
- Definition of Verifiable Credential, W3C: https://www.w3.org/TR/vc-data-model/
- Trust over IP: https://trustoverip.org/wp-content/uploads/Introduction-to-ToIP-V2.0-2021-11-17.pdf
- Christopher Allen, The Path to Self-Sovereign Identity: https://www.coindesk.com/markets/2016/04/27/the-path-to-self-sovereign-identity

Version

1.0 - August 6, 2024

References - RecetasW3 Smart Contract - Source Code

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

contract RecetasW3 {
    struct Medicamento {
        string codigo;
        uint256 cantidad;
        string lote; // Nuevo campo para el lote
    }

    struct Receta {
        string hash;
        bool dispensada;
        string DIDfarmacia;
        Medicamento[] medicamentos;
        uint256 bloque; // Nuevo campo para el bloque de la transacción
    }

    mapping(string => Receta) public recetas;

    event RecetaEmitida(string hash);
    event RecetaDispensada(string hash, string DIDfarmacia, string lote);

    function emitirReceta(string memory _hash) public {
        require(bytes(_hash).length != 0, unicode"El hash no puede estar vacío.");
        require(bytes(recetas[_hash].hash).length == 0, unicode"La receta ya ha sido
    emitida.");

        Receta storage receta = recetas[_hash];
        receta.hash = _hash;
        receta.dispensada = false;
        receta.DIDfarmacia = "";
        receta.bloque = block.number; // Guardar el bloque de la transacción

        emit RecetaEmitida(_hash);
    }
```

```
    function dispensarMedicamento(string memory _hash, string memory
   _DIDfarmacia, Medicamento[] memory _medicamentos)
        public
    {
        Receta storage receta = recetas[_hash];
        require(bytes(recetas[_hash].hash).length != 0, unicode"La receta no existe.");
        require(!receta.dispensada, unicode"La receta ya fue dispensada.");

        receta.DIDfarmacia = _DIDfarmacia;
        receta.dispensada = true;

        for (uint256 i = 0; i < _medicamentos.length; i++) {
            receta.medicamentos.push(
                Medicamento({
                    codigo: _medicamentos[i].codigo,
                    cantidad: _medicamentos[i].cantidad,
                    lote: _medicamentos[i].lote // Añadir el lote aquí
                })
            );
        }

        emit RecetaDispensada(_hash, _DIDfarmacia, _medicamentos[0].lote); // Emitir
    evento con el lote del primer medicamento
    }

    function verificarReceta(string memory _hash) public view returns (Receta memory)
    {
        return recetas[_hash];
    }
}
```

References - SimpleSidetreeAnchor Smart Contract - Source Code

```
/*
 * Copyright 2020 - Transmute Industries Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *     http://www.apache.org/licenses/LICENSE-2.0
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
pragma solidity 0.8.16;
contract SimpleSidetreeAnchor {
    uint256 public transactionNumber = 0;
    event Anchor(
        bytes32 anchorFileHash,
        uint256 indexed transactionNumber,
        uint256 numberOfOperations,
        address writer
    );
    function anchorHash(bytes32 _anchorHash, uint256 _numberOfOperations)
        public
    {
        emit Anchor(_anchorHash, transactionNumber, _numberOfOperations , msg.sender);
        transactionNumber = transactionNumber + 1;
    }
}
```