COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO: ANEXO B

VACCINATION CARD MANAGEMENT TOOL POWERED BY BLOCKCHAIN

Alejandro Alzaga Sáez

Supervisor: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Vaccination Card Management Tool Powered by Blockchain

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Alejandro Alzaga Sáez Fecha: 28/ 06/ 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO

Digitally signed by 71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO
Date: 2023.06.29 08:31:45 +02'00'

Fdo.: Atilano Fernández-Pacheco Sánchez-Migallón Fecha: ……/ ……/ ……

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO: ANEXO B

VACCINATION CARD MANAGEMENT TOOL POWERED BY BLOCKCHAIN

Alejandro Alzaga Sáez

Supervisor: Atilano Fernández-Pacheco Sánchez-Migallón

HERRAMIENTA DE GESTIÓN DE CARTILLAS DE VACUNACIÓN POTENCIADA POR BLOCKCHAIN

Autor: Alzaga Sáez, Alejandro

Director: Fernández-Pacheco Sánchez-Migallón, Atilano

Entidad Colaboradora: ICAI - Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

En este Trabajo de Fin de Grado se ha diseñado y desarrollado un sistema de gestión de cartillas de vacunación que hace uso de una red blockchain para almacenar los registros de vacunación. El sistema comprende dos programas, uno optimizado para ordenador que usarán los trabajadores de la industria de la salud, y otro para móvil que usarán los usuarios. Se ha logrado un sistema fácil de usar y navegar que cumple con el objetivo de almacenar y visualizar correctamente la información sobre las vacunaciones en la red blockchain.

Palabras clave: Blockchain, cartillas de vacunación, inmutabilidad, DApps, contratos inteligentes.

1. Introducción

El auge de la tecnología blockchain ha generado amplio interés debido a sus aplicaciones potenciales más allá de las criptomonedas y los NFT. Desde sus orígenes en 2008, cuando Satoshi Nakamoto introdujo el concepto de una red peer-to-peer para abordar el problema del doble gasto [1], la tecnología blockchain ha evolucionado, transformándose en una herramienta poderosa para transacciones seguras y descentralizadas. La introducción de contratos inteligentes por parte de Nakamoto y el posterior desarrollo de plataformas de contratos inteligentes programables como Ethereum por Vitalik Buterin expandieron las posibilidades de la tecnología blockchain más allá de las transacciones financieras.

En la actualidad, estamos entrando a la era de Blockchain 4.0, caracterizada por la visión de un internet descentralizado e inteligente impulsado por protocolos blockchain [2]. En este contexto, numerosas industrias están explorando los beneficios posibles de la implementación de blockchain. Una de estas industrias es la de la salud, donde la necesidad de una distribución y gestión eficientes de las vacunas se ha vuelto cada vez más crítica. El uso de la tecnología blockchain en la gestión de registros de vacunación ofrece ventajas significativas.

Haciendo uso de la tecnología blockchain, los individuos tienen un mayor control sobre sus registros de vacunación, eliminando el riesgo de que las tarjetas físicas en papel se pierdan o se dañen. Los proveedores de atención médica pueden acceder a información precisa y actualizada, lo que conduce a un desempeño de sus funciones más eficiente. La naturaleza compartida y transparente de la blockchain garantiza que las acciones de todos los participantes se registren de manera inmutable, abordando desafíos como la transferencia lenta de datos, la coordinación de consentimientos y la agregación de datos con fines de investigación [3].

La implementación de blockchain en la industria de las vacunas no solo mejora la gestión de los registros individuales, sino que también contribuye a la homogeneización de las tarjetas de vacunación en diferentes territorios. Con un sistema estandarizado e interoperable, los individuos pueden ver fácilmente sus registros desde cualquier lugar, y los administradores pueden agregar, ver y actualizar los registros de manera fluida mientras interactúan con la blockchain.

2. Definición del proyecto

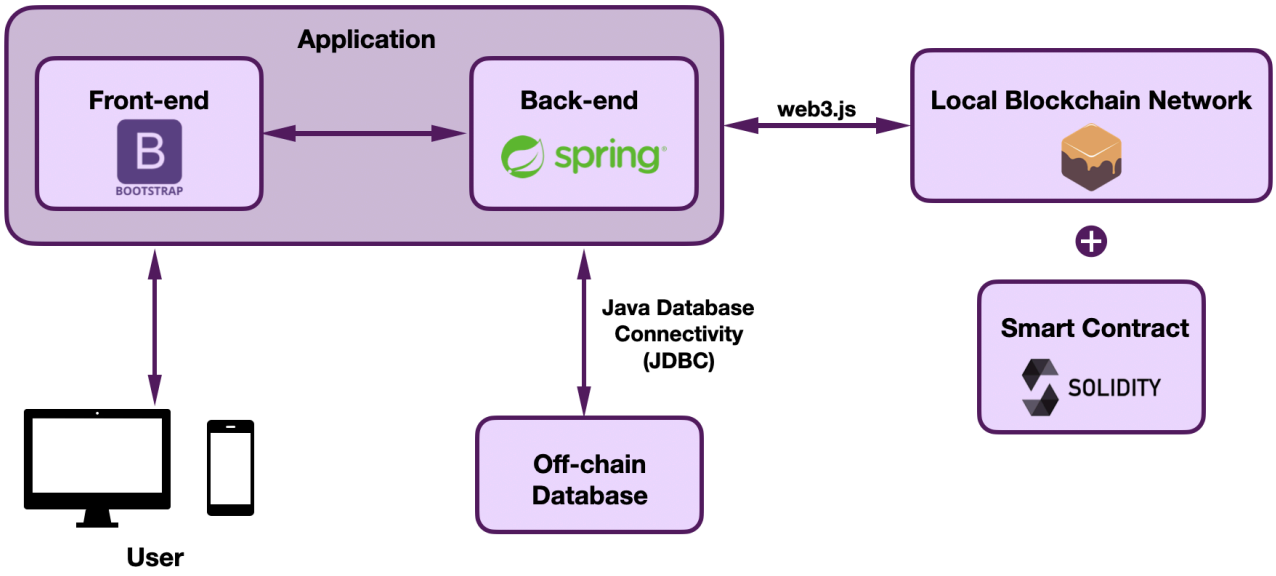
En este Trabajo de Fin de Grado (TFG) se ha buscado diseñar un sistema fácil de usar y navegar que se base en el almacenado de los registros de vacunación de los usuarios en la blockchain, garantizando así la integridad e inmutabilidad de estos datos de naturaleza sensible. Con este objetivo en mente, se ha desarrollado un contrato inteligente que interactúa con una red local de blockchain y permite a usuarios y trabajadores de la industria de la salud añadir, ver y modificar registros de vacunación.

Para definir el sistema en sí, primero se hizo un análisis del mercado y de la tecnología blockchain para entender las necesidades que requerían ser cubiertas, y las posibilidades que la tecnología blockchain ofrece. Este análisis ayudó a definir, mediante historias de usuario, las funcionalidades que el sistema debe ofrecer a los diversos actores que harán uso de la aplicación. Así pues, se decidió crear un sistema que se apoyase en tres pilares: el contrato inteligente que interactuará con la red blockchain, el back-end del sistema apoyado en el framework Spring Boot y una base de datos fuera de la cadena de bloques, y la aplicación tanto web como móvil con la que los usuarios interactuarán, y que a su vez se conectará con el contrato inteligente con el uso de la librería web3.js.

Las funcionalidades principales definidas son, en el caso de los trabajadores de la industria de la salud, permitirles añadir, ver o actualizar registros de los usuarios, y en el caso de los usuarios poder ver sus registros y futuras citas de vacunación.

3. Descripción del sistema

El sistema desarrollado se compone de la aplicación, la red local de Ethereum creada en Ganache, el contrato inteligente y la base de datos relacional establecida fuera de la cadena.



Fuente: Elaboración propia

Figura 1: Diagrama de la arquitectura del proyecto

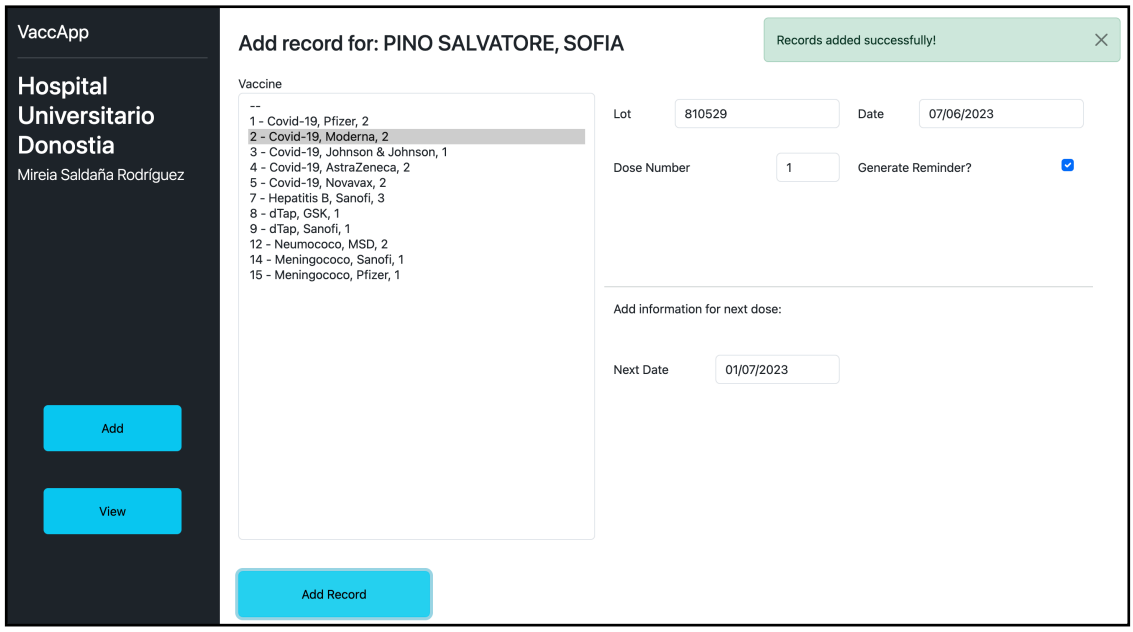
La red de blockchain elegida ha sido Ethereum debido a su amplia documentación y amplio número de herramientas a su disposición. Ethereum además nació con el objetivo de ser una plataforma para la programación de contractos inteligentes, y por tanto es una elección sensata para el desarrollo de aplicaciones descentralizadas (DApps) como esta. Todas las interacciones y transacciones con esta red se harán a través del contrato inteligente, compilado en el lenguaje de programación Solidity.

La aplicación en sí se puede dividir en el front-end o interfaz de usuario, y el back-end o servidor. El back-end se ha diseñado con el framework Spring Boot en mente, que facilita la modularización e implementación de aplicaciones web. La aplicación interactúa con web3.js con el contracto inteligente y, por ende, con la red de blockchain, y a través de una REST API diseñada con una base de datos fuera de la cadena de bloques, que almacena los datos de la aplicación como los usuarios, doctores u hospitales, que no requieren ser almacenados en la blockchain, haciendo menos costos y más rápido su acceso, y mejorando en definitiva la experiencia del usuario al usar la aplicación. En cuanto a la interfaz de usuario, se han diseñado una aplicación web para el uso de los trabajadores de la industria de la salud, y una aplicación móvil para los usuarios promedios.

4. Resultados

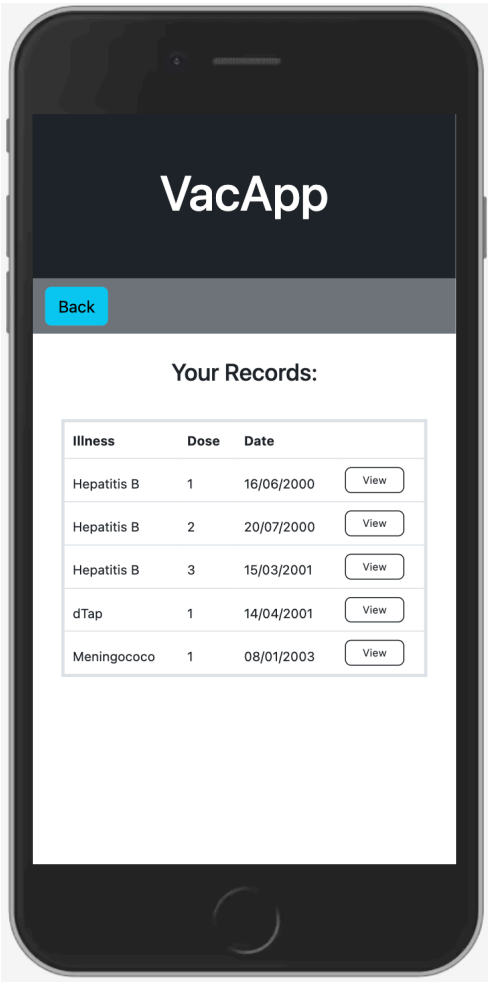
Los resultados de este TFG desarrollado han sido satisfactorios, y los objetivos marcados se han alcanzado. El contrato inteligente interactúa correctamente con la blockchain y registra las transacciones de añadir y actualizar registros. La API recoge la información de la base de datos apropiadamente, permitiendo a los usuarios realizar las funciones de inicio de sesión y visualización, y la interfaz desarrollada consigue el objetivo de la fácil navegabilidad, la cohesión, y el acceso rápido y eficaz de los datos y registros.

De esta manera, la aplicación, REST API y contrato inteligente desarrollados permiten a los doctores almacenar nuevos registros, generar futuras citas de vacunación, ver y actualizar registros ya sean pendientes o no, y a los usuarios consultar sus cartillas y futuras citas en cualquier momento.



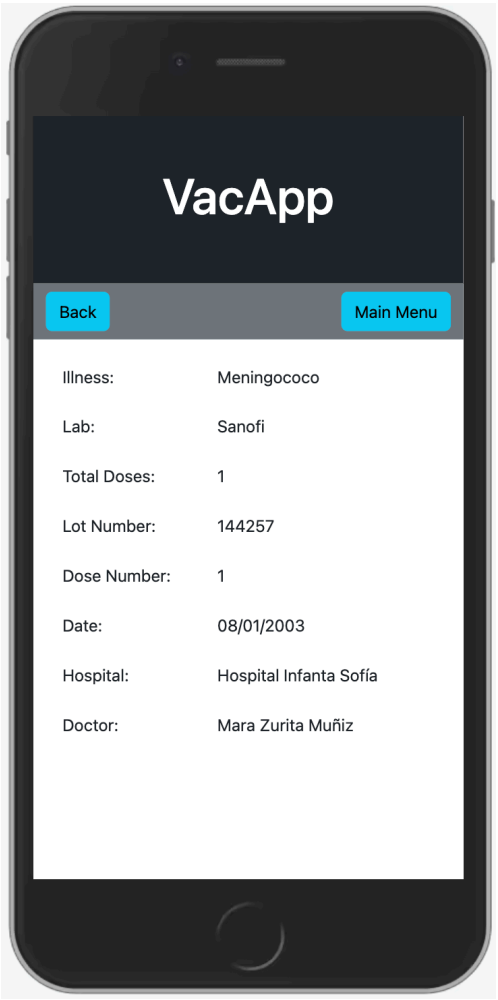
Fuente: Elaboración propia

Figura 2: Vista de administrador para añadir registros



Fuente: Elaboración propia

Figura 3: Vista del usuario de cartillas de vacunación



Fuente: Elaboración propia

Figura 4: Vista del usuario de información detallada

5. Conclusiones

Con el desarrollo de este TFG se han alcanzado los objetivos propuestos, demostrando la viabilidad de una herramienta como esta y los beneficios que ofrece usar la tecnología blockchain para almacenar y tratar datos sensibles como las cartillas de vacunación. Los trabajadores pueden realizar todas las funciones definidas inicialmente, y los usuarios acceder a estos datos fácil y rápidamente.

El desarrollo de este sistema planta los cimientos de futuras herramientas que puedan expandir en estas funcionalidades, en particular centrando la atención en la cadena de suministro de las vacunas. Blockchain garantiza la inmutabilidad, integridad y trazabilidad de los datos, lo cual lo convierte en una tecnología de indispensable valor para el almacenado de datos sobre vacunas y sobre su recorrido desde los laboratorios hasta los pacientes.

6. Referencias

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] LeewayHertz. (n.d.). Blockchain 4.0. Retrieved from https://www.leewayhertz.com/blockchain-4-0/

[3] Dubovitskaya A, Xu Z, Ryu S, Schumacher M, Wang F. Secure and Trustable Electronic Medical Records Sharing using Blockchain. AMIA Annu Symp Proc. 2018 Apr 16;2017:650-659. PMID: 29854130; PMCID: PMC5977675.

VACCINATION CARD MANAGEMENT TOOL POWERED BY BLOCKCHAIN

Author: Alzaga Sáez, Alejandro

Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano

Collaborating Entity: ICAI - Universidad Pontificia Comillas

ABSTRACT

In this Trabajo de Fin de Grado a vaccination card management system has been designed and developed that utilizes a blockchain network to store vaccination records. The system consists of two programs, one optimized for computers to be used by healthcare industry workers, and another for mobile devices to be used by users. A user-friendly and navigable system has been achieved, fulfilling the objective of storing and correctly displaying vaccination information on the blockchain network.

Keywords: Blockchain, vaccination cards, immutability, DApps, smart contracts.

1. Introduction

The rise of blockchain technology has captured widespread attention due to its potential applications beyond cryptocurrencies and NFTs. With origins dating back to 2008, when Satoshi Nakamoto introduced the concept of a peer-to-peer network to address the double spending problem [1], blockchain has evolved into a powerful tool for secure and decentralized transactions. The introduction of smart contracts by Nakamoto and the subsequent development of programmable smart contract platforms like Ethereum by Vitalik Buterin expanded the possibilities of blockchain technology beyond financial transactions.

Nowadays, we are entering the era of Blockchain 4.0, characterized by the vision of a decentralized and intelligent internet driven by blockchain protocols [5]. In this context, numerous industries are exploring the potential benefits of blockchain

implementation. One such industry is vaccines, where the need for efficient vaccine distribution and management has become increasingly critical. The use of blockchain technology in managing vaccination records offers significant advantages.

By leveraging blockchain, individuals gain greater control over their vaccination records, eliminating the risk of physical cards being lost or damaged. Healthcare providers can access accurate and up-to-date information, leading to more efficient healthcare delivery. The shared and transparent nature of blockchain ensures that all participants' actions are recorded immutably, addressing challenges such as slow data transfer, consent coordination, and data aggregation for research purposes [7].

Implementing blockchain in the vaccine industry not only improves individual record management but also contributes to the homogenization of vaccination cards across different territories. With a standardized and interoperable system, individuals can easily view their records from anywhere, and administrators can seamlessly add, view, and update records while interacting with the blockchain.

2. Project definition

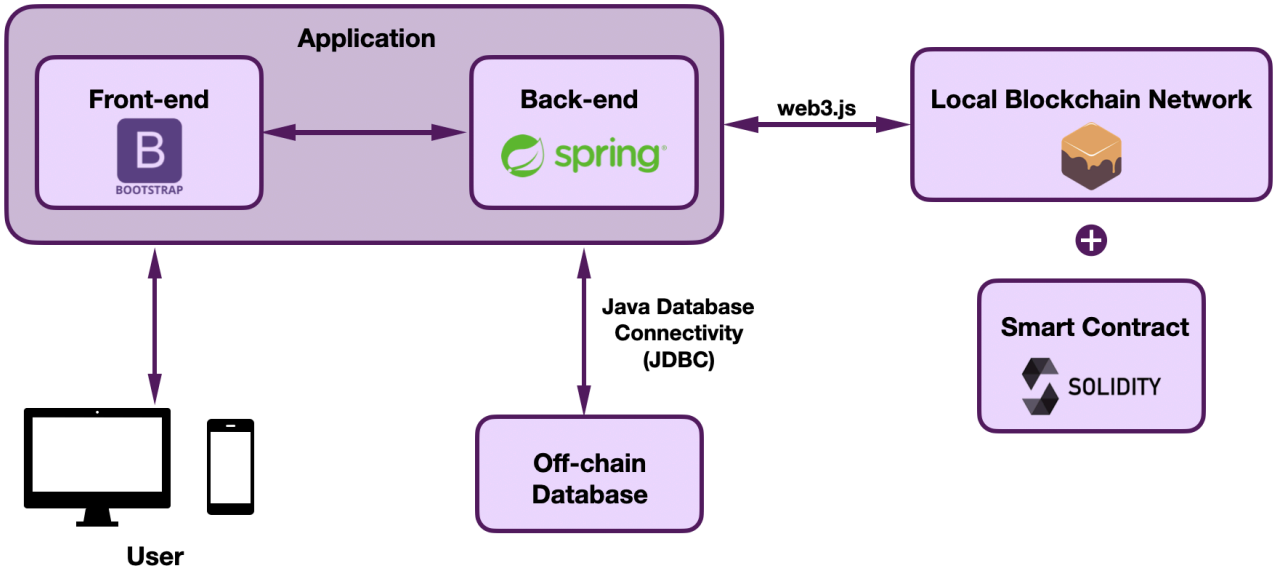
In this Trabajo de Fin de Grado (TFG) the aim was to design a user-friendly and navigable system that is based on storing users' vaccination records on the blockchain, thus ensuring the integrity and immutability of this sensitive data. With this objective in mind, a smart contract was developed to interact with a local blockchain network, enabling users and healthcare industry workers to add, view, and modify vaccination records.

To define the system itself, an analysis of the market and blockchain technology was conducted to understand the needs that needed to be addressed and the possibilities that blockchain technology offers. This analysis helped define the system's functionalities, through user stories, that the system should offer to the various actors who will use the application. Therefore, it was decided to create a system supported by three pillars: the smart contract that interacts with the blockchain network, the system's backend supported by the Spring Boot framework and an off-chain database, and the web and mobile application with which users will interact, that will in turn connect to the smart contract using the web3.js library.

The main functionalities defined are, for healthcare industry workers, allowing them to add, view, or update user records, and for users to be able to view their records and future vaccination appointments.

3. System description

The developed system consists of the application, the local Ethereum network created in Ganache, the smart contract, and the relational database established off-chain.



Source: Own elaboration

Figure 5: Architectural diagram of the project

The chosen blockchain network is Ethereum due to its extensive documentation and wide range of tools available. Ethereum was designed to be a platform for programming smart contracts, making it a sensible choice for developing decentralized applications (DApps) like this one. All interactions and transactions with this network will be done through the smart contract, compiled in the Solidity programming language.

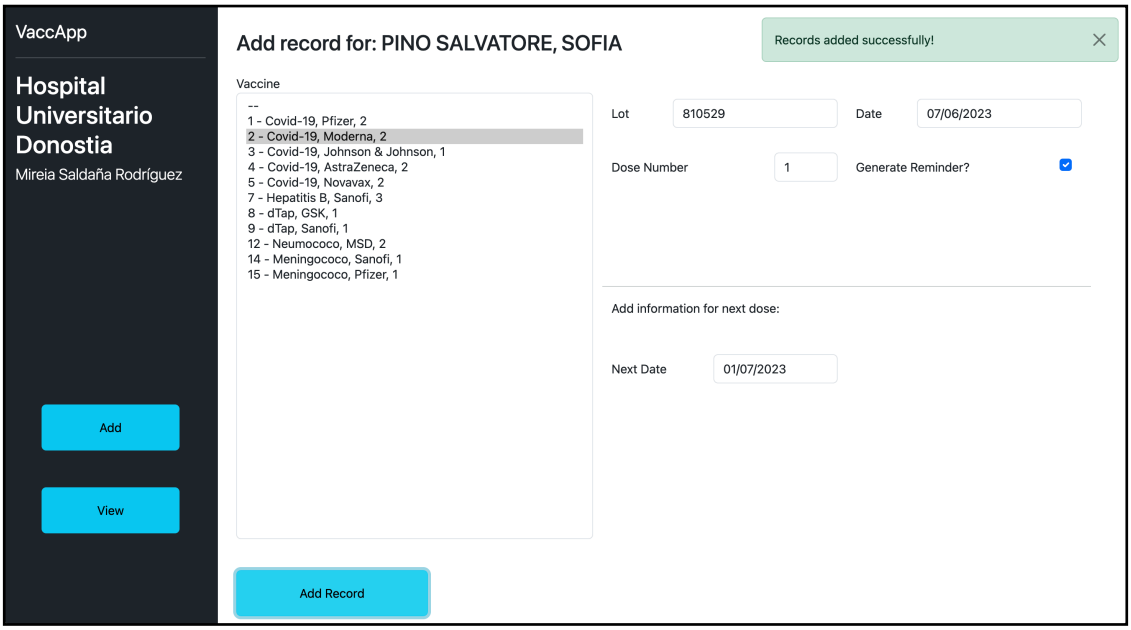
The application itself can be divided into the front-end or user interface and the back-end or server. The back-end has been designed with the Spring Boot framework in mind, which facilitates the modularization and implementation of web applications.

The application interacts with web3.js to communicate with the smart contract and, consequently, with the blockchain network. It also includes a REST API designed with an off-chain database that stores application data such as users, doctors, and hospitals. This data does not need to be stored on the blockchain, which reduces costs, improves access speed, and enhances the overall user experience. As for the user interface, a web application has been designed for healthcare industry workers, and a mobile application for average users.

4. Results

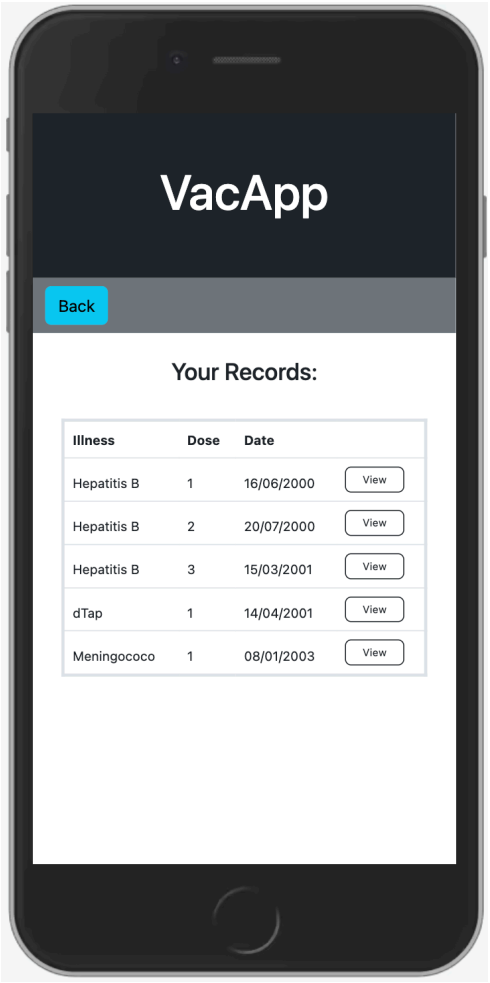
The results of this developed TFG have been satisfactory, and the set objectives have been achieved. The smart contract interacts correctly with the blockchain and records the transactions of adding and updating records. The API retrieves the information from the database appropriately, allowing users to perform login and viewing functions, and the developed interface achieves the goal of easy navigation, cohesiveness, and quick and efficient access to data and records.

In this way, the developed application, REST API, and smart contract allow doctors to store new records, generate future vaccination appointments, view and update both pending and completed records, and enable users to consult their vaccination records and future appointments at any time.



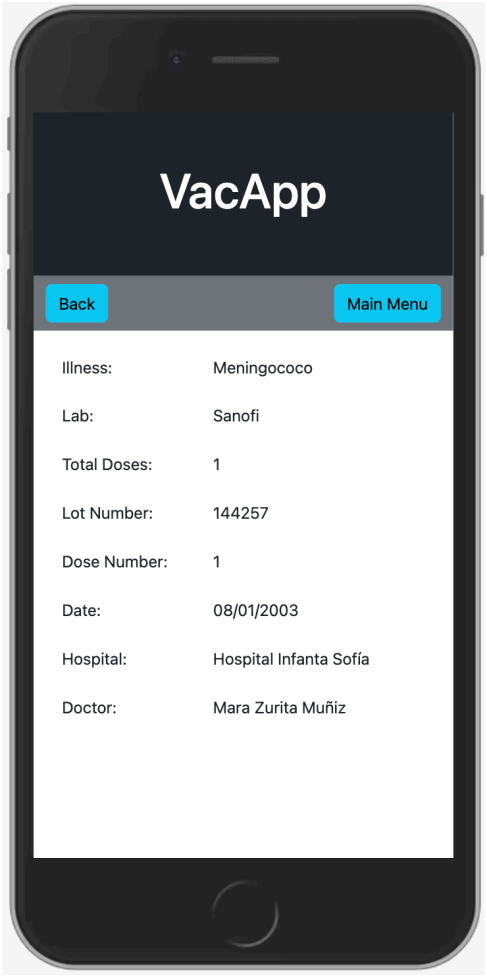
Source: Own elaboration

Figure 6: Admin view for adding records



Source: Own elaboration

Figure 7: User view of vaccination cards



Source: Own elaboration

Figure 8: User view of extended information

5. Conclusions

With the development of this TFG the proposed objectives have been achieved, demonstrating the viability of a tool like this and the benefits it offers to store and handle sensitive data such as vaccination records using blockchain technology. The healthcare workers can perform all the initially defined functions, and users can access this data easily and quickly.

The development of this system lays the groundwork for future tools that can expand on these functionalities, particularly focusing on the vaccine supply chain. Blockchain ensures the immutability, integrity, and traceability of data, making it an invaluable technology for storing data on vaccines and their journey from laboratories to patients.

6. References

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] LeewayHertz. (n.d.). Blockchain 4.0. Retrieved from https://www.leewayhertz.com/blockchain-4-0/

[3] Dubovitskaya A, Xu Z, Ryu S, Schumacher M, Wang F. Secure and Trustable Electronic Medical Records Sharing using Blockchain. AMIA Annu Symp Proc. 2018 Apr 16;2017:650-659. PMID: 29854130; PMCID: PMC5977675.

CONTENTS

FIGURE INDEX

TABLE INDEX

1 INTRODUCTION

The rapid pace of technological advancement continues to shape our world, and one term that has consistently captured attention is blockchain. While often associated with non-fungible tokens (NFTs) and cryptocurrencies, the potential of blockchain extends far beyond these realms. This distributed ledger technology possesses a multitude of applications yet to be fully explored and exploited.

The origins of blockchain can be traced back to 2008 when Satoshi Nakamoto proposed a peer-to-peer network to address the double spending problem [1]. This marked the beginning of what we now recognize as blockchain. Building upon established cryptographic techniques for ownership management and a consensus algorithm for tracking coin ownership, Nakamoto introduced a decentralized currency. Notably, the concept of "smart contracts" emerged earlier in 1994, defined by Nick Szabo as digital protocols using mathematical algorithms to execute transactions and establish secure connections across computer networks [2]. However, it was Nakamoto who effectively brought smart contracts to life. Subsequently, in 2014, Ethereum, developed by Vitalik Buterin, introduced a platform for programmable smart contracts [3]. This marked a significant milestone, expanding the potential applications of smart contracts and blockchain beyond financial transactions, and was coined as "blockchain 2.0." Further developments followed, leading to "blockchain 3.0," which emphasized decentralized applications (DApps) and was seen as a framework for business and institutional use [4]. Currently, we are witnessing the emergence of Blockchain 4.0, focusing on Web3 and aiming to create an autonomous, open, and intelligent internet driven by decentralized protocols provided by blockchain technology [5].

Numerous industries stand to benefit immensely from the implementation of blockchain technology. One industry in particular, which has gained significant attention in recent years, is the vaccine industry. The ongoing global immunization efforts and the need for efficient vaccine distribution and management have underscored the importance of robust and secure systems for maintaining vaccination records [6]. Blockchain's inherent features hold immense promise in revolutionizing the management of individual vaccination cards.

Leveraging blockchain technology for vaccination card management offers a multitude of potential benefits. Individuals gain greater control and ownership over their vaccination records, eliminating the reliance on physical cards that can be easily lost or damaged. Healthcare providers can access verified and up-to-date information, enabling more efficient and accurate healthcare delivery. Moreover, blockchain's ability to provide a shared, immutable, and transparent history of all participants' actions within the network addresses challenges such as slow data transfer, consent coordination in patient transfers, and data aggregation for research purposes [7].

2 DESCRIPTION OF TECHNOLOGIES

2.1 BLOCKCHAIN

According to IBM, blockchain is a shared and immutable ledger that facilitates the recording of transactions and tracking of assets in a business network. An asset can be tangible, such as a house, car, cash, or land, or intangible, including intellectual property, patents, copyrights, and branding. With each transaction, a data block is added to the chain, containing relevant information such as who, what, when, where, how much, and even additional conditions [8]. The chain represents a literal connection between blocks, maintaining a chronological sequence of events. Importantly, the blockchain is irreversible, ensuring the integrity of recorded data.

In this decentralized system, no participant has the ability to modify uploaded data. Updates to the information can only be made through new transactions. This decentralized and distributed ledger structure is one of the significant benefits of blockchain, as it eliminates the reliance on a single "master administrator" and reduces the risk of a single point of failure. Moreover, the shared ledger system prevents double spending issue, which was one of the original objectives of Bitcoin as proposed by Nakamoto [1].

For this project, blockchain technology is employed to record vaccination records transactions, including the addition, update, and viewing of users' vaccination records. Given the sensitive nature of this data, leveraging the enhanced security, traceability, and data integrity provided by blockchain is an obvious choice.

2.2 ETHEREUM

At its core, Ethereum is a decentralized global software platform powered by blockchain technology. It has been specifically designed to offer scalability, programmability, security, and decentralization [9]. Ethereum is the preferred blockchain for developers and enterprises engaged in the development of decentralized applications (DApps) due to its native support for smart contracts. As mentioned earlier, a key distinction between Ethereum and Bitcoin is that Ethereum is programmable, enabling the construction and deployment of decentralized

applications on its network [10]. This feature empowers users to execute not only simple transactions but also complex functions and programs that interact with the Ethereum blockchain.

2.2.1 SMART CONTRACTS

Smart contracts are code-based agreements that run on a blockchain. They enable the automation and self-enforcement of contract terms, ensuring accuracy, transparency, and integrity in transactions without relying on intermediaries. In practical terms, smart contracts function like any other programming language, with specific inputs yielding different outputs. However, the unique aspect of smart contracts is their ability to interact with and record transactions on the blockchain.

Despite their seemingly simple nature, smart contracts offer immense opportunities. They can be utilized to record financial transactions between entities or individuals, track the entire journey of a food item from its raw materials to its presence on a supermarket shelf, or, in the context of this project, facilitate the administration of vaccines to individuals.

The execution of these contracts is carried out by various Ethereum clients, each identified by a unique address. Smart contracts can involve interactions with other clients or with the contract address itself to upload or retrieve specific information

In order to write these contracts multiple coding languages have been developed. The two most popular ones are Solidity and Vyper, that have replaced Serpent, one of the initial programming languages for writing smart contracts on the Ethereum platform.

- **Solidity**: Solidity is the most widely used programming language for developing smart contracts on the Ethereum platform. It is a statically typed and contract-oriented language. Solidity code is compiled into bytecode that can be executed on the Ethereum Virtual Machine (EVM). Here is an example of code:

```
pragma solidity ^0.8.0;

contract HelloWorld {
    string private message;
    function setMessage(string memory _newMessage) public {
        message = _newMessage;
    }

    function getMessage() public view returns (string memory)    {
        return message;
    }
}
```

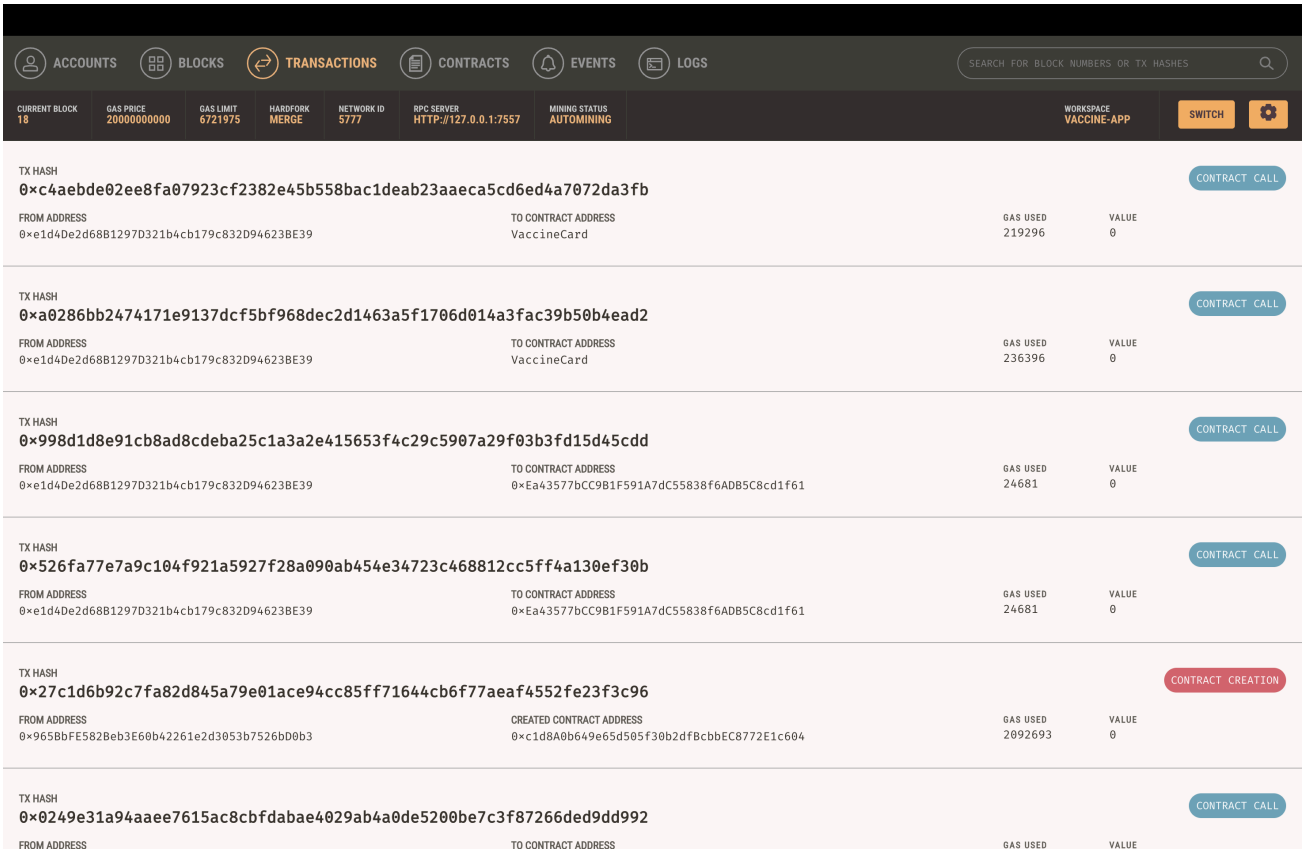
- **Vyper**: Vyper is a high-level, Python-inspired programming language specifically designed for writing secure and auditable smart contracts on the Ethereum platform. It emphasizes code simplicity and readability. Here is an example of code:

```
# Example smart contract to implement a simple auction
@public
@payable
def bid():
    highestBid: uint256 = self.balance
    refundAddress: address = msg.sender
    if highestBid > 0:
        send(refundAddress, highestBid)
    highestBid = msg.value
```

For this project, Solidity was the chosen language due to functionality, tools and extensions available and overall ease-of-use.

2.2.2 GANACHE

In order to implement a blockchain-based app, a blockchain network is required. Public networks are available, but each transaction incurs in a real-world cost, even if using testnets. That is where Ganache comes in handy, as it allows developers to create a private, local Ethereum network in our device, one that simulates currency to execute the transactions. No real cost is attached to those transactions, allowing us to deploy our contract and interact with it freely.



Source: Own elaboration

Figure 9: Screen view of Ganache, transactions tab

2.3 APP

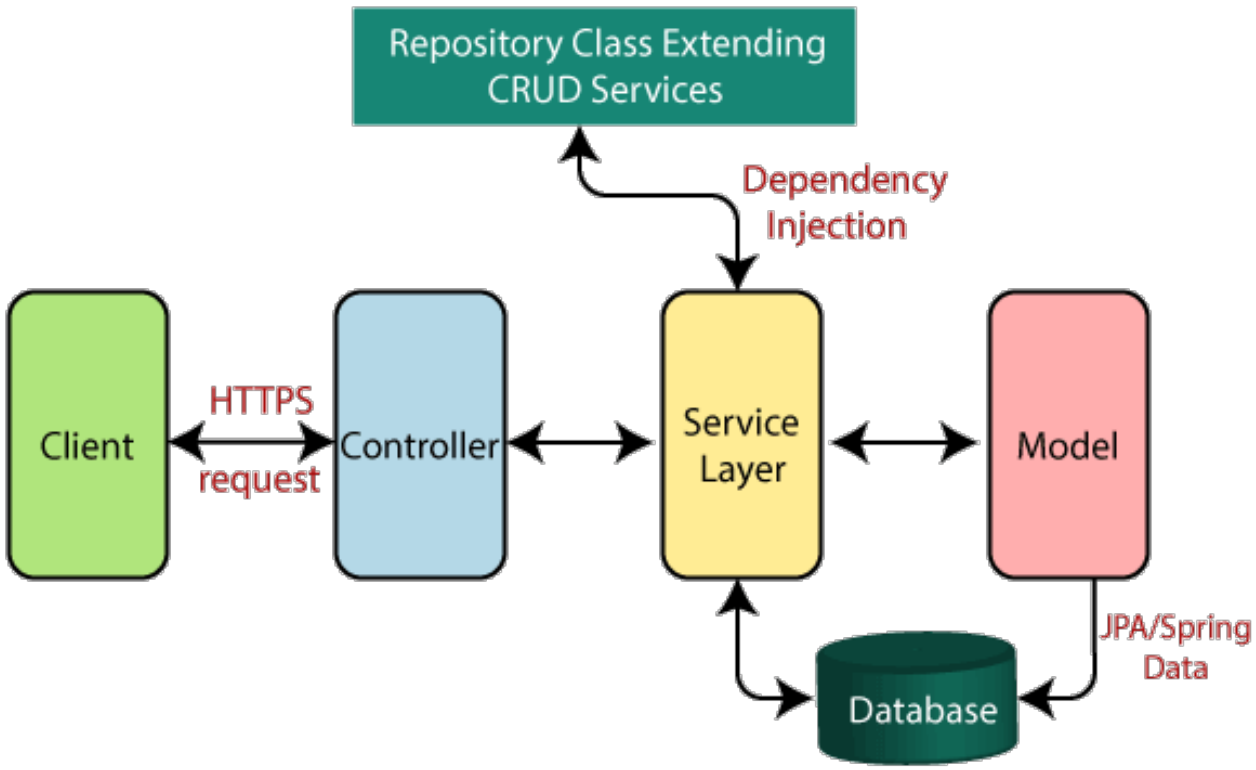
Here we will detail the multiple technologies and coding languages used in order to develop the app itself, from the back-end to the front-end.

2.3.1 SPRING BOOT

Spring Boot is an open-source Java framework designed to simplify the development of stand-alone, production-grade applications. One of the key features of Spring Boot is its intrinsic server capability, which allows applications to be packaged as executable JAR files. This means that the application can be run with a simple command, without the need for deploying it to a separate server.

Spring Boot also provides a comprehensive ecosystem of modules and libraries that integrate with other popular frameworks, such as Spring Data for database access,

Spring Security for authentication and authorization, and Spring MVC for building web applications. Most of this will be of great use for our purposes.



Source: https://www.grupomost.com/software-factory/

Figure 10: Spring Boot framework structure diagram

2.3.2 HTML

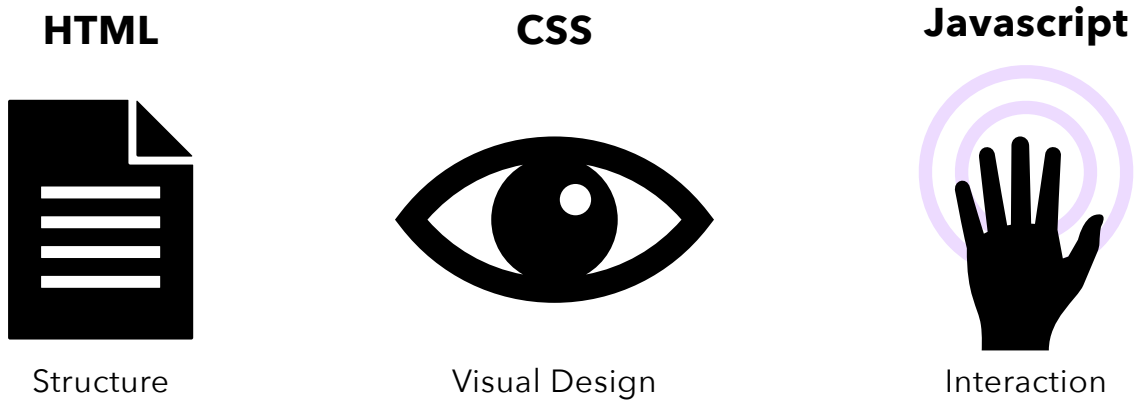
HTML will be used to code the multiple views of the web app itself, that will include a user view (mobile-based) and a admin view (web-based). These HTML documents are the foundation of each web page, that will be complimented with JavaScript and CSS to generate the full package.

2.3.3 JAVASCRIPT

High-level, object-oriented programming language that is native to every web browser. It has been used mainly to implement the Web3.js library used to interact with the smart contract and the Ganache local blockchain network.

2.3.4 CSS

CSS, short for Cascading Style Sheets, is a language used to style and format HTML Web documents. It allows to change parameters for different web elements to give them distinct looks. Instead of generating these CSS files from scratch, usually Bootstrap is used, which offers a library of different frameworks and individual elements that can be implemented in your website and further modified to your liking.



Source: Own elaboration

Figure 11: Illustration of foundations of the developed app

2.3.5 JAVA DATABASE CONNECTIVITY

Blockchain database capabilities are strong and secure as previously discussed, but they are not as rapid as regular databases. For that very reason, usually DApps implement a hybrid approach to data management, using on-chain and off-chain databases depending on the requirements of the data stored. For our application, we can distinguish two main datasets to be stored: vaccine records and login information. For login information, we will use Java Database Conenctivity (JDBC) instead of on-chain databases.

2.4 WEB3.JS

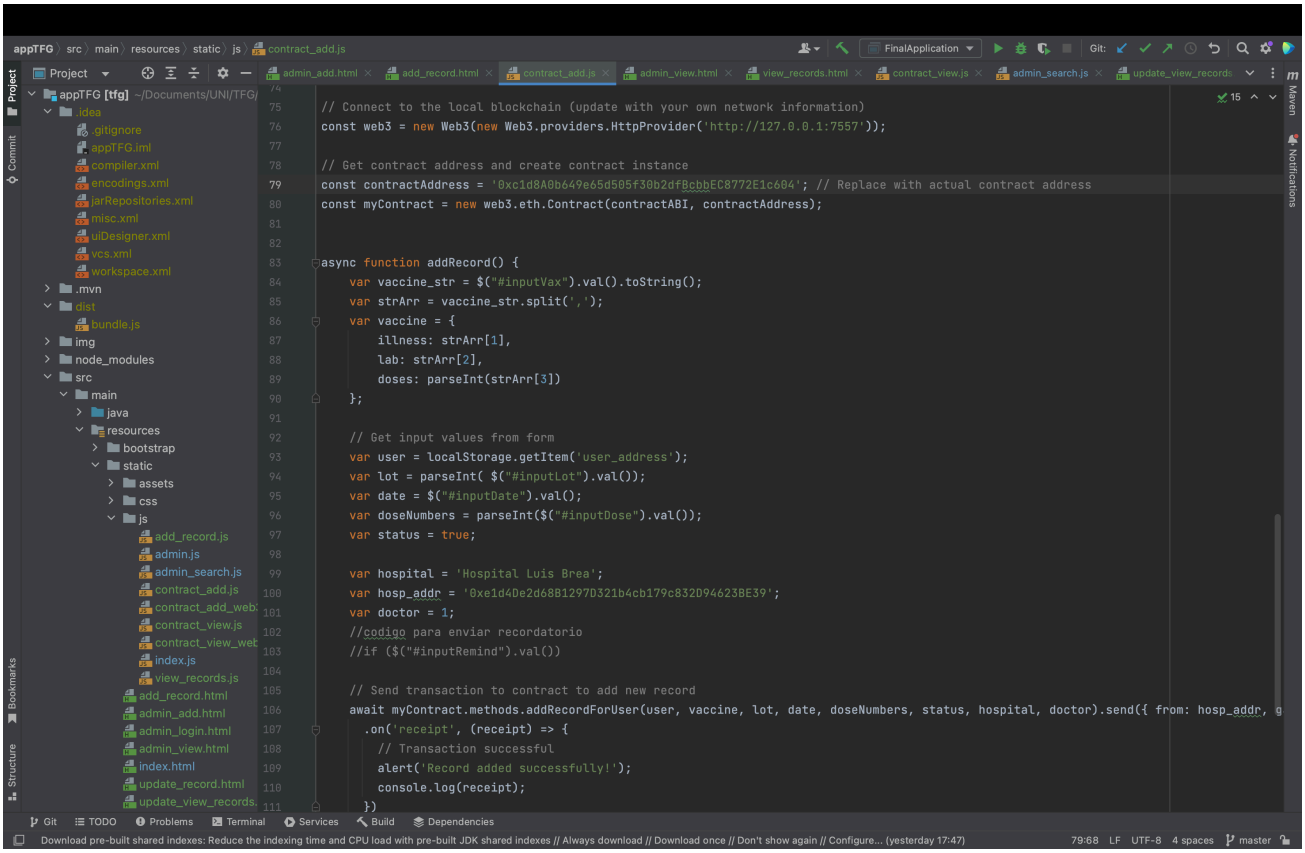
Web3 is a JavaScript library that provides an interface for interacting with Ethereum blockchain networks. It allows developers to build DApps that can read from and write to the blockchain. Web3 provides functions to connect to Ethereum nodes, send transactions, interact with smart contracts, and handle events.

2.5 IDE

IDEs (Integrated Development Environment) are software applications that include tools useful for software development, such as a code editor, a compiler or interpreter, debugging tools. For this project two of them were used:

2.5.1 INTELLIJ

IntelliJ IDEA is a popular Java IDE developed by JetBrains. IntelliJ supports various programming languages and frameworks through extensions available to users, so it was used for Java (Spring Boot), but also for HTML and JavaScript.



Source: Own elaboration

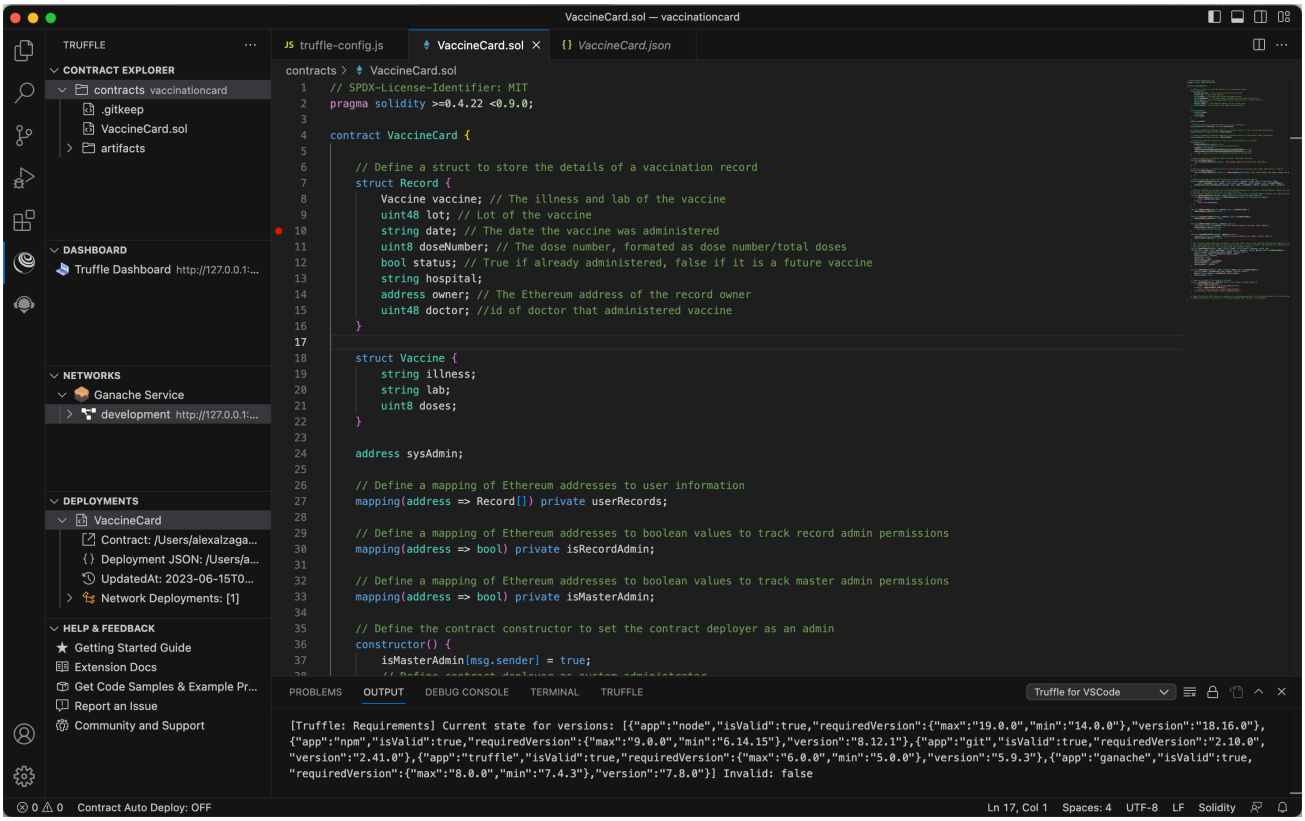
Figure 12: View of IntelliJ user interface

2.5.2 VS CODE

Similarly, Visual Studio Code is a lightweight and highly extensible source code editor developed by Microsoft. It supports multiple programming languages as well, even more than IntelliJ, as it has a bigger range of free extensions to be used. In this project it was used to develop the Solidity smart contract.

2.5.2.1 TRUFFLE SUITE

Truffle Suite is a development framework and suite of tools for building, testing, and deploying smart contracts on the Ethereum blockchain. It includes Truffle, which is a development environment and testing framework, Ganache, and Drizzle, a collection of front-end libraries for building DApps. Its extension for VS Code was also used, allowing us to connect to our previously-created Ganache local network.

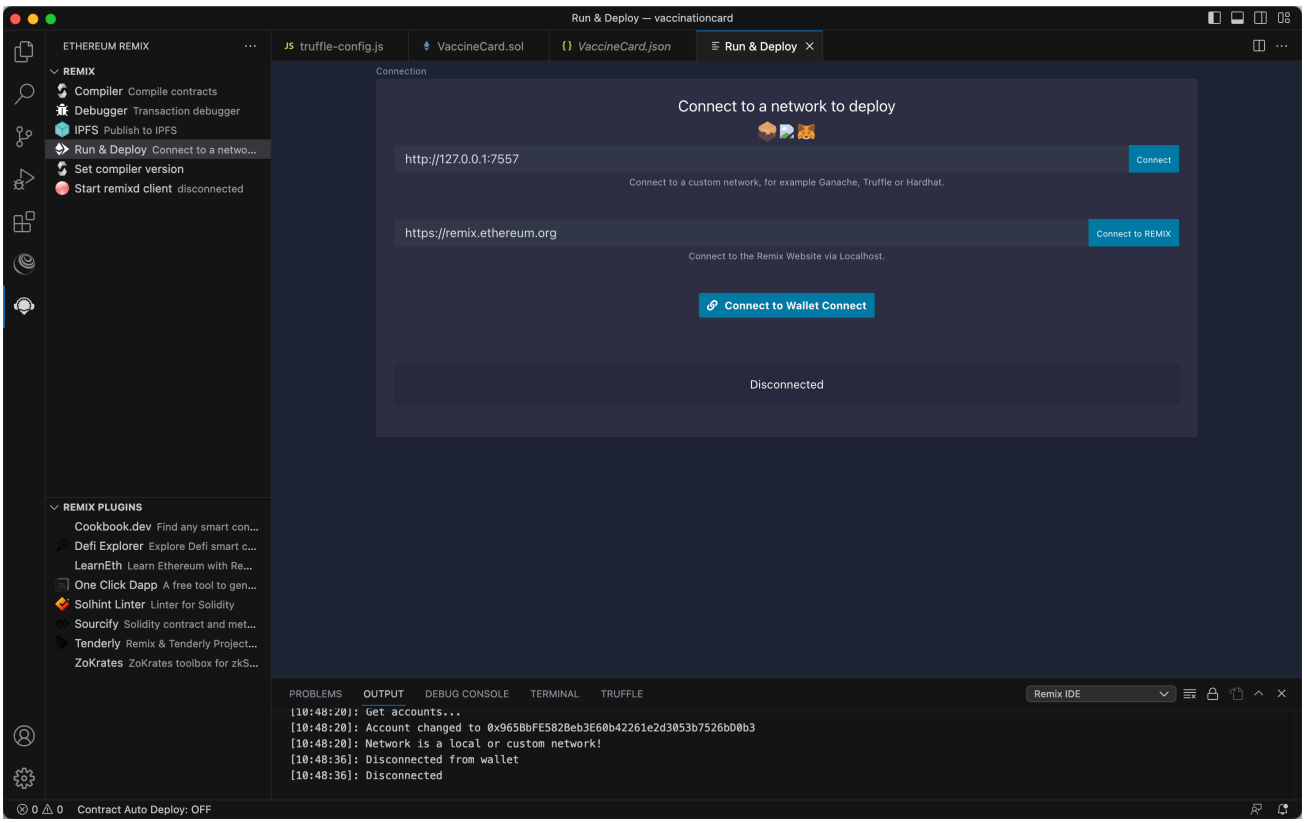


Source: Own elaboration

Figure 13: View of VS Code user interface, including Truffle Suite plugin

2.5.2.2 ETHEREUM REMIX

Ethereum Remix is a web-based IDE specifically designed for developing and testing smart contracts on the Ethereum blockchain. VS Code offers an extensions for it that was used in this project. The aforementioned extension allows users to directly test contracts and its functions, in order to debug them



Source: Own elaboration

Figure 14: View of VS Code user interface, including Ethereum Remix plugin

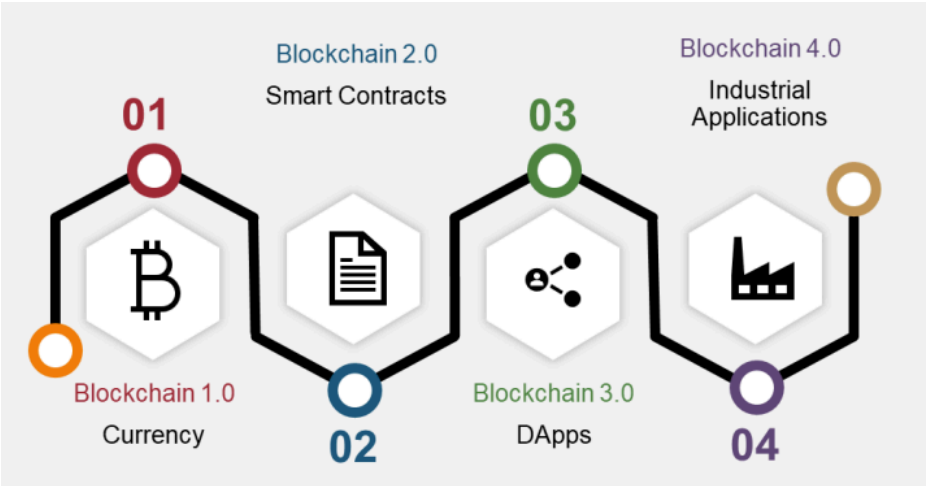
3 STATE OF THE ART

3.1 BLOCKCHAIN

Blockchain technology has witnessed significant advancements in recent years, leading to its widespread adoption in various industries. Its potential applications include, among several others, fashion supply chains streamlining, food safety regulations and tracking of food items from growers to consumers, or tracing the origins of precious gemstones to ensure ethical sourcing [11]. The versatility of blockchain is evident through its integration into numerous sectors, highlighting the advantages it offers and the inherent strengths it possesses.

3.2 BLOCKCHAIN 4.0

The current iteration of blockchain technology, coined as Blockchain 4.0, has Web3.0 and the Metaverse as its main key components. As mentioned before, each main Blockchain iteration was marked by a remarkable milestone: 1.0 brought the cryptocurrency, 2.0 the smart contracts and 3.0 the DApps. Blockchain 4.0, instead of focusing on solving and polishing the previous generation (like 3.0 did), focuses on using this technology to innovate. This generation advances greatly in terms of scalability, interoperability and security features [12]. For the purposes of this project, it is essential to delve into Web 3.0 and how it promises to change the internet as we, consumers, know it and make use of it.



Source: Mohanty, D.; Anand, D.; Aljahdali, H.M.; Villar, S.G. Blockchain Interoperability: Towards a Sustainable Payment System. Sustainability 2022, 14, 913. https://doi.org/10.3390/su14020913

Figure 15: Illustration of blockchain evolution and phases

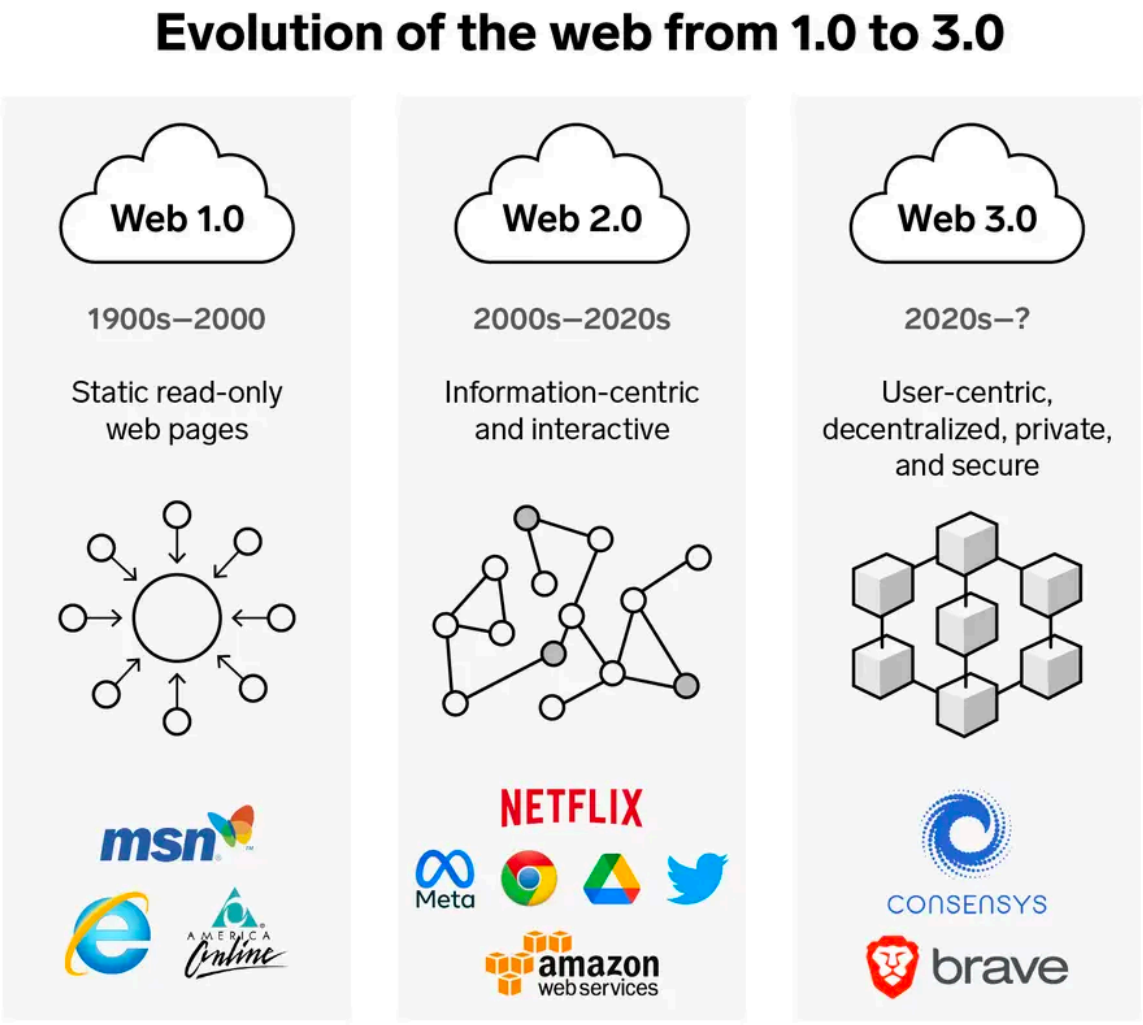
3.2.1 WEB 3.0

Web 3.0 is the next evolution of the World Wide Web. Its main foundations are aiming to be DApps, blockchain technologies, machine learning and artificial intelligence, with the aim of making a more intelligent and adaptive web that improves the user experience [13]. To better understand and explain the importance and the potential shift towards Web 3.0, I will briefly describe the first to iterations.

Web 1.0 was invented in 1989 by Tim Berners-Lee, and he was the computer scientist that first wrote the Hypertext Markup Language, commonly known as HTML, as well as the Hypertext Transfer Protocol (HTTP). Its widespread adoption, however, arrived a few years later with the releases of the first browsers. In 1993 the release of Mosaic (later renamed Netscape Navigator) helped bringing Web 1.0 to the spotlight, with later years seeing the likes of Microsoft, Yahoo or Google join the scene.

Around the 2000 the first mentions of Web 2.0 saw the light of day, aiming to deliver an upgraded and more interactive web, that came to fruition years later with the disruptive social network Facebook. It has not been until the developments of cryptocurrencies and blockchain in general that web 3.0 has become a common buzzword. While Web 2.0 giants like Amazon, Google and Facebook parent Meta grew quickly by collecting and centralizing massive amounts of customer data and monetizing them in the process, Web 3.0's global peer-to-peer (P2P) network could be the great equalizer that makes it that much harder for these tech giants to grow by hoarding data, and gives individuals a higher amount of control over web content and who can access and profit from their personal data [13].

The promises and opportunities of Web 3.0 of higher privacy and transparency make it an attractive technology to be used for application that deal with sensible data. Moreover, Web 3.0’s objective of giving AI and machine learning more importance in delivering content means web 3.0 will be more responsive and intelligent, with its data more logically organized and its organization decentralized to better tailor the experience for different uses and adding security by steering clear from single points of failure.



Source: https://www.businessinsider.com/personal-finance/what-is-web3

Figure 16: Evolution of the web from 1.0 to 3.0

3.3 BLOCKCHAIN NETWORKS

There are multiple types of blockchain networks:

- **Public blockchains**: these are the most well-known, as blockchains like the bitcoin or the Ethereum one are of this type. These networks have no access restrictions, so anyone with an Internet connection can send transactions to the blockchain or become a validator. Public blockchains also include testnets, which are networks for the sole purpose of testing applications. The use of these networks (testnets as well) incurs in costs for the users using them.

- **Private blockchains**: the main difference is that these networks require invitation from administrators to join them. Usually they are referred to as Distributed Ledgers (DLT).

- **Hybrid blockchains**: hybrid chains have a combination of centralization and decentralization, and its exact specifications can vary.

- **Consortium blockchains**: consortium chains are a hybrid of public and private blockchains, where single entities form together a single blockchain network that they all operate and validate. These blockchains are permissioned as well. They are particularly useful in industries where multiple entities work in a single, common goal, like supply chain management.

  Consortium blockchains have an advantage in terms of efficiency and scalability compared to public blockchains. This is primarily due to the smaller number of nodes involved in validating transactions.

- **Sidechains**: these are blockchain ledgers that run Parallelly to a primary one. The entries of the primary and side chains can be linked to one another, but this allows the sidechain to work differently internally (record-keeping methods, consortium algorithms,…).

3.4 CONSENSUS PROTOCOLS

The idea for bitcoin came from the desire to solve the double spending issue. In cryptocurrencies, solving this is essential, as a digital coin that cannot solve it would be deemed useless: a user A could buy assets B and C with the same coin X. When solving the double-spending problem, it needs to be confirmed that A can only buy B or C with X, and if two transactions are made simultaneously, only one can be valid.

The whole benefit of the blockchain is having these transactions that all parts of the system agree upon, making each one unique, without duplicates, ensuring the integrity of the system. To achieve these agreements, a consensus protocol has to be used.

Consensus is a fundamental problem in distributed computing. Systems that are composed by multiple agents have to ensure reliability even if some of the processes are faulty, and they have to do so by reaching consensus. There are multiple protocols and algorithms used in distributed computing to reach consensus effectively, but any consensus protocol must satisfy 3 properties:

- **Termination**: Eventually, every correct process decides some value.

- **Integrity**: If all the correct processes proposed the same value *v*, then any correct process must decide *v*.

- **Agreement**: Every correct process must agree on the same value.

Blockchain is just one of the multiple distributed systems where the consensus problem has to be solved. In the realm of blockchain, the protocols contemplated to solve this issue are as follows: [14]

- **Proof of Work**: one of the first protocols used in blockchain applications. It is based on computing the hash values and validating the transactions until a specified number of trailing zeros are found in the hash value. It was designed for public networks, and uses the computational resources from the systems in the node to reach consensus. The disadvantage is that it is a rather power-hungry protocol that uses a lot of computational power and electricity, making it a not-so-efficient approach, and a less scalable one.

- **Proof of Stake**: this is the protocol used by Ethereum. Instead of computational power, validators are selected according to their economic stake in the network, avoiding centralization and reducing power consumption. Aspiring validators stake their own currency, with the one that staked the most getting the chance to validate the block. Once validated, the rest of validators check the accuracy of the node, and if the majority determines it is accurate, the validator gets their stake back plus a bonus fee, thus rewarding good behavior.

- **Proof of Space**: known as PoSpace, this protocol uses disk storage to validate transactions. Potential validators allocate storage to try and get the node to validate.

- **Proof of Elapsed Time**: developed by Intel Corporation, this protocol is predominantly used in private networks, and the hardware used for this protocol is specially designed for the protocol itself. Each node in the network is assigned a random waiting time, and the first node to complete the randomly chosen period validates the new block, while the rest of processors are put to sleep during that wait time by the specialized hardware.

3.5 BLOCKCHAIN AND VACCINATION CARDS

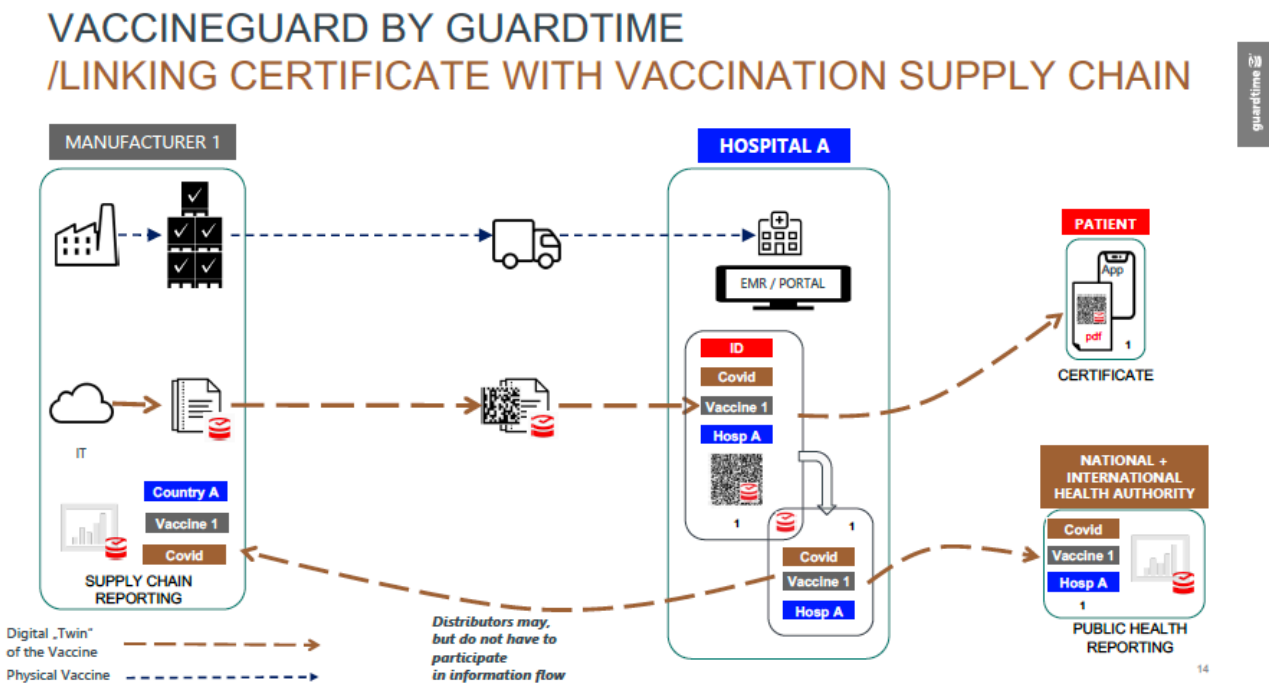
The management of vaccination cards plays a crucial role in public health initiatives. Traditionally, vaccination cards have been issued in physical formats, such as paper-based records, which are prone to loss, damage, and fraudulent activities. Vaccination cards haves also been predominantly managed through centralized systems, where healthcare providers and governmental authorities maintained and updated individual records. The introduction of blockchain can be a paradigm shift, changing to a all-digital approach, and decentralizing the processing of data and records of vaccines applied. Furthermore, in Spain's case, each autonomous region has different vaccinations cards, making transferring from one another an inconvenient task, and worsening traceability all around.

Some cases of blockchain being present in healthcare systems already exist, and these are two examples of it being properly implemented for the management of vaccination records.

3.5.1 ESTONIA

On October 1st, 2020, the Estonian Government confirmed its partnership alongside the World Health Organization for a blockchain-based COVID-19 vaccination certificate solution, that would be led by Estonian blockchain firm Guardtime [15]. But this was not the first time Estonia made use of blockchain technology in their healthcare system.

Estonia's medical services information is 99% digitized, and it is the first country that uses blockchain technology in healthcare. Ever since 2008, Estonia has used 'hash-linked time-stamping', and in 2011, alongside Guardtime, Estonia started working on making health records immutable with the Keyless Signature Infrastructure blockchain technology. This culminated in 2016, when the Estonian e-health foundation launched a blockchain technology to secure the health records of the patients [16].



Source: https://medicalnotes.co/issue-36/

Figure 17: Diagram of functionality of VaccineGuard certificate, developed by Guardtime

3.5.2 VITALPASS

Another really interesting and relevant story for the purposes of our project is that of VitalPass, the vaccination certificate that makes use of Blockchain that was developed and first implemented in Colombia during 2021 [17]. In order to prevent counterfeiting and keep track of vaccines administered, two Colombian companies (AUNA Ideas, a health industry company, and Davivienda, a finance industry one) joined forces and created VitalPass, a digital vaccination certificate built on Blockchain technology with the help of KoiBanx, a token and Blockchain payment platform. The tool itself (VitalPass that is) would work similarly to the apps that were implemented in Europe as a means of checking vaccination records on the go, but VitalPass had the unique factor in that the use of Blockchain granted an extra layer of security and credibility.

VitalPass isn't foolproof, as it still depends on humans (in this case the health workers) to properly and accurately introduce the information regarding the vaccines administered and the patients that got them in the chain, so human error and deception could still be an issue, but that is unfortunately a given, a factor that will always play a part. In May 2022, the European Union acknowledged VitalPass, making these certificates valid in the 27 countries that compose the EU [18].

4 PROJECT DESCRIPTION

4.1 MOTIVATION

Through these first sections of the paper it has been discussed how the healthcare industry could benefit from the use of blockchain, with two examples of countries adopting blockchain and implementing it successfully. It is clear that there is a market for a tool like this, that it is useful and, being an emerging technology that is constantly evolving, that there is still plenty of untapped potential yet to be unearthed. Those are the very reasons why we decided to undertake this problem: we have seen it can be done, we can determine the benefits it provides, so the clear next step is to expand it and develop a tool that allows us to make use of the features of blockchain with vaccination cards and records, without limiting it to COVID-19 vaccination certificates.

The various benefits that blockchain provides would be largely useful for the task at hand: decentralizing the system to guarantee a higher degree of privacy, giving the users a higher degree of ownership of its data, and guaranteeing transparency, traceability and immutability of the data, the two latter ones being two of the common buzzwords that best serve our particular example.

The concept of traceability refers to the ability to track and verify the origin, movement, and handling of items across a network of participants. In the context of the vaccine industry, this could involve tracing the entire supply chain of vaccines, from production to administration, ensuring transparency and accountability at every stage. By recording each transaction and event on a blockchain, stakeholders can have real-time visibility into the movement of vaccines, reducing the risks of counterfeit or substandard products.

Immutability, ensures that once information is recorded on the blockchain, it cannot be altered or tampered with retroactively. In the context of vaccination cards, this means that the recorded data, such as vaccination dates, types of vaccines administered, and associated information, remains secure and trustworthy. This immutability instills confidence in the integrity of vaccination records, preventing

unauthorized modifications and providing a reliable source of information for individuals, healthcare providers, and authorities.

Overall, adopting blockchain technology offers several key benefits in biomedical and health care applications, such as decentralized management, immutable audit trail, data provenance, robustness/availability, and security/privacy. [19]

With all of this in mind, it becomes apparent that blockchain and the healthcare system are a match made in heaven: all the benefits and features provided by its implementation would benefit the system and improve it in significant ways.

4.2 MAIN OBJECTIVES

- **Create a user-friendly interface to access vaccination records easily:**

The main goal of the project is to create an easy-to-use tool that lets the user access vaccination records on their favorite device. The app has to be easy to understand and navigate and the information presented has to be well presented.

- **Implement a secure, immutable and traceable system:**

The information we are storing is highly sensitive, and as discussed in previous sections, it is imperative that these records and data can be accessible whenever we need, and that the information itself is accurate and impossible to modify once it is in the blockchain.

- **Explore the possibilities and potential of blockchain technology:**

Finally, this is an ever-growing technology, and through the development of this project we also aimed to discover and understand new applications that such a powerful technology could have in this or other fields.

4.3 METHODOLOGY

The timeline followed for this project is included below. In terms of the timing of this project, the first objective was understanding and familiarizing ourselves with blockchain technology and how we want to implement it for our particular case of use. This will be the backbone of the program so it was imperative to get it working first, and getting familiar with it before going further into the development process. This involved learning a brand new technology, understanding its ins and outs, how security can work, or how these parts all connect to one another. This also involved familiarizing with Solidity, Ganache and the Ethereum network, Truffle and Remix in order to implement everything blockchain related. After that, the smart contract was developed, which was the crux of the application. Finally, the work on the user and admin interfaces could be started, alongside web3.js implementation to get all the parts of the system to interact properly.

	December	January	February	March	April	May	June
Research & Learning							
Project Structure							
Coding & Design							
Testing							
Report Wrtiting							

Table 1: GANT diagram of project development timeline

The development process, the longest one, is divided in various sections: smart contract, application back-end, mobile-based user UI, PC-based admin UI and web3.js implementation. In regards to the work ethic and methodology used to develop the project, we worked based on a mix between waterfall model (Analysis, Design, Coding, Testing) and an Agile framework in the development process, aiming to keep track of the progress during the development process and having the possibility to revert changes earlier in the development pipeline.

4.4 ECONOMICAL ESTIMATIONS

In the following section we will carry out the calculations of the development and maintenance costs of the project, from the material costs to the human costs. The projected profits and/or losses will also be calculated, in the hypothetic scenario where the project is commercialized. This projection will contemplate two cases: one rather optimistic one, and another one in the pessimistic side, to better reflect the range of possibilities that could happen.

4.4.1 MATERIAL RESOURCES

The material resources for this project were two PCs and two monitors, as the project was carried out between two different countries, so the material used changed halfway. The specifications for all material used is included in the tables below.

Laptop 1: Macbook Pro 13" 2017 i5	
Processor	2.3GHz dual-core Intel Core i5, Turbo Boost up to 3.6GHz, with 64MB of eDRAM
Operating system	macOS Ventura 13.1
Graphics Card	Intel Iris Plus Graphics 640
Display	Retina, 13.3-inch IPS 2560x1600 pixels
Memory	8GB of 2133MHz LPDDR3 onboard memory
Disk Storage	128GB SSD
Price	1399 €

Table 2: Specifications of first laptop used

Laptop 2: Macbook Air M2 13" 2022	
Processor	Apple M2 8-core CPU, 8-core GPU, 16-core Neural Engine
Operating system	macOS Ventura 13.4
Graphics Card	Integrated in M2 chip
Display	Liquid Retina, 13.6-inch IPS 2560x1664
Memory	16GB unified memory
Disk Storage	256GB SSD
Price	1262,54 €

Table 3: Specifications of second laptop used

Monitor 1: Acer KB2 (Refurbished)	
Screen Size	23.8″
Resolution	Full HD 1920x1080
Panel Tech	IPS AMD Free-sync
Audio	-
Price	92,60 €

Table 4: Specifications of first monitor used

Monitor 2: Samsung U28E570	
Screen Size	28″
Resolution	4K UHD 3840x2160
Panel Tech	TN, W-LED, AMD Free-sync
Audio	-
Price	180 €

Table 5: Specifications of second monitor used

The first computer has had an approximate lifespan of 1607 days, from the day it was bought to the day it has been substituted by the newest one. This equals 38.568 hours of lifespan, where we have to account for hours of work itself, as the computer has been used for other purposes. In terms of the new one, the lifespan of devices such as laptops according to Agencia Tributaria is 5 years, or 43.800 hours. In the case of the monitors, Monitor 1 has been used for approximately 9 months, or 6480 hours, and Monitor 2 can be considered to have a lifespan of 6 years, or 52.560 hours. Taking that into account, the estimate of hours worked in the project is as follows:

Activity	Hours Dedicated
Planning and Meetings	10
Analysis and Research	100
Design	50
Development	300
Testing	
Documentation	250
Total	710

Table 6: Breakdown of hours dedicated to the project

The division of hours worked is roughly as follows: with set of equipment 1 (Laptop 1 & Monitor 1) 245 hours of the total hours were completed, which equates to the whole analysis and research workload and a share of the rest of activities' hours dedicated. The rest of the total hours were fulfilled with the set of equipment 2, which adds up to 465 hours.

$$\text{Hourly Cost (Laptop 1)} = \frac{1399€}{38.568} \approx 0{,}03627€/\text{hour}$$
$$\text{Total Cost (Laptop 1)} = (0{,}03627€/\text{hour}) \times 245 \approx 8{,}89€$$

Ecuation 1: Total cost of laptop 1 for this project's development

$$\text{Hourly Cost (Laptop 2)} = \frac{1262{,}54€}{43.800} \approx 0{,}028825€/\text{hour}$$
$$\text{Total Cost (Laptop 2)} = (0{,}028825€/\text{hour}) \times 465 \approx 13{,}40€$$

Ecuation 2: Total cost of laptop 2 for this project's development

$$\text{Hourly Cost (Monitor 1)} = \frac{92{,}60€}{6480} \approx 0{,}0143€/\text{hour}$$
$$\text{Total Cost (Monitor 1)} = (0{,}0143€/\text{hour}) \times 245 \approx 3{,}50€$$

Ecuation 3: Total cost of monitor 1 for this project's development

$$\text{Hourly Cost (Monitor 2)} = \frac{180€}{52.560} \approx 0{,}003425€/\text{hour}$$
$$\text{Total Cost (Monitor 2)} = (0{,}003425€/\text{hour}) \times 465 \approx 1{,}59€$$

Ecuation 4: Total cost of monitor 2 for this project's development

Software costs are not included, as all software used is open-source and free, or bundled with the laptops.

In terms of maintenance costs, these would be divided by the pricing of hosting the application, releasing it on the App Store and Google Play, and deploying the smart contract in the Ethereum public network. The price of hosting this app was estimated with Heroku pricing. An application like this would require high availability, handling of high volumes of concurrent requests and low latency, thus requiring an advanced package, which can be estimated to be around 250€ per month. The release of the app (for the users) would entail a 25€ one-time payment for the Google Play store, and an annual 99€ fee for the Apple Store. Finally, the deployment of the smart contract has an estimated gas usage of 2.711.701, which would be equivalent to 116,39€ to be paid at the time of deployment.

In the table below includes the material cost breakdown:

Type	Section	Cost	
		One-Time	Annual
Development	Equipment	27,38 €	
	Smart Contract Deployment	116,39 €	
Maintenance	Hosting		3.000 €
	App Release	25 €	99 €
Total		168,77 €	3.099 €

Table 7: Breakdown of total material cost for development and further maintenance

4.4.2 HUMAN RESOURCES

Human Resources come down to the value of the work hours of one software developer. To estimate this, it is necessary to take a look at the average salary of a software developer in Spain, which according to Indeed is approximately 34.043,33€ annualy. This was calculated as an average of multiple software developer-related positions. Project development ran for 6 months, so total cost is:

$$\text{Total Development Human Cost} = \frac{34.043{,}44€/\text{year}}{6 \text{ months}} \approx 17.021{,}67€/\text{hour}$$

Ecuation 5: Total human cost of development for this project

On the other hand, maintenance costs can vary depending on the option chosen to carry out the task. The more realistic and reasonable one would be having a software developer to focus on the maintenance, possible issues or improvements to the app. Annual cost for maintenance then would be the annual salary of a software developer, therefore 34.043,33€.

4.4.3 TOTAL COST

Following the estimations of material and human costs done, the total estimations and projected profits or losses can be gauged. This application aims to give support to healthcare organizations and systems of countries, and therefore the main goal is to sell the application and its support and maintenance to interested parties. With that objective in mind, total costs will be estimated, to establish the break-even cost so that the value to start achieving profits can be known. Total costs, adding all previous values found, would be a fixed cost of 17.190,44€ and an annual cost of 37.142,33€.

The endgame for this application would be selling it to governments, so that these countries can implement this tool to manage their vaccination record system. For this estimation, Spain will be the focus, and 3 different scenarios will be devised. These range from an optimistic one, to a optimistic and more realistic one, to finally a pessimistic one.

4.4.3.1 SCENARIO A

In this scenario all 17 Autonomous Regions (AR) of Spain adopt this tool to manage their vaccination card systems. Each record upload costs 238718 gas, or about 7,33€, and it will be assumed that cost of uploading records will be taken care of by the government on a per-use basis, meaning they will pay for the services of the application, and each month the total cost of gas used by uploaded records will be calculated and paid by them. Therefore, in the calculation costs we will only estimate

profits and losses based on the cost of development and maintenance, and the earnings from governments paying for the license and use of the application.

Scenario A would be the most optimistic, where as mentioned every single AR adopts the application to manage their vaccination cards and records. Payments for the license will be taken into the estimation as if they were annual, even if the license itself is paid for periods of 2 or more years at a time. Cost per AR would vary, as number of residents would be taken into account: number of residents and cost of license would be directly proportional. For all 17 ARs, we will gauge the average cost per year of the license to 40.000€. With that in mind, cost for the first and subsequent years would be as follows:

Scenario A		
Concept	**1st year**	**Subsequent years**
Development Costs	17.190,44 €	-
Maintenance Costs	37.142,33 €	37.142,33 €
Licenses Sold	680.000 €	680.000 €
Total Revenue	625.667,23 €	642.857,67 €

Table 8: Breakdown of total revenue per year in an optimistic scenario

These profits could open up the possibility of signing a small team of developers to better assist with maintenance across the country.

4.4.3.2 SCENARIO B

Scenario B is a more realistic approach to the estimations, where 15 of the 17 ARs adopt the system (all except Cataluña and País Vasco), and the annual cost of the license averages 25.000€.

Scenario B		
Concept	**1st year**	**Subsequent years**
Development Costs	17.190,44 €	-
Maintenance Costs	37.142,33 €	37.142,33 €
Licenses Sold	375.000 €	375.000 €
Total Revenue	320.667,23 €	337.857,67 €

Table 9: Breakdown of total revenue per year in a realistic and positive scenario

This scenario is still positive and opens up the possibility of setting up the small development and maintenance team as well.

4.4.3.2 SCENARIO C

This is a pessimistic scenario, where only Madrid adopts this system. Because of the amount of residents of Madrid, the annual cost of license could be estimated to approximately 50.000€.

Scenario C		
Concept	**1st year**	**Subsequent years**
Development Costs	17.190,44 €	-
Maintenance Costs	37.142,33 €	37.142,33 €
Licenses Sold	50.000 €	50.000 €
Total Revenue	4.332,77 €	12.857,67 €

Table 10: Breakdown of total revenue per year in a pessimistic scenario

With this estimated cost of license, we would incur in loses for the first year when taking into account development costs, but each subsequent year would net profits, recovering those loses already in the second year.

5 SYSTEM DEVELOPMENT

This chapter focuses on the development process of the vaccination record management system, encompassing various aspects such as model analysis, design, and implementation. The system development phase plays a pivotal role in transforming the conceptual ideas into a functional and practical solution.

Model Analysis section will delve into the requirements of the system from a user and administrator perspective, while the Design section will explain in further detail the structure and components of the system, providing reasoning behind the decision-making done in the process. Finally, the Implementation section focuses on the three pillars that make the actual application work: the smart contract, the API, and the website and app themselves.

5.1 MODEL ANALYSIS

The design of the vaccination record management system is driven by the need to provide users with easy access to their records, regardless of their location. Additionally, the system aims to enhance convenience by allowing users to view their future appointments. For administrators, the system strives to provide seamless interaction with the blockchain while enabling them to efficiently add, view, and update vaccination records. The underlying goal of the system is to replace traditional paper-based vaccination records and cards, while also promoting the standardization and homogenization of vaccination records across territories.

The user-centric design of the system ensures that individuals can effortlessly view their vaccination records. With the app, users can easily access their records from anywhere, at any time, using a secure and user-friendly interface. This accessibility allows users to have a comprehensive overview of their vaccination history, facilitating important decisions related to healthcare and travel requirements. Furthermore, the system's ability to display future appointments adds an extra layer of convenience for users. By providing information about upcoming vaccinations or appointments, users can effectively plan their schedules and make necessary preparations. This feature not only enhances the user experience but also encourages proactive engagement in healthcare management.

From an administrator's perspective, the system streamlines the process of managing vaccination records. Administrators can efficiently add new records, view existing records, and update information as required. The seamless integration with the blockchain technology ensures data integrity and security, while minimizing administrative burdens. By leveraging the capabilities of the blockchain, administrators can confidently manage vaccination records, knowing that the information is tamper-proof and transparent.

5.1.1 USER STORIES

User stories defined the requirements that each actor of the system has: they detail what each user expects from the application, what actions they want to perform and the goals they try to achieve through the use of the system. The four actors defined are:

- Users: the regular population that will use the app and have their records stored in the blockchain.
- Hospitals: the hospital themselves, that will be part of the implemented system.
- Doctors: the hospital workers that will interact directly with the system
- Healthcare System: the actual healthcare system, that manages all hospitals of their territory.

The user stories developed are the following:

- As a user, I want to be able to store and access my records on the go.
- As a user, I want to have my records digitally and securely stored, instead of relying on paper-based cards.
- As a user, I want to be able to see future appointments I might have.
- As a user, I want to be able to move to another region and avoid the hassle that exists nowadays when transferring vaccination cards.

- As a hospital, I want to be able to store all records in a secure, immutable manner, without relying on paper-based records.

- As a hospital, I want a system that makes the transfer of records between regions more convenient.

- As a doctor, I want to have a streamlined system, easy-to-use, to manage all things related with records.

- As a doctor, I want to add records for a patient.

- As a doctor, I want to view all records for a particular patient, filtering them by various attributes.

- As a doctor, I want to be able to generate future appointments.

- As a doctor, I want to be able to update records, or mark future appointments as completed.

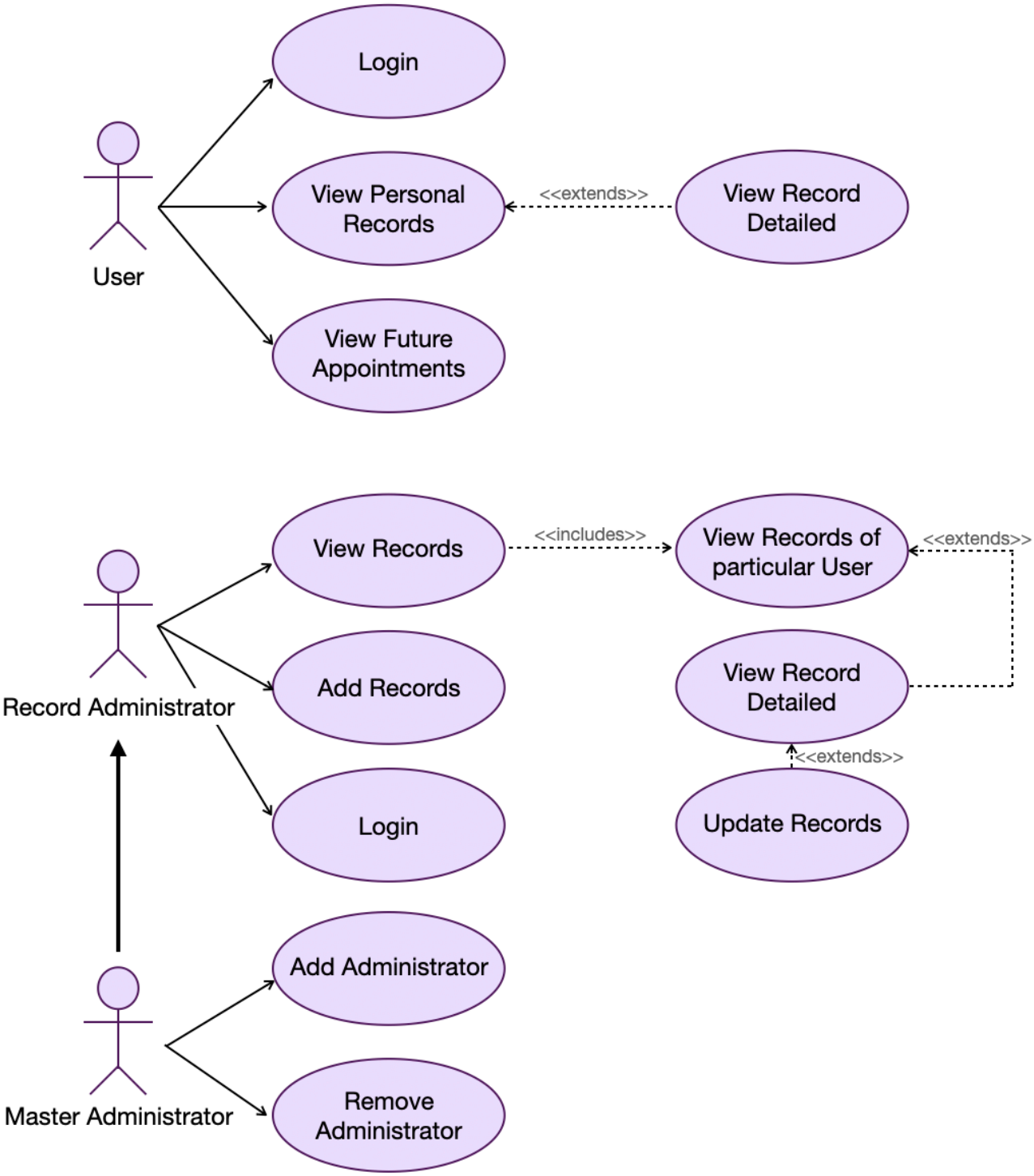
- As the healthcare system, I want to be able to manage the hospitals that participate in the system.

5.1.2 USE CASES

With user stories already defined in the previous section, the diagram of uses cases can be defined. These diagram presents an overview of the different actors (or types of users of the application) and the actions they can perform in the application itself. This diagram is shown in Figure 18.

The three actors defined are Users, Record Administrators and Master Administrators. Users are the regular people that use the app to track their own records or their future appointments. Record Administrators are the health workers that use the app. They can add and view, and the latter offers various options for filtering results, expand details and updating the records themselves. Adding records also offers the

option to add follow-up appointments for vaccine that require multiple doses. Master Administrators could be the healthcare systems of the cities or countries that implement the system. They can perform all actions a Record Administrator can, but also are in charge of adding or removing Record Administrators.



Source: Own elaboration

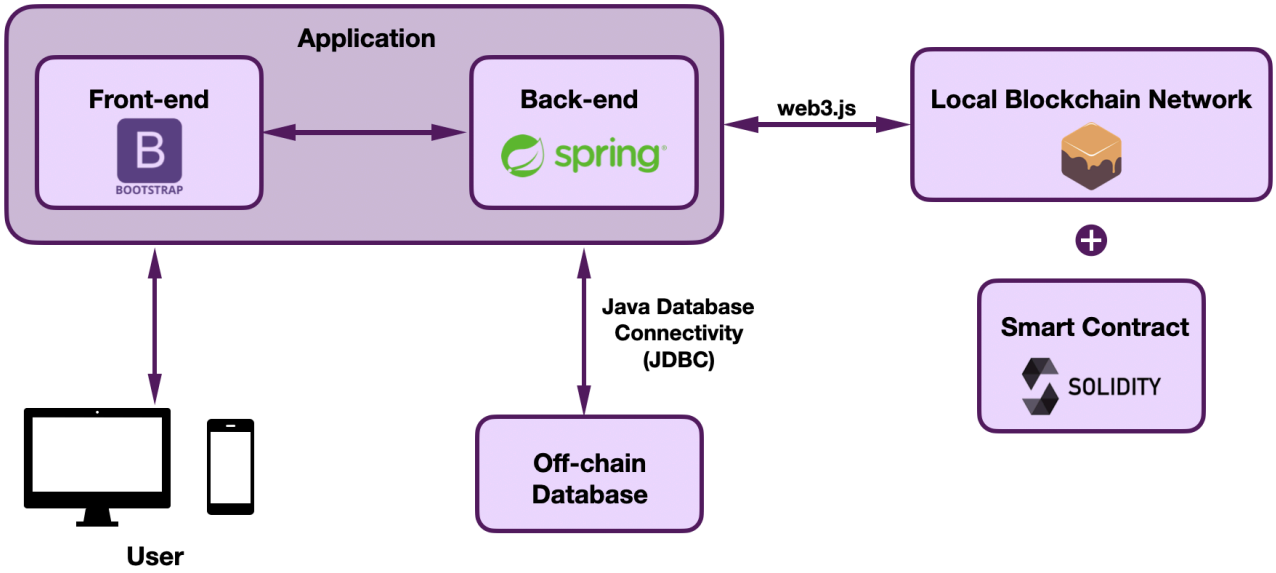
Figure 18: Use cases diagram of the project

5.2 DESIGN

This section will focus on the design choices made for the project, ranging from its architecture and its data structures to its functionalities and interaction with the blockchain network. The various subsection will offer insight into the why and the how, delving into the insides of the app: its structure, its database and its main functions.

5.2.1 ARCHITECTURE DESIGN

Below it is included a figure that illustrates the architecture of the system developed in this project:



Source: Own elaboration

Figure 19: Architectural diagram of the project

The system has the application as its core. The application is divided by two: its front-end, which is the visible part with which users (via mobile) and administrators (via PC) interact with the system. This front-end uses Bootstrap frameworks for its design, and is written in HTML and CSS. The back-end is the engine and heart of the application, it is what makes it work, and it is written in Java using the Spring Boot framework. This framework is useful as it standardizes web and micro-service creation and facilitates developers the creation of modular applications [20]. The application will be connected to two databases: an off-chain database, implemented with JDBC and containing login information and basic hospital and vaccine data; and the on-chain

database, containing all the vaccination records. The on-chain database is implemented via Ganache in a local private Ethereum network to test the functionality of the application without incurring in real-world monetary costs, as discussed previously. The interaction between the application and the private network is done with the web3 javascript library, which is in turn able to connect with the smart contract written in Solidity. This single contract contains all the functions needed for the use cases that the application contemplates, and the web3 library allows sending and receiving of data between the back-end and the smart contract, so that the users interacting with the front-end can upload or view the records themselves.

5.2.1.2 SPRING BOOT FRAMEWORK

Spring Boot is a powerful Java-based framework that simplifies the development of robust and scalable web applications. It is built on top of the popular Spring Framework and provides a streamlined approach to building standalone, production-grade applications.

For this project, this framework is divided in four main pillars that define how the application interacts with the API and, therefore, with the off-chain database. These parts are:

- **Service**: The service layer encapsulates the business logic of the application. It acts as an intermediary between the controller and the implement layers. The service layer defines all functions to be implemented for a particular part of the application. They are interfaces annotated with @Service, and thus they do not contain the actual implementation of these functions.

```
@Service
public interface HospitalService {

    List<HospitalDTO> getHospitals();

    HospitalDTO getHospById(Long id_hosp);

    HospitalDTO getHospByName(String name);

}
```

- **Repository**: The repository layer provides an abstraction for data access and persistence. It deals with data storage, retrieval, and manipulation. In the case of this project, repositories only extend the CrudRepository class from Java, which allows us to perform some of the searches in the database with already implemented an existent functions, like findById(), which finds the entry in a particular table by the variable annotated with @Id in the table declaration.

- **Implementation (Impl)**: The implementation layer contains the actual implementation of the functions defined in the service layer. By convention, the implementation classes are named with a suffix of "Impl" to indicate that they provide the concrete implementation of the corresponding interface.

```
@Component
public class HospitalImpl implements HospitalService {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Autowired
    private HospitalRepository hospRepo;

    @Override
    public List<HospitalDTO> getHospitals(){
        return StreamSupport.stream(hospRepo.findAll().spliterator(),false)
                .map(obj -> new HospitalDTO(
                        obj.getId(),
                        obj.getAddress(),
                        obj.getName(),
                        obj.getLocation(),
                        obj.getCity()))
                .toList();
    }

    @Override
    public HospitalDTO getHospById(Long id_hosp){
        Optional<HospitalTable> opt_hosp = hospRepo.findById(id_hosp);
        HospitalTable table = opt_hosp.get();
        HospitalDTO hospital = new
HospitalDTO(table.getId(),table.getAddress(),table.getName(),table.getLocation(),table.get
City());

        return hospital;
    }

    @Override
    public HospitalDTO getHospByName(String name){
        name = name.replace('-',' ');
        String query =
                """
                   SELECT *
                   FROM HOSPITAL
                   WHERE NAME = '""" +name+"'";
        List<HospitalDTO> joinList = jdbcTemplate.query(
```

```
                query,
                (rs,rowNum) ->
                        new HospitalDTO(rs.getLong("HOSPITAL_ID"),
rs.getString("ADDRESS"), rs.getString("NAME"), rs.getString("LOCATION"),
rs.getString("CITY")));
        return joinList.get(0);
    }
}
```

- **Controller**: The controller layer is responsible for handling incoming requests, processing them, and returning appropriate responses. It acts as the entry point to the application and plays a crucial role in defining the API endpoints. Controllers are typically annotated with @Controller or @RestController, the latter in this case, and the API endpoints are defined by @RequestMapping("/api"), which makes all the endpoints that will be defined in that controller to start with /api, plus the corresponding mapping that is determined per function. Each function is tagged by its HTTP method (GET, POST, DELETE...), as seen in the code below. Controllers instantiate an Impl class to call the functions needed for each defined endpoint.

```
@RestController
@RequestMapping("/api")
public class HospitalController {

    @Autowired
    private HospitalImpl hospitalService;

    @GetMapping("/hospitals")
    public ResponseEntity<List<HospitalDTO>> getHospitals(){

        var hospitals = hospitalService.getHospitals();

        return ResponseEntity.ok().body(hospitals);
    }

    @GetMapping("/hospitals/{id}")
    public ResponseEntity<HospitalDTO> getHospById(@PathVariable("id") Long id){
        var hospital = hospitalService.getHospById(id);
        return ResponseEntity.ok().body(hospital);
    }

    @GetMapping("/hospitals/name/{name}")
    public ResponseEntity<HospitalDTO> getHospByName(@PathVariable("name") String name){
        var hospital = hospitalService.getHospByName(name);
        return ResponseEntity.ok().body(hospital);
    }
}
```

By separating the application into these four distinct parts, Spring Boot promotes a modular and organized approach to application development. It allows for better

maintainability, testability, and flexibility in the application architecture, making it easier to evolve and scale the application as needed.

To properly implement the endpoints and interactions with the database, two more elements are needed. The **model** package includes Table files, which are the declaration of the tables created in the database. These files also include all the get functions for the multiple variables. Finally, **DTOs** are Java Records that define the objects that all functions implmented in Impl will return, and in the application's multiple javascript files. An example of a DTO is included below.

```
public record HospitalDTO (
        Long id_hosp,
        String address,
        String name,
        String location,
        String city
){
}
```

5.2.2 DATA STRUCTURES

In this section we will delve into the overall data structures used in the project, divided in the off-chain database and blockchain structures.

5.2.2.1 OFF-CHAIN DATABASE

As previously discussed, the off-chain database contains all information that will be used for the application except from the vaccination records themselves, which are the more sensitive pieces of data, and therefore the ones stored in the blockchain.

Figure 20 contains the database structure, which has been implemented using JDBC as mentioned in Section 2.3.5. The tables included are:

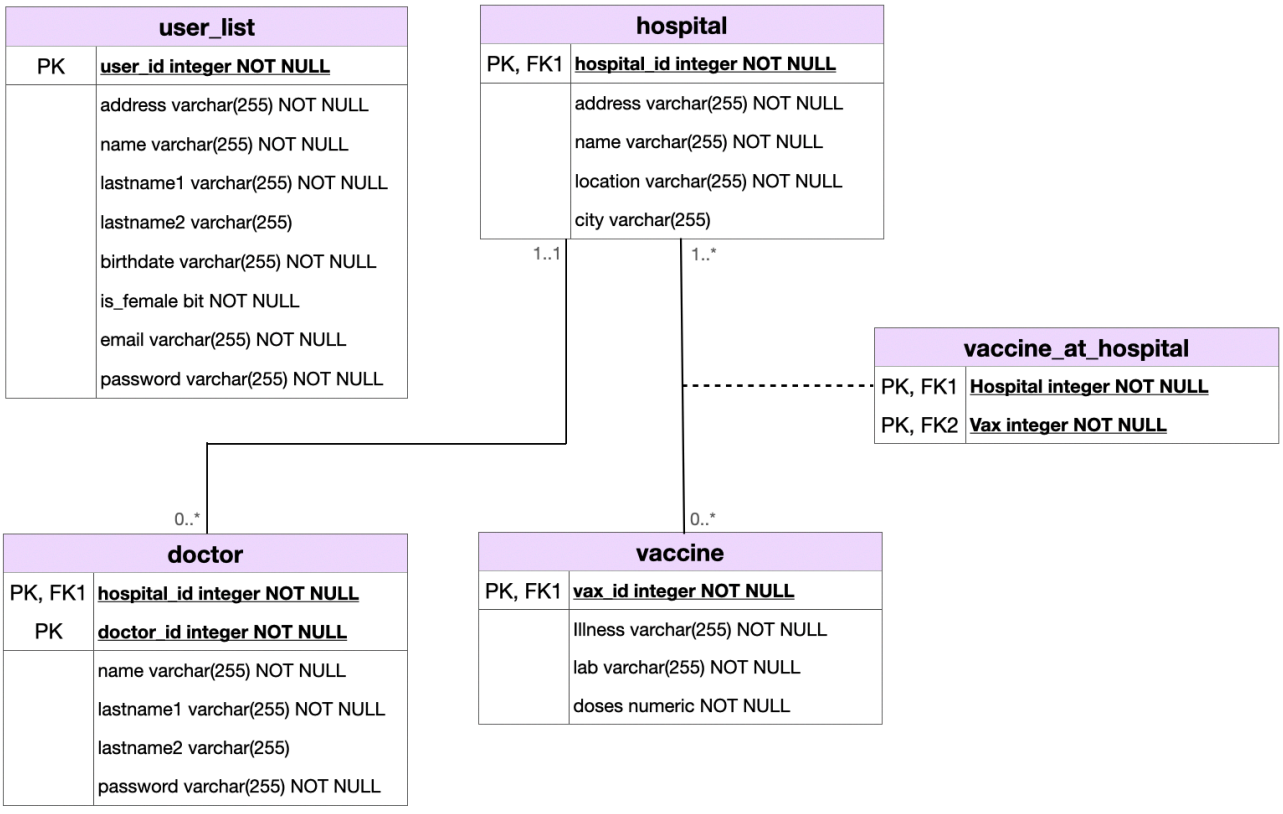
- **user_list**: this table contains all user data apart from their records. The data includes their full name, their date of birth, their gender and the address of the user's blockchain account. It also contains the login information of the user, with an email and hashed password.

- **doctor**: this table contains basic information for each doctor, as well as their hashed password. The primary key is a combination of a doctor id and the id of the hospital they work in.

- **hospital:** this table contains basic information about each hospital, including its unique identifier in the system, the address of the Ethereum network account and its name, name of the address and city it is located in.

- **vaccine**: this table contains the general information for each vaccine. The data it stores is common among vaccines and serves as its identifier. For example, lot number is not included, as a vaccine for the same illness and from the same lab could have multiple different lot numbers.

- **vaccine_at_hospital**: this table relates the **hospital** and **vaccine** tables. It just stores the id of each.



Source: Own elaboration

Figure 20: Entity-relation diagram of the off-chain database

The relations between tables are quite straightforward. Each Doctor will belong to 1 Hospital, and each Hospital can have any number of Doctors, including 0 if the account has been registered but no doctors are in the system yet. Vaccines belong to 1 or multiple hospitals, as they are stored in the database if there is at least 1 Hospital that offers them, and one Hospital can have any number of vaccines, from 0 to multiple. Users are isolated from the rest of the system.

As a side note, all passwords were stored hashed. The regular, plain text passwords were salted first, which is a process where a cryptographically secure random string is added to a password before it's hashed, making it exponentially difficult for an attacker to know the exact original password. After that they are stored in the database, and password validation in the application uses the hashed versions.

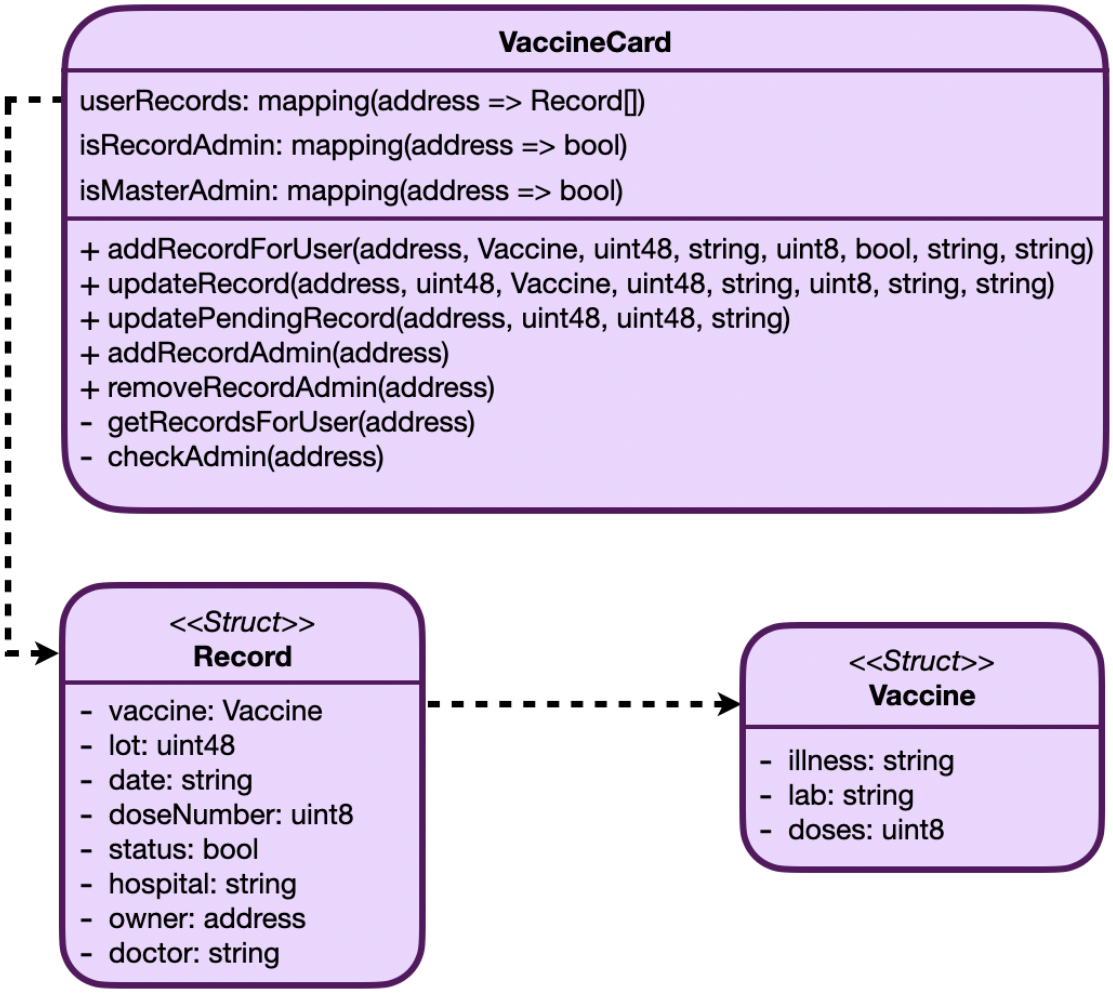
5.2.2.2 BLOCKCHAIN

As detailed in the architecture design section, the functionality to interact with the local private network set up in Ganache is contained in a single smart contract. The details of the contract are illustrated in the class diagram of Figure 21.

The contract named VaccineCard contains all functions and structures needed to interact with the on-chain database that contains all records. VaccineCard class defines 3 types of mappings, which are similar to variables that are attached in this case to each account of the local Ethereum network. This mapping is done to the addresses of each account. Variable userRecords assigns a Record (a custom structure that will be explained shortly) array to each address, which will be used to store all the records associated to each user. The other 2 mapped variables are used to handle the permissions for the functions. All hospital accounts will be defined in the scope of the smart contract as Record Administrators, and variable isRecordAdmin maps each address to a boolean value, that is true if the address belongs to a hospital account. To manage the assignment or removal of Record Administrators, the last mapped variable isMasterAdmin designates another mapped boolean to each address. Master Administrators, as illustrated in the use case diagram, have the same options as Record Administrators, but can also remove or add Record Administrators.

The smart contract includes two defined structs: `Record` and `Vaccine`. The `Record` struct represents a vaccination record and contains various fields such as the vaccine details, lot number, date the vaccine was administered, dose number, status (whether the vaccine has been administered, or is pending), hospital where it was administered, user's address, and the doctor who administered the vaccine. The `Vaccine` struct represents the details of a specific vaccine and includes fields for the illness it targets, the lab that produced it, and the number of doses required.

The constructor for the smart contract defines the message sender, in other words the address that deploys the smart contract in the local private network, as a Master Administrator, and defines 3 of the 10 accounts created in the network as hospitals by setting their isRecordAdmin mapped variable as true.



Source: Own elaboration

Figure 21: Class diagram of VaccineCard.sol smart contract

The contract also defines two modifiers: onlyMasterAdmin and onlyRecordAdmin. These modifiers are used to restrict access to certain functions based on the caller's permissions. The onlyMasterAdmin modifier requires the account that executes the call to be a Master Administrator, and the onlyRecordAdmin modifier requires either Master or Record Administrators to perform the call.

Finally, The contract provides various functions to manage vaccination records and administration permissions. These functions are defined in the diagram, but will be explained in further detail in section 5.3.1.

5.2.3 SEQUENCE DIAGRAM

In order to better illustrate the workflow and functionality of the application, we will delve into two of the main cases of use by explaining in great detail two sequence diagrams that explain the flow of each case of use in depth. The first case covered is the action performed by an Record Administrator to add a record for a certain user. The sequence diagram itself is included below in Figure 22.

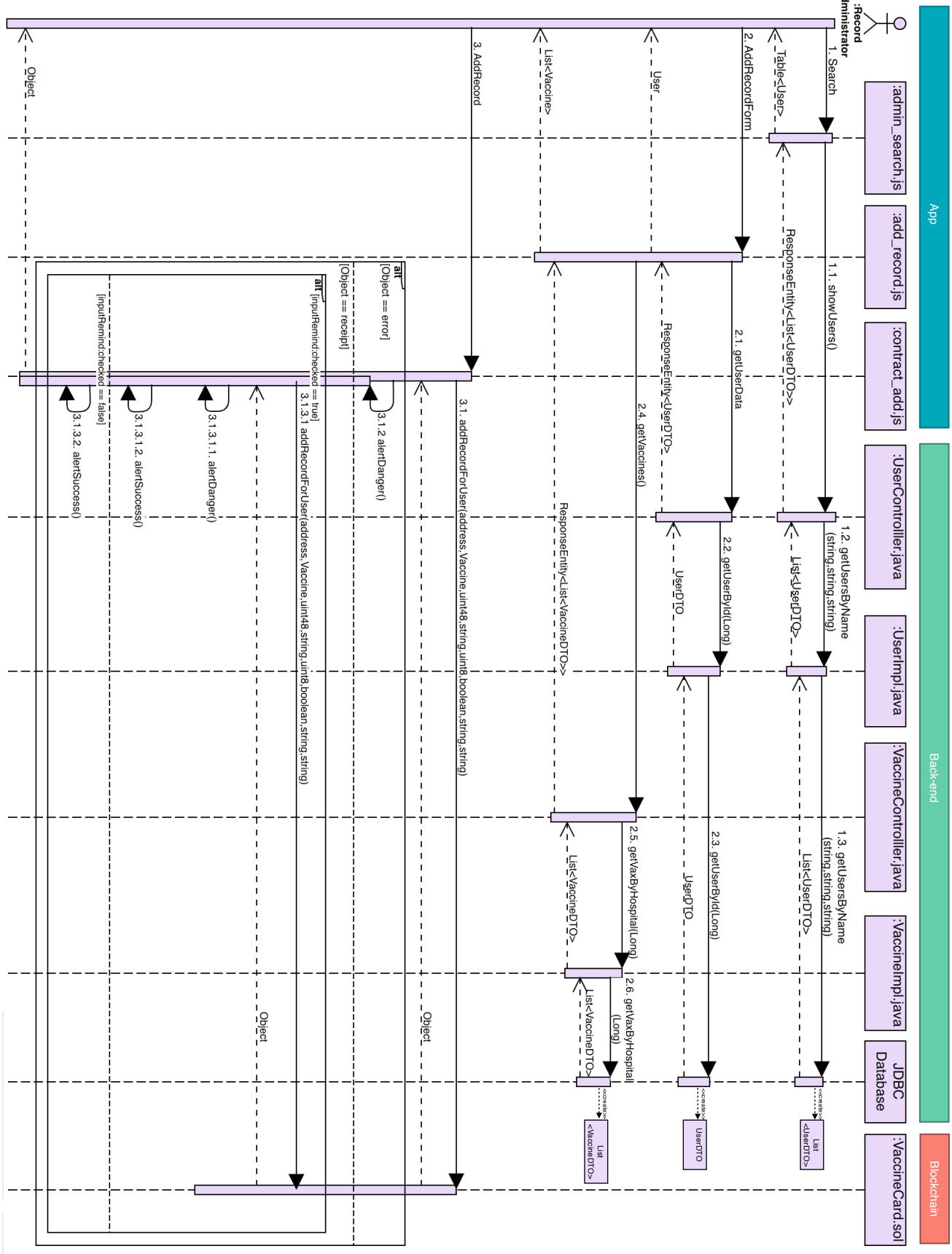
For this case, the prerequisites not considered in the overall sequence are that:

- The Record Administrator is already logged in

- The user list is non-empty, and the user searched for exists in the database.

With those requisites accounted for, the sequence starts with the Record Administrator filling the search bar with the name and/or last names of the user they are wishing to add a record for. The petition goes from the app's javascript to UserController through the endpoint defined in it, in this case 'api/users/{name}/{lastname1}/{lastname2}. The detailed explanation of these endpoints and the API itself are included in Section 5.3.2. UserController instantiates UserImpl which has, as detailed in Section 5.2.1.2, the proper implementation of these methods that interact with the off-chain database. The database returns a list of users that fit the search criteria, and they are shown to the Record Administrator (RA) in a table.

Once the RA chooses the user he wants to add the record to, the website directs them to a page with the appropriate form to be filled. During the loading of the



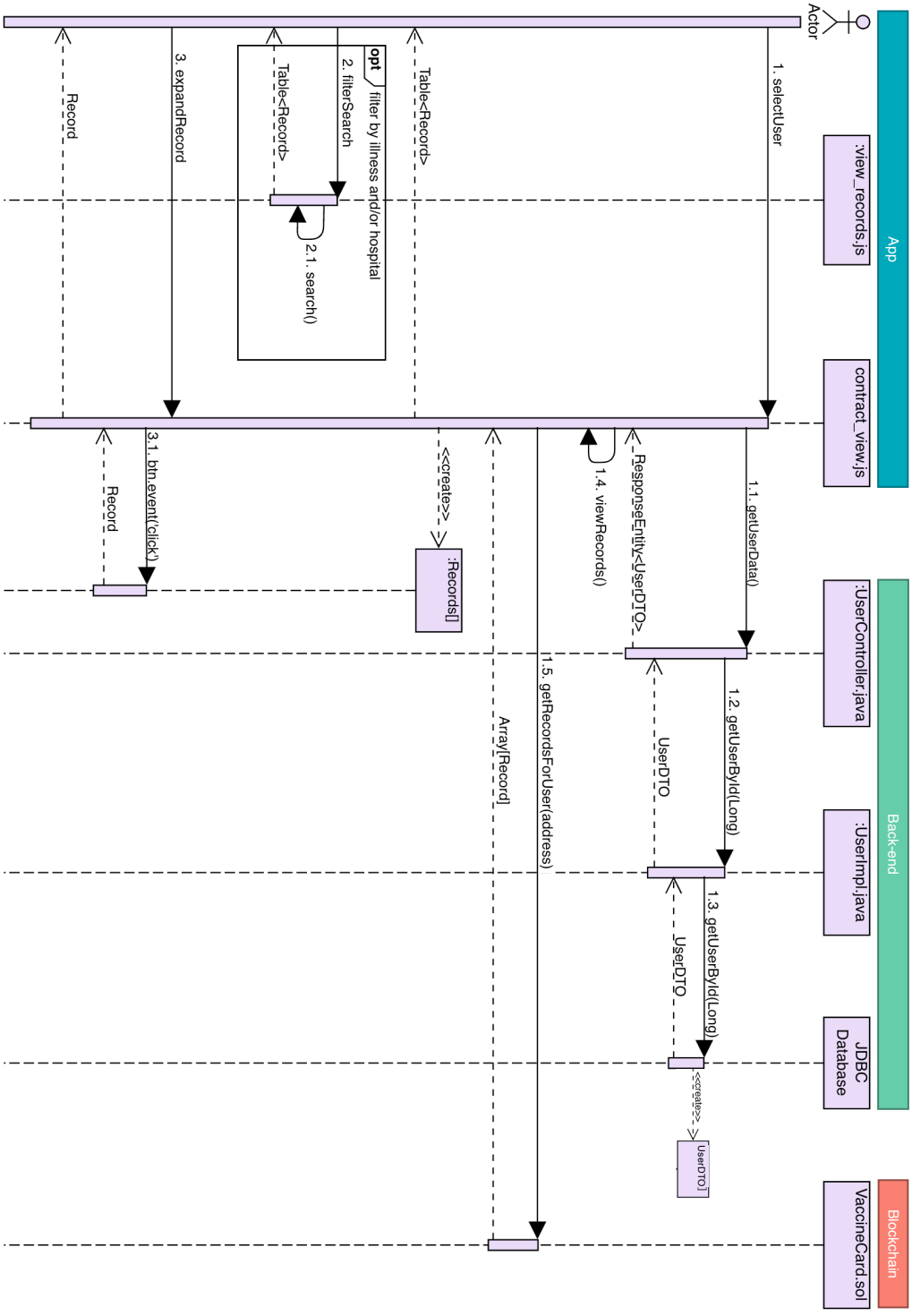
Source: Own elaboration

Figure 22: Sequence diagram of the case of use of adding a record as an administrator

HTML document, two more calls are performed: getUserData retrieves all the user's information needed to be stored in the blockchain, and getVaccines() makes use of VaccineController and VaccineImpl to get all vaccines available in the hospital that the RA works in. Following that, the RA can fill the information in the form and submit it, which will make use of the web3.js library to directly connect to the smart contract, and thus with the local blockchain network. Every transaction with the network will return a response Object that can be either a 'receipt', indicating the transaction was successful and returning information such as the transaction hash, the block hash and number, the gas used or the addresses of the sender and the contract, or an 'error', which contains the cause of the error.

Errors and receipts are reflected in the UI by alerts, that are generated in contract_add.js depending on the response received after executing the transaction. The sequence ends if the object received is an 'error' and shows it to the RA, but if the transaction is successful we contemplate two more cases. When adding a record, the RA has the possibility of 'generating a reminder'. This option is used for vaccines that require more than 1 dose to complete the vaccination, and allows the RA to add, alongside the original record, a "future" record for a future appointment. This record is tagged as not administered through a variable named 'status', and the RA only has to add the date for the appointment; the vaccine will be the same as the one added for the first record and the dose number will be +1 of the current one. In case the option to generate the reminder is unchecked by the RA, the sequence ends on a success alert shown to them, but if checked, the process repeats once more, adding a second record, and performing the same actions depending on whether the response returned is a 'receipt' or an 'error'.

The second sequence shown is the view records for a user sequence. This sequence is similar for both the user viewing their own records, and a RA accessing them, but the sequence shows the flow for a RA viewing them. The diagram is included in Figure 23.



Source: Own elaboration

Figure 23: Sequence diagram of the case of use of viewing a user's records, and open one in detail

The prerequisites are similar as before, but now another one is added:

- The RA has already searched the user whose records they want to view.

As the process of searching works in the same way for viewing and adding records, and it has been illustrated in the previous sequence diagram, this process has been omitted from this sequence diagram. The sequence, therefore, starts with the RA selecting the user whose records they want to view. This actions starts to calls: one to retrieve the user's data, and once the data is retrieved, its address is used to instantiate the smart contract method 'getRecordsForUser' to get an array with all the records associated in the blockchain network to the user's address. Then, the results table is filled in 'contract_view.js'; this file also creates a Record array that stores the results. Once results are shown to the RA through the UI, they are able to filter the search by inputting the illness the vaccine targets, or the hospital where it was administered, to filter the records shown in the table. This search is performed in real time: each character introduced starts the filter process, which is done in 'view_records.js' by hiding or displaying the rows of the table. Whenever the RA chooses to do so, they can expand the information of a record by clicking its corresponding button, which opens a new HTML document with all the record's information that is retrieved from the Record array created earlier. This page allows the RA to update any fields of the record as well through various options; this will be shown in chapter 6 of the report.

5.3 IMPLEMENTATION

In this section, divided in three parts, we will detail further the functionalities and the actual system that was developed for this project. These three parts are the three main components and foundations of the system: the smart contract, the API and finally the app itself, divided as well in the administrator-side website, and the user-side app.

The source code for the smart contract will be provided in the Appendix X, while the code for the API and the overall app will be accessible through a GitHub repository.

5.3.1 SMART CONTRACT

The structure of the smart contract, compiled with Solidity, was detailed in the previous Section 5.2.2.2, and in this section we will examine the functions of the contract in detail:

- **addRecordForUser**: Adds a new vaccination record for a specific user. The function has all the data included in the struct record as an input, and it simply creates a new Record with all that information and adds it into the userRecords array of the user address specified. This function has a onlyRecordAdmin modifier.

- **getRecordsForUser**: Retrieves all vaccination records for a specific user. This function returns all records if the address passed as an input is a valid user, or an empty record otherwise. This function does not have a particular modifier; instead, it requires that the address that calls the function is either a Master Administrator, a Record Administrator, or the same address passed as the input, this way allowing administrators or the user themselves to view the user's records.

- **updateRecord**: Updates an existing vaccination record for a user. Similarly to addRecordForUser, this function has an onlyRecordAdmin modifier and has all Record attributes minus status as inputs, but also the index of the Record to be updated. The function checks whether the index is valid, loads the Record and updates its values.

- **updatePendingRecord**: Updates only the status, lot number and doctor of a vaccination record. This function works exactly like updateRecord, but it is used to update records that are pending, meaning they are records for vaccines yet-to-be administered, generated either by the user scheduling a session, or a doctor generating a reminder.

- **addRecordAdmin** and **removeRecordAdmin**: Used by the master admin to add or remove record administrators. This is only accessible by Master Administrator.

- **checkAdmin**: Checks the admin status of a specific address. This function has been used mainly for testing purposes.

5.3.2 API

After going over the smart contract and its functions, the next section explores the endpoints defined in the API, which interact with the off-chain database and encompass the overall functionalities of the app related to off-chain operations. All the endpoints in this section aim to retrieve information from the database, and therefore, they are implemented as GET methods.

The following endpoints have been developed to facilitate data retrieval:

- **/api/users**: This endpoint retrieves all the users registered in the system. It returns a list of user objects containing their relevant information.

- **/api/users/{id}**: Using this endpoint, you can retrieve the user linked to the specified ID. It returns the user object associated with the provided ID.

- **/api/users/{name}/{lastname1}/{lastname2}**: This endpoint allows you to retrieve user(s) based on the name and last names provided. It supports null fields for any of the three path variables, meaning you can search for users using a combination of name and last names, or only one of them.

- **/api/hospitals**: With this endpoint, you can retrieve all the hospitals defined in the system's database. It returns a list of hospital objects containing their details.

- **/api/hospitals/{id}**: This endpoint retrieves the hospital linked to the specified ID. It returns the hospital object associated with the provided ID.

- **/api/hospitals/name/{name}**: Using this endpoint, you can retrieve the hospital linked to the specified name. It returns the hospital object associated with the provided name.

- **/api/doctors/hospital/{id}**: This endpoint fetches all the doctors who have worked in the hospital specified by the ID. It returns a list of doctor objects associated with the provided hospital ID.

- **/api/vaccines/hospital/{id}**: This endpoint retrieves all the vaccines provided by a specific hospital, identified by the ID. It returns a list of vaccine objects associated with the provided hospital ID.

- **/api/vaccine/{id}**: This endpoint allows you to retrieve the vaccine associated with the specified ID. It returns the vaccine object associated with the provided ID.

These endpoints provide convenient ways to fetch information from the off-chain database, enabling seamless integration of the blockchain-based application with external systems and ensuring efficient data retrieval for various functionalities.

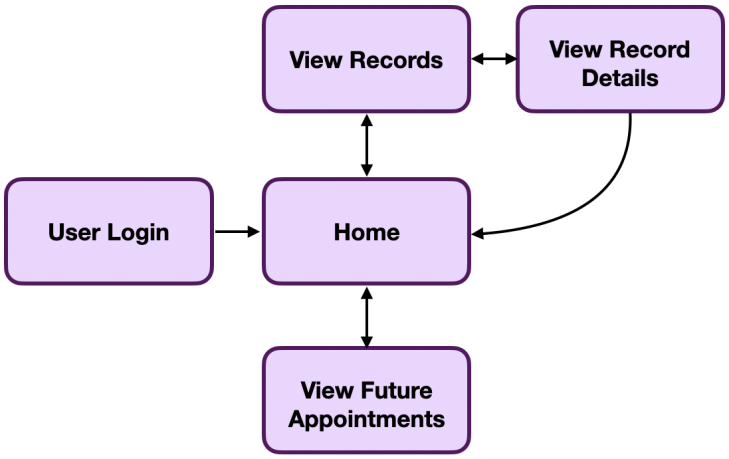
5.3.3 APP

Lastly, the final foundation of the project is the app itself. The objective of this interface for both users and administrators is to interact with the API and the blockchain network seamlessly, in a intuitive and simple manner. The design of the app has been done with a combination of HTML, javascript and CSS, and is divided in two experiences: the user side and the administrator side.

In the following two subsections the structure of both the user and administrator side will be explored, and Chapter 6 will include screenshots of the actual UI.

5.3.3.1 USER SIDE

The user side is built with mobile phones in mind, and as such the interface and design is optimized for mobile view. Figure 24 illustrate the navigability diagram of the user app. Users are required to login first, and then are greeted with a home page that allows them to view their records or view their future appointments. Choosing the option to view records shows only vaccines applied (the yet-to-be applied records are shown in the 'future appointments' option), with the option to expand the details of any of the records as well.

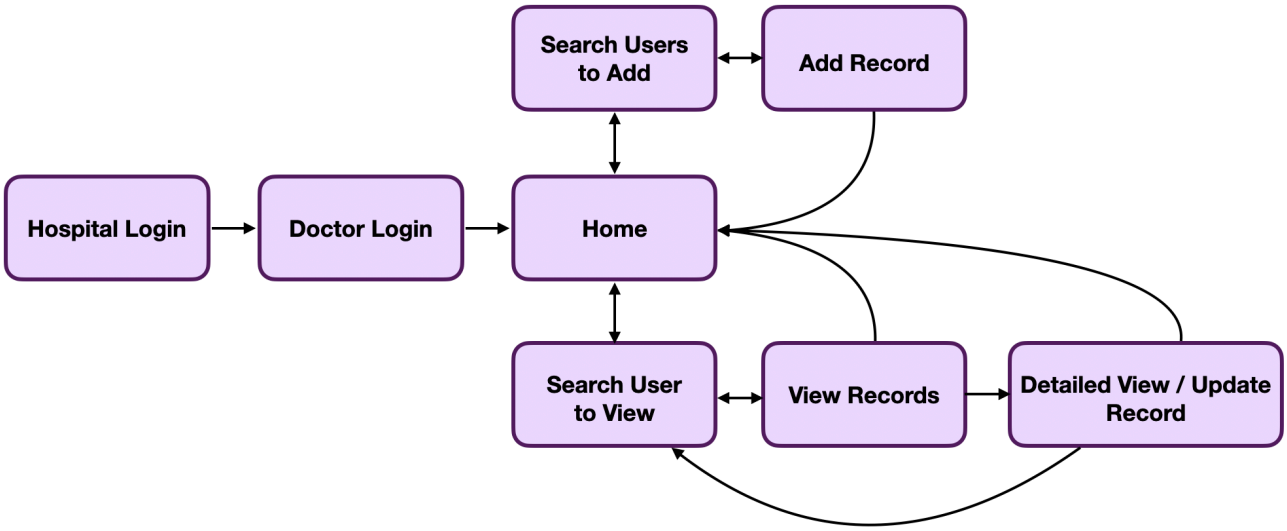


Source: Own elaboration

Figure 24: Navigability diagram for the user application

5.3.3.2 ADMIN SIDE

The admin side, on the other hand, is built with laptops and PCs in mind, and the interface reflects that. In terms of the navigation, a doctor logging in is first greeted with a page where a dropdown menu shows all hospitals in the system. Upon choosing their hospital, they are shown a similar page where they select their name in a list of doctors, and enter their password. Successfully logging in shows the home page, with the two main options available fixed in a sidebar that is visible at all times. Choosing either 'View' or 'Add' reveals the search user option. In the 'Add' case, after selecting the desired user a form is shown, with all fields to be input by the doctor in order to add a record to the blockchain. In the 'View' case, a similar page to the search option appears. This view shows all records and gives the doctor the option to filter the table by illness or hospital. Expanding the information for a given record shows a similar form than 'Add Record' with options disabled, being view-only. This page also allows the doctor to update a record. Two options are give, depending on the status of the record (vaccine administered or pending). Choosing either option enables the appropriate fields to be filled in order to complete the updating task.



Source: Own elaboration

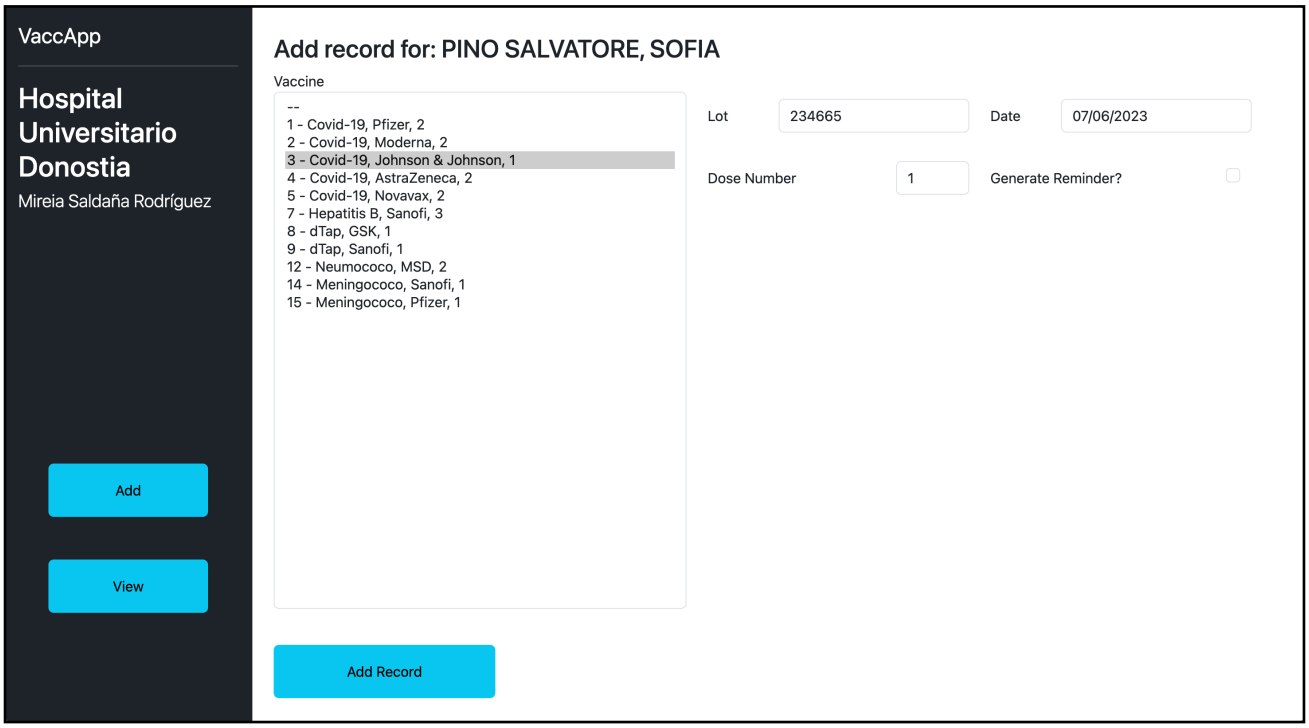
Figure 25: Navigability diagram for the administrator website

6 RESULTS ANALYSIS & CONCLUSIONS

In this chapter we will demonstrate the results obtained with this project, and how the aforementioned objectives have been achieved. The chapter will delve into the flow of the application from the doctor's and user's perspectives, explaining the functionalities of the application and including figures alongside to better illustrate the user experience. The detailed flow of the application is included in Appendix IV.

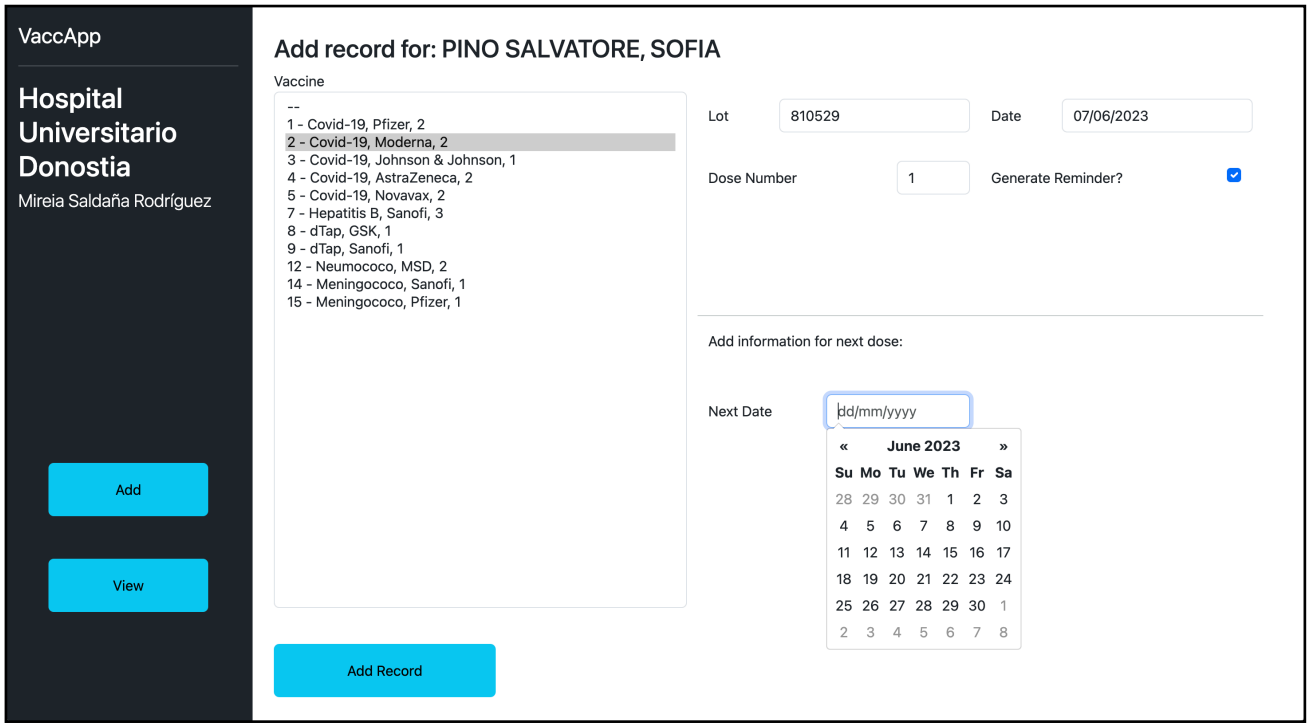
Doctors can add or view records for a particular user upon searching them. Figure 26 shows an example of the 'Add Record' view, where a doctor can add a record for a user filling the input fields given. A reminder for a future appointment can also be generated (Figure 27).

Doctors are able to view full vaccination cards through the 'View' option, and can also view extended details of each particular record. This view also allows doctor to update the opened record. Figures 28 and 29 show these views.



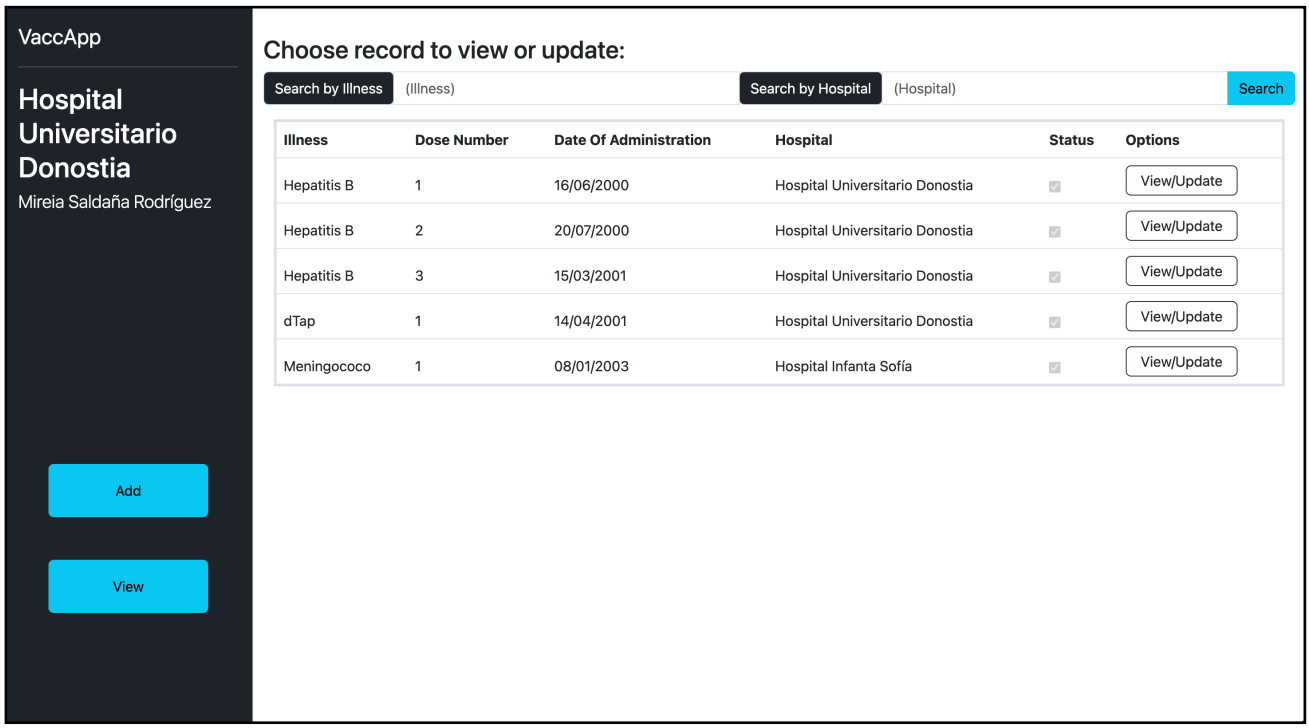
Source: Own elaboration

Figure 26: Add record view



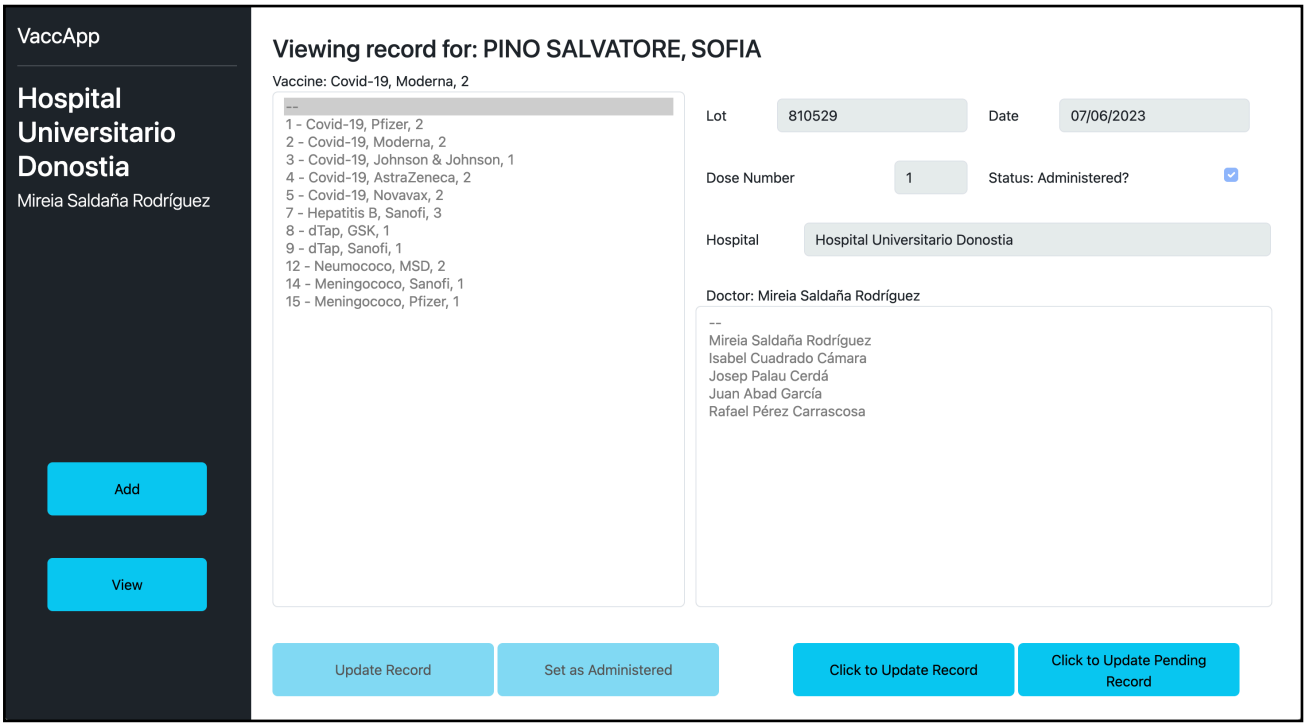
Source: Own elaboration

Figure 27: Add record and reminder for future appointment view



Source: Own elaboration

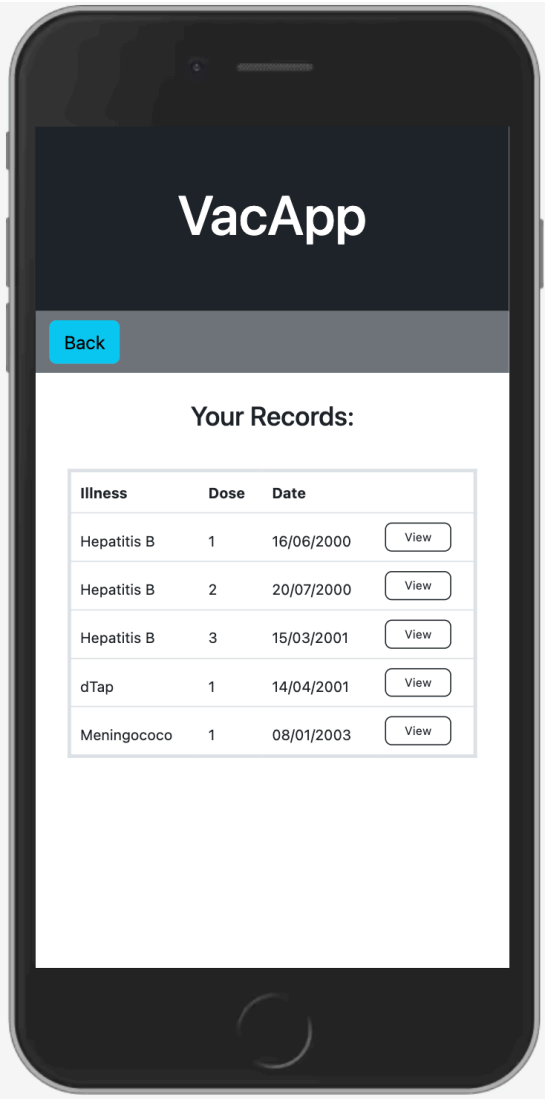
Figure 28: View records view



Source: Own elaboration

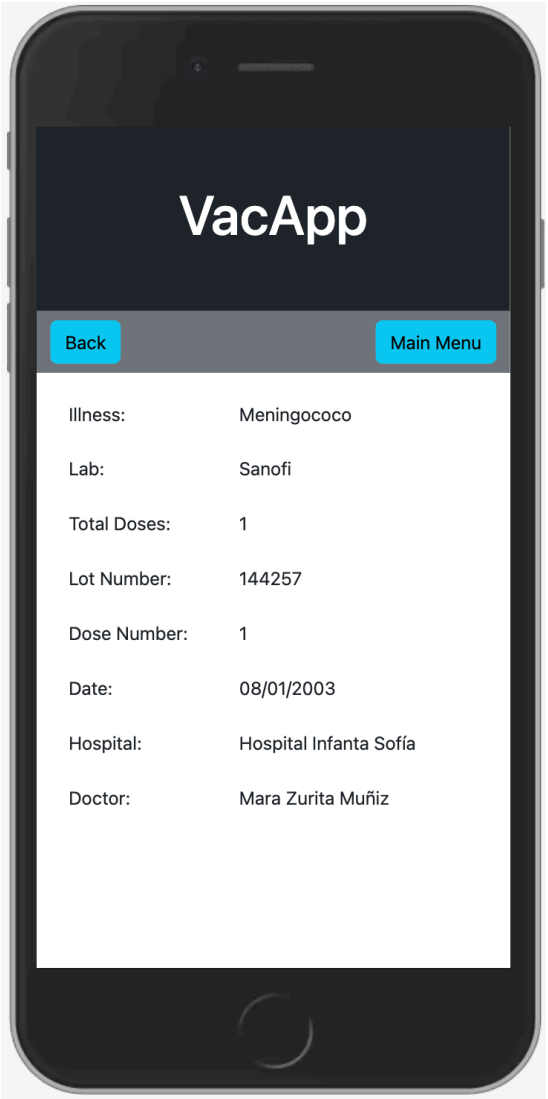
Figure 29: View expanded record information view

In terms of the user experience, they can either view their own records (Figure 30) or the future appointments. Note that the former does not include records that have a pending status: those are shown in the 'Future appointments' option. Finally, users can view any record in detail as well (Figure 31).



Source: Own elaboration

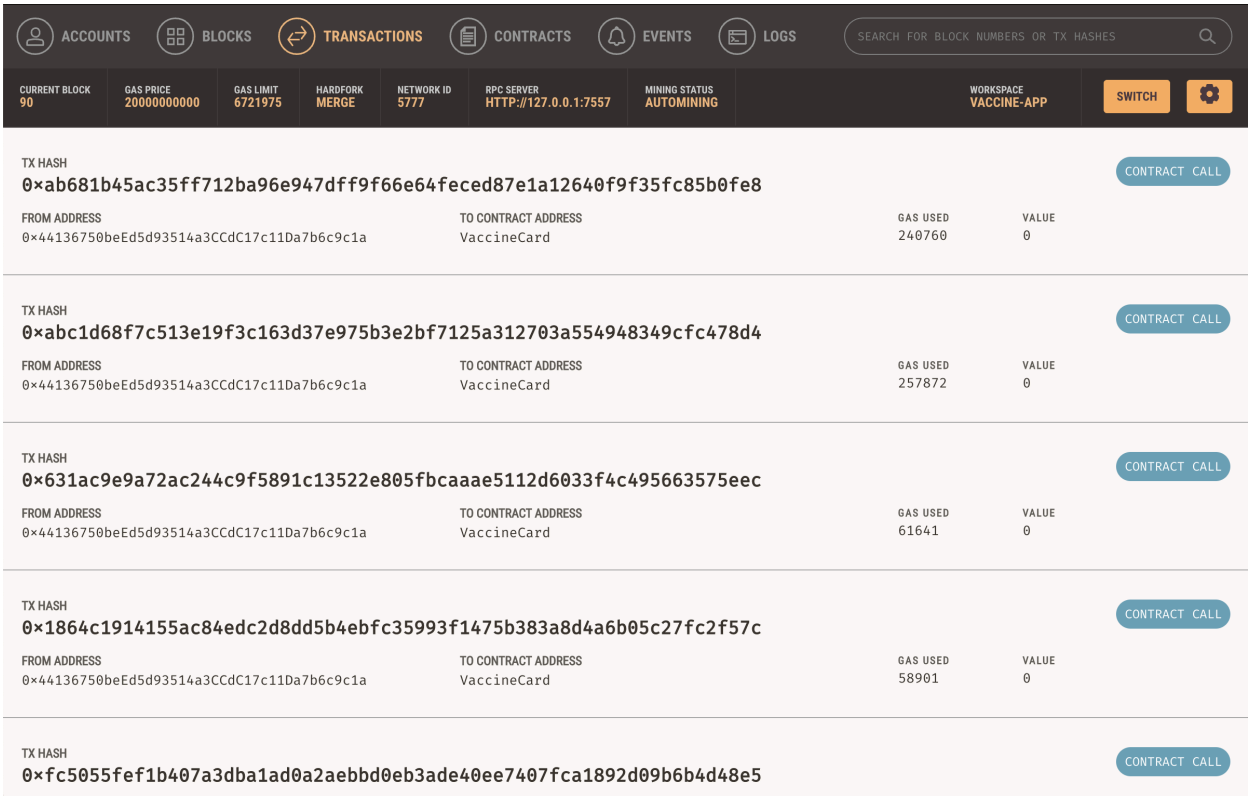
Figure 30: View user's own records view



Source: Own elaboration

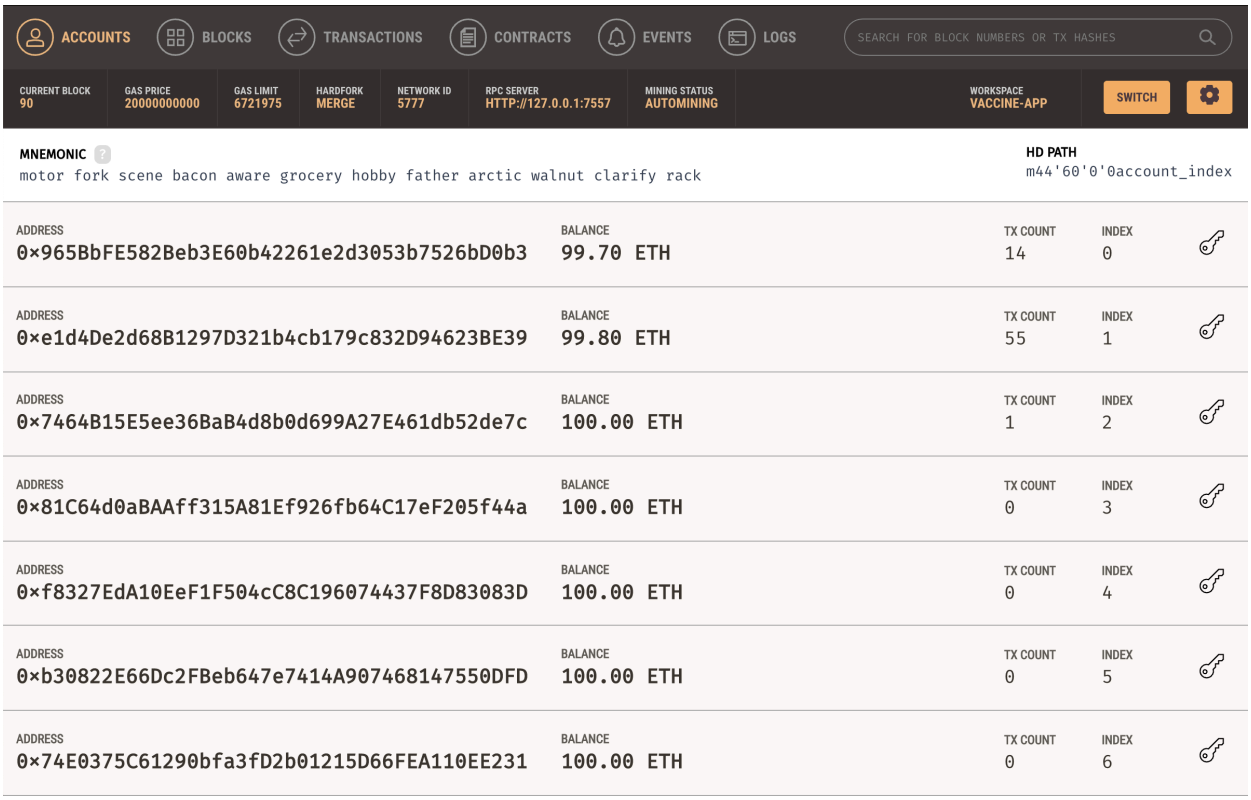
Figure 31: Extended record information view

As a final note, we will inspect the interaction with the local private Ethereum network. As it can be spotted in Figure 32, transactions are correctly recorded, with the hospital address as sender for all doctor's methods. User's view method provides the user as the address, but view calls are not recorded as transactions, as they do not incur in gas costs. Figure 33 shows the accounts of the network: as expected, accounts from hospitals or contract deployers have spent some of the currency from their wallets when making transactions.



Source: Own elaboration

Figure 32: Transaction page from Ganache



Source: Own elaboration

Figure 33: Local blockchain network accounts

Regarding the conclusions, the development of this project had, as stated in Section 4.2, three main objectives to be achieved:

- **Create a user-friendly interface to access vaccination records easily:** although bare bones, the interface achieves the objective set. Vaccine records are easy to access for both users and administrators, and information is presented in a clear and comprehensive manner. Both the user and administrator apps are easy to navigate, and the functionalities they offer are presented to users and administrators distinctly.

- **Implement a secure, immutable and traceable system:** records are properly stored in the blockchain, ensuring the immutability, integrity and overall security of the data stored. The application also includes hashing and salting of stored passwords to improve security in terms of accessing the system.

- **Explore the possibilities and potential of blockchain technology:** this objective focused on learning through research and experience while working on the project. The lessons and information learned throughout the whole process is reflected in a following paragraph, centered around the future steps that could be taken in order to improve the system, its viability and its value.

As it can be discerned, the objectives have been achieved, and the system offers the security and ease-of-use we set to achieve at the start of the project.

7 FUTURE WORK

In terms of future work and overall findings during the research and development of the project, it is evident that blockchain can be a essential asset in the healthcare industry, providing security for all the sensible data that exists in it. The benefits it provides are not as widely known by the population, but blockchain truly is an exciting technology that can improve our lives without us necessarily noticing it.

These possibilities that blockchain provides are some of the focuses of the future steps that could be taken in order to improve this system. These future works, additions or modifications include:

- Improve user interactivity via appointment scheduling, option to view close hospital or health centers or on-screen notifications.

- Study alternative blockchain networks to deploy the system, aiming to reduce costs and improve latency, as Ethereum, while popular, is not the better option in those fields.

- Improve overall security to better align with Web 3.0 and OAuth2.0 standards.

- Expand the blockchain database to include vaccines and their lots. This would be the biggest undertaking but the most obvious and beneficial one. Including them in the system would allow healthcare providers to manage the supply chain throughout the whole process, improving traceability and allowing to manage potential issues more effectively.

- Improve the overall look-and-feel of the application, with a particular focus on user experience.

Blockchain offers tremendous possibilities in the health industry. The reinforcing and positive thought is that, while it might be a not fully explored and evolved technology, it already improves existing systems, and is shown to work already for big territories. It is a matter of time that blockchain is a part of more of these essential

industries, and with the apparent improvements that it generates, we can only hope that this happens sooner rather than later.

BIBLIOGRAPHY

[1] Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] Pontem Network. (n.d.). The History of Smart Contracts. Retrieved from https://pontem.network/posts/the-history-of-smart-contracts

[3] Chainlink Blog. (March 7, 2023). Smart Contract Platforms. Retrieved from https://blog.chain.link/smart-contract-platforms/

[4] Kanwal, I. (October 14, 2022). History and Evolution of Blockchain Technology. Retrieved from https://www.bloctechsolutions.com/blog/history-and-evolution-of-blockchain-technology/

[5] LeewayHertz. (n.d.). Blockchain 4.0. Retrieved from https://www.leewayhertz.com/blockchain-4-0/

[6] Kamenivskyy Y, Palisetti A, Hamze L, Saberi S. A Blockchain-Based Solution for COVID-19 Vaccine Distribution. IEEE Engineering Management Review. 2022 Jan 25;50(1):43–53. doi: 10.1109/EMR.2022.3145656. PMCID: PMC9088775.

[7] Dubovitskaya A, Xu Z, Ryu S, Schumacher M, Wang F. Secure and Trustable Electronic Medical Records Sharing using Blockchain. AMIA Annu Symp Proc. 2018 Apr 16;2017:650-659. PMID: 29854130; PMCID: PMC5977675.

[8] IBM. (n.d.). Blockchain. Retrieved from https://www.ibm.com/topics/blockchain

[9] Frankenfield, J. (September 27, 2022). What Is Ethereum and How Does It Work? Retrieved from https://www.investopedia.com/terms/e/ethereum.asp

[10] Ethereum Foundation. (n.d.). What is Ethereum? Retrieved from https://ethereum.org/en/what-is-ethereum/

[11] DeNisco Rayome, A. (March 1, 2018). 6 ways companies are using blockchain to drive value right now. TechRepublic. Retrieved from https://www.techrepublic.com/article/6-ways-companies-are-using-blockchain-to-drive-value-right-now/

[12] Moghe, U. (n.d.). Blockchain 4.0: The Future of Distributed Ledgers. Retrieved from https://ideausher.com/blog/blockchain-4-0/

[13] Essex D, Kerner S, Gillis A. (n.d.). What is Web 3.0 (Web3)? Definition, guide and history. Retrieved from https://www.techtarget.com/whatis/definition/Web-30

[14] Lalithnarayan C. (January 28, 2021). An Overview of Consensus Protocols in Blockchain. Retrieved from https://www.section.io/engineering-education/blockchain-consensus-protocols/

[15] Ledger Insights. (October 2, 2020). World Health Organization partners with Estonia for blockchain COVID-19 certificates. Retrieved from https://www.ledgerinsights.com/world-health-organization-partners-with-estonia-for-blockchain-covid-19-certificates/

[16] InnoHEALTH Magazine. (2021). What is Estonia doing with blockchain in providing healthcare to its citizens? Retrieved from https://innohealthmagazine.com/2021/in-focus/what-is-estonia-doing-with-block-chain-in-providing-healthcare-to-its-citizens/

[17] Báez González, J. R. (June 29, 2021). Vitalpass, El Certificado de vacunación de Tecnología blockchain que empezó a implementarse en Colombia. Anadolu Ajansı. Retrieved December 18, 2022, from https://www.aa.com.tr/es/mundo/vitalpass-el-certificado-de-vacunaci%C3%B3n-de-tecnolog%C3%ADa-blockchain-que-empez%C3%B3-a-implementarse-en-colombia/2289118

[18] Abad, J. (May 30, 2022). Certificado de Vacunación Colombiano con registro en blockchain fue convalidado por la Unión Europea. Cointelegraph. Retrieved December 18, 2022, from https://es.cointelegraph.com/news/colombian-vaccination-certificate-with-blockchain-registration-was-validated-by-the-european-union

[19] Kuo, T. T., Kim, H. E., & Ohno-Machado, L. (2017). Blockchain distributed ledger technologies for biomedical and health care applications. J Am Med Inform Assoc, 24(6), 1211-1220. doi: 10.1093/jamia/ocx068. PMID: 29077941. Retrieved from https://pubmed.ncbi.nlm.nih.gov/29077941/

[20] IBM. (n.d.). ¿Qué es Java Spring Boot?. Retrieved from https://www.ibm.com/mx-es/topics/java-spring-boot

APPENDIX I: SUSTAINABLE DEVELOPMENT GOALS (SDG)

As stated on their website, the Sustainable Development Goals (SDGs) are a universal call to action for all countries to promote prosperity while protecting the planet. They recognize that ending poverty must be accompanied by strategies that foster economic growth and address various social needs, including education, health, and more. These goals have become even more crucial in the context of the COVID-19 pandemic, providing a vital framework for global recovery efforts.

Source: Courtesy of United Nations

Figure 34: Sustainable development goals of the United Nations

In the case of this project, it is evident that it aligns closely with SDG 3: Good Health and Well-Being. The tool developed aims to improve the management, data storage, and access of vaccination records for every user. By leveraging blockchain technology, the efficiency of the vaccine industry can be enhanced, and the availability, security, and accessibility of vaccination data can be improved. This contributes to the broader goal of ensuring good health and well-being for all.

By streamlining the process of storing and accessing vaccination records, the tool enables healthcare providers to have an accurate and up-to-date overview of an individual's vaccination history. Furthermore, it empowers individuals to have better control over their health data and ensuring continuity of care.

Moreover, this project also aligns with SDG 9: Industry, Innovation, and Infrastructure. By adopting blockchain technology to create a decentralized and secure system for managing vaccination records, the project promotes innovation and infrastructure development in the healthcare sector. It offers a more efficient and transparent solution for data management, reducing reliance on traditional paper-based systems and enhancing interoperability between different healthcare providers and institutions.

The long-term relevance of this project is evident as vaccines continue to play a crucial role in global health. With ongoing advancements in vaccine research and the need for widespread vaccination programs, an efficient and secure system for managing vaccination records becomes increasingly important. By aligning with SDG 3 and SDG 9, this tool contributes to the broader sustainable development agenda and supports the global efforts to ensure better health outcomes, promote innovation, and build resilient healthcare systems for the future.

APPENDIX II: SMART CONTRACT SOURCE CODE

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract VaccineCard {

    // Define a struct to store the details of a vaccination record
    struct Record {
        Vaccine vaccine;     // The vaccine administered
        uint48 lot;          // Lot of the vaccine
        string date;         // The date the vaccine was administered
        uint8 doseNumber;    // The dose number of the vaccine administered
        bool status;         // True if already administered, false if it is a future vaccine
        string hospital;     // Hospital where it was administered
        address owner;       // The Ethereum address of the record owner
        string doctor;       // Doctor that administered vaccine
    }

    // Define a struct to store the details of a vaccine
    struct Vaccine {
        string illness;      // The illness the vaccine targets
        string lab;          // Lab that produced the vaccine
        uint8 doses;         // Total number of doses to complete the vaccination
    }

    address sysAdmin;

    // Define a mapping of Ethereum addresses to user records
    mapping(address => Record[]) private userRecords;

    // Define a mapping of Ethereum addresses to boolean values to track record admin permissions
    mapping(address => bool) private isRecordAdmin;

    // Define a mapping of Ethereum addresses to boolean values to track master admin permissions
    mapping(address => bool) private isMasterAdmin;

    // Define the contract constructor to set the contract deployer as an admin
    constructor() {
        // Define contract deployer as system administrator
        sysAdmin = msg.sender;
        isMasterAdmin[sysAdmin] = true;
        // Define hospitals as record administrators; rest of addresses for users
        isRecordAdmin[0xe1d4De2d68B1297D321b4cb179c832D94623BE39] = true;
        isRecordAdmin[0x7464B15E5ee36BaB4d8b0d699A27E461db52de7c] = true;
        isRecordAdmin[0x44136750beEd5d93514a3CCdC17c11Da7b6c9c1a] = true;
    }

    // Define a modifier to restrict access to master admin-only functions
    modifier onlyMasterAdmin() {
        require(isMasterAdmin[msg.sender], "Only master admins can access this function");
        _;
    }

    // Define a modifier to restrict access to record admin-only functions (all except add/removal of
        admins)
    modifier onlyRecordAdmin() {
        require(isRecordAdmin[msg.sender] || isMasterAdmin[msg.sender], "Only record admins and
master admins can access this function");
        _;
    }
```

```
    // Define a function to add a new vaccination record to the records mapping
    function addRecordForUser(address _user, Vaccine memory _vaccine, uint48 _lot, string memory
_date, uint8 _doseNumber, bool status, string memory _hospital, string memory _doctor) public
onlyRecordAdmin {
        userRecords[_user].push(Record(_vaccine, _lot, _date, _doseNumber, status, _hospital, _user,
_doctor));
    }

    // Define a function to retrieve all of the vaccination records for a specific Ethereum address.
       Admins can view the records of
    // any user, while regular users can only view their own
    // The function checks if the caller is an admin or if the caller's Ethereum address matches the
       _address parameter, and returns the records accordingly.
    function getRecordsForUser(address _user) public view returns (Record[] memory) {
        if (isRecordAdmin[msg.sender] || isMasterAdmin[msg.sender] || msg.sender == _user) {
            return userRecords[_user];
        } else {
            return new Record[](0);
        }
    }

    // This function takes additional parameters _user and _index, where _user specifies the Ethereum
       address of the user whose record is being updated,
    // and _index specifies the index of the record in the user's array of records. The function then
       updates the record with the new values provided,
    // but only admins can call this function using the onlyRecordAdmin modifier.
    function updateRecord(address _user, uint48 _index, Vaccine memory _vaccine, uint48 _lot,
     string memory _date, uint8 _doseNumber, string memory _hospital, string memory _doctor) public
onlyRecordAdmin {
        require(_index < userRecords[_user].length, "Invalid index");
        Record storage record = userRecords[_user][_index];
        record.vaccine = _vaccine;
        record.lot = _lot;
        record.date = _date;
        record.doseNumber = _doseNumber;
        record.hospital = _hospital;
        record.doctor = _doctor;
    }

    // Function to update only the status of the record, for pending vaccines
    function updateStatus(address _user, uint48 _index) public onlyRecordAdmin {
        require(_index < userRecords[_user].length, "Invalid index");
        Record storage record = userRecords[_user][_index];
        record.status = true;
    }

    // Administration management functions
    function addRecordAdmin(address _address) public onlyMasterAdmin {
        isRecordAdmin[_address] = true;
    }

    function removeRecordAdmin(address _address) public onlyMasterAdmin {
        isRecordAdmin[_address] = false;
    }

    function checkAdmin(address _address) public view returns (string memory) {
        if (isRecordAdmin[_address]) {
            return "This account is a record administrator";
        } else if (isMasterAdmin[_address]) {
            return "This account is a master administrator";
        } else return "This account is not an administrator";
    }
}
```

APPENDIX III: INSTALLATION MANUAL

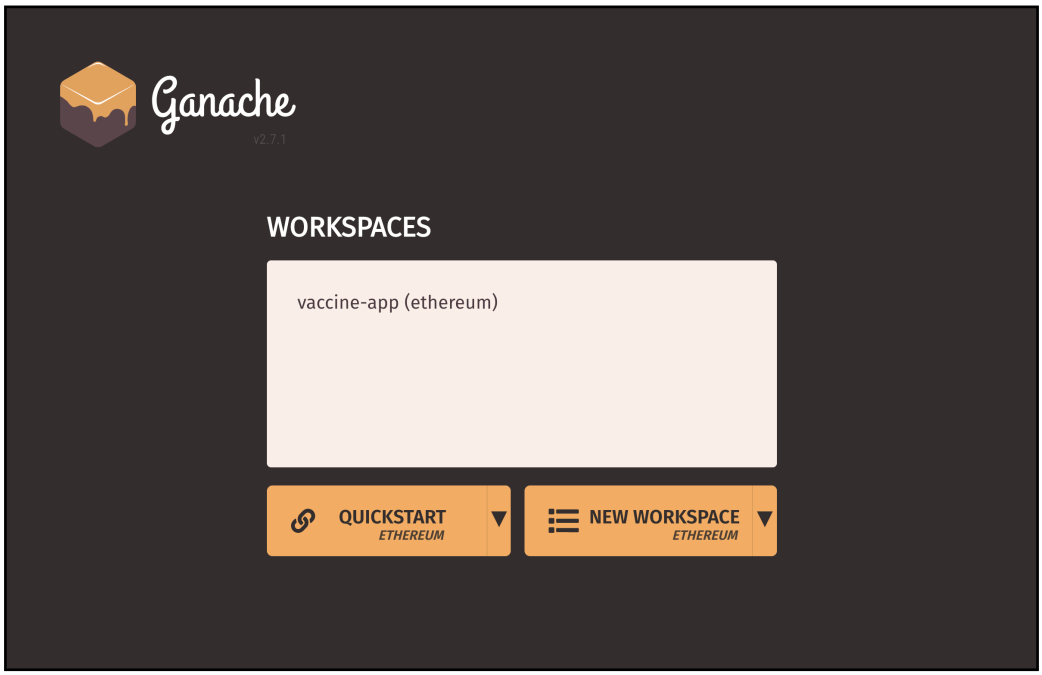
This appendix details the programs to be installed in order to properly set up the work environment for the project. In order to run the application, the following tools are needed:

- Ganache (used to set up the local private blockchain network).
- Truffle IDE (it is recommended to use its plugin in Visual Studio Code).
- IntelliJ (optional but recommended as it is optimized for both Java and HTML).

A.III.1 GANACHE

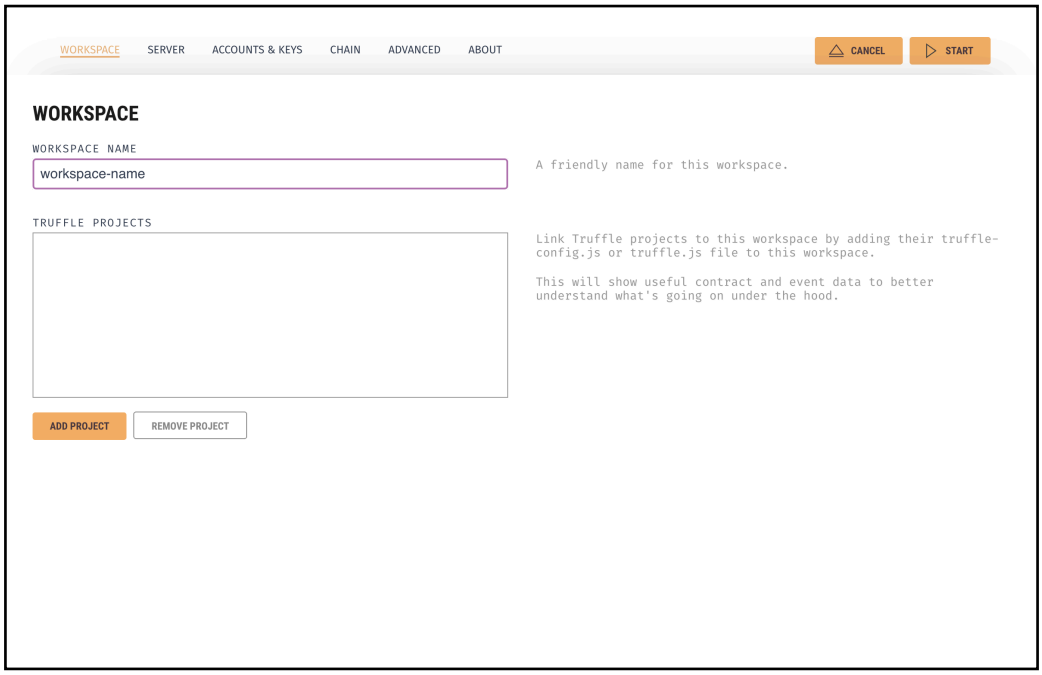
Once downloaded in your device, these are the steps to create the local private network in Ganache.

1. Open Ganache and create a new workspace. It is recommended to use the default Hostname. (Optional) Edit the number of accounts and wallet balance in 'Accounts & Keys'.



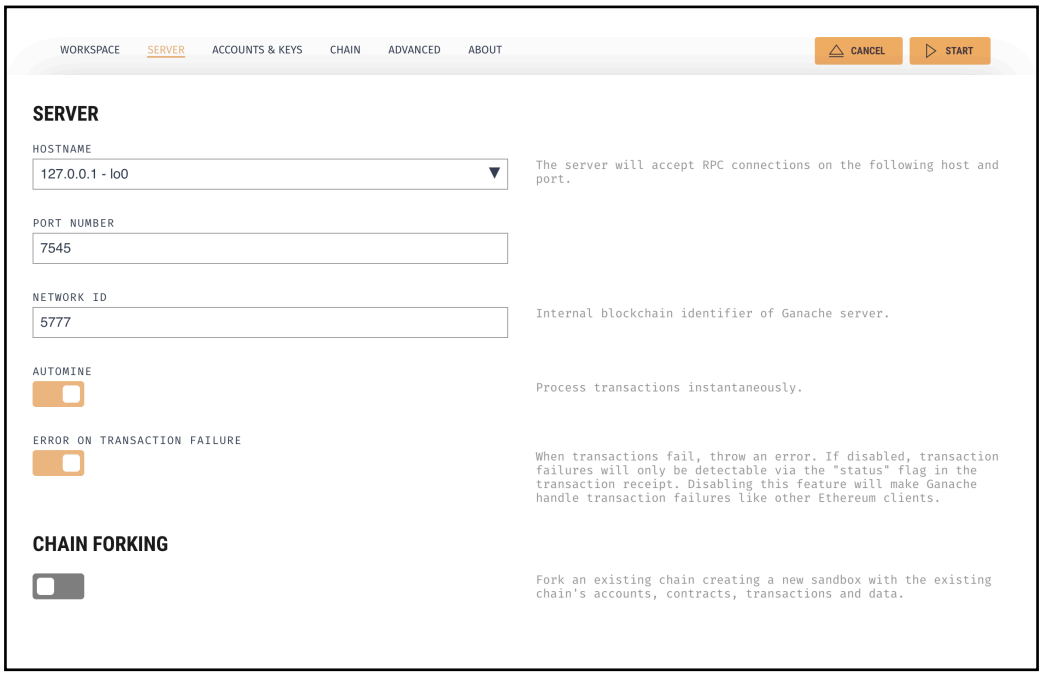
Source: Own elaboration

Figure 35: Ganache welcome page



Source: Own elaboration

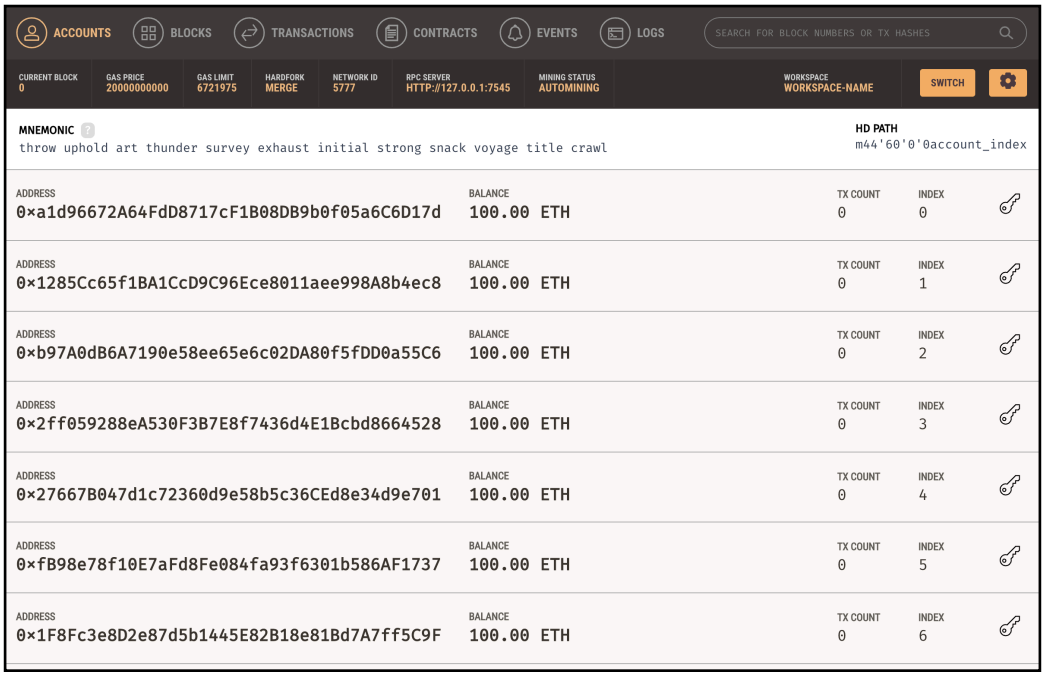
Figure 36: Workspace creation view



Source: Own elaboration

Figure 37: Server details option

2. Press ‘Start’ to create the workspace, that will open automatically.



Source: Own elaboration

Figure 38: Main view of network accounts

A.III.2 TRUFFLE IDE

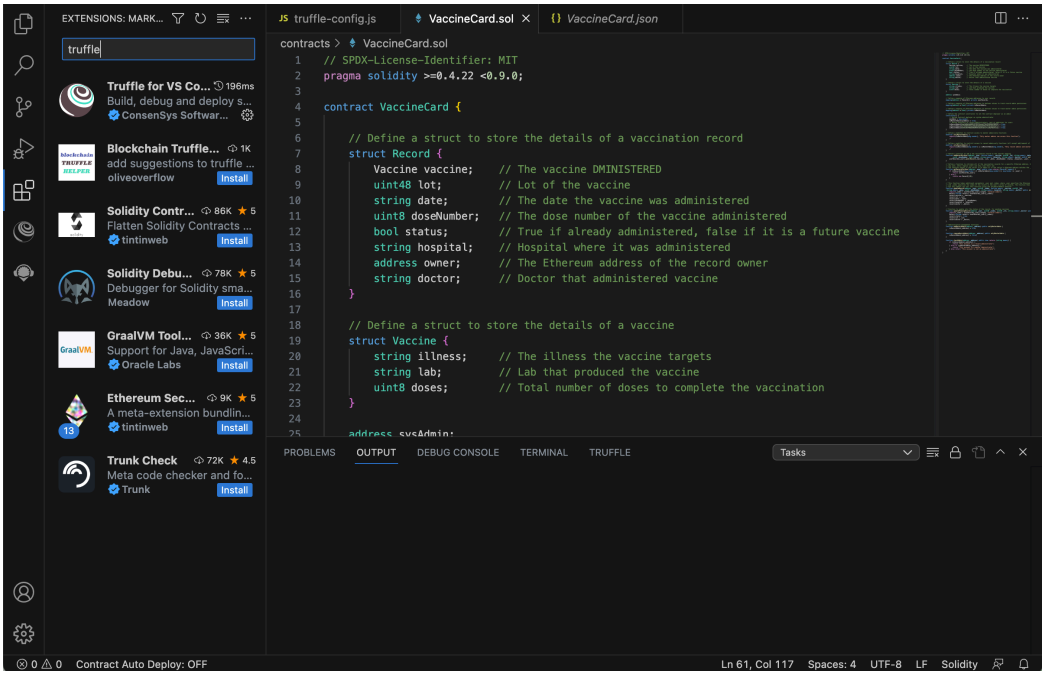
This section goes over the deployment of the smart contract in the Ethereum local network created with Ganache. This step requires previous installation of VS Code.

1. Install the Truffle VS Code plugin through the 'Extensions' tab (Figure 39).

2. In the 'Explorer' tab, search the folder that contains the smart contract. Right click on it and select 'Deploy Contract'. In the dropdown menu choose 'Create a New Network' (Figure 40).

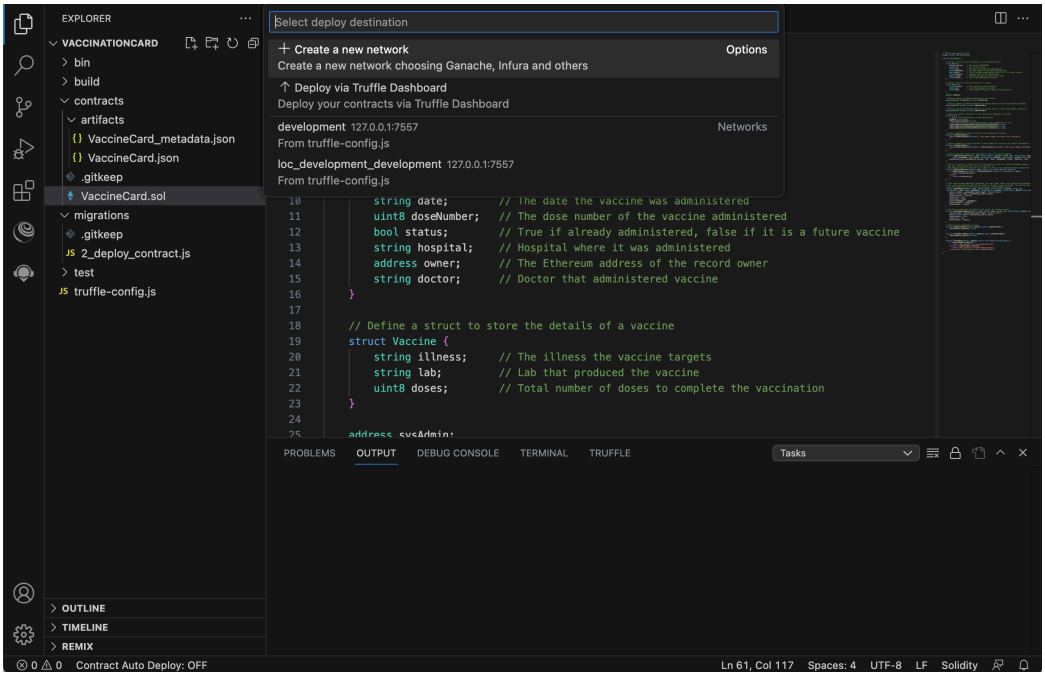
3. When asked, select 'Ganache Service' as the destination, 'Local' as the type, and enter the port number designated in the creation of the network (Figure 41).

4. Choose a name for the network. If successful, you should be shown the network set up in Ganache as an option, as illustrated in the last step of Figure 41. Select it.



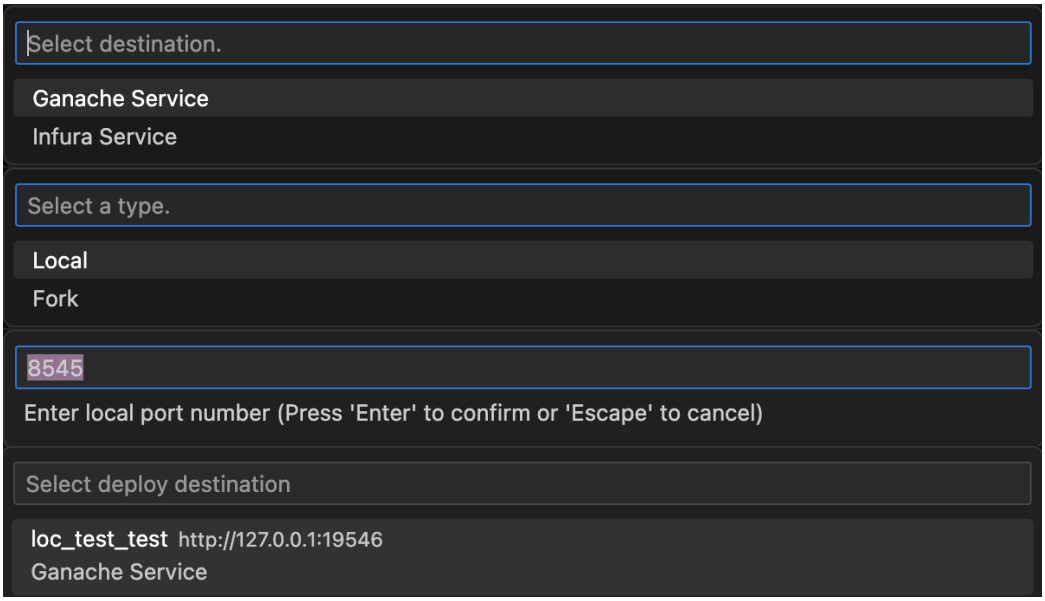
Source: Own elaboration

Figure 39: Truffle extension download in VS Code



Source: Own elaboration

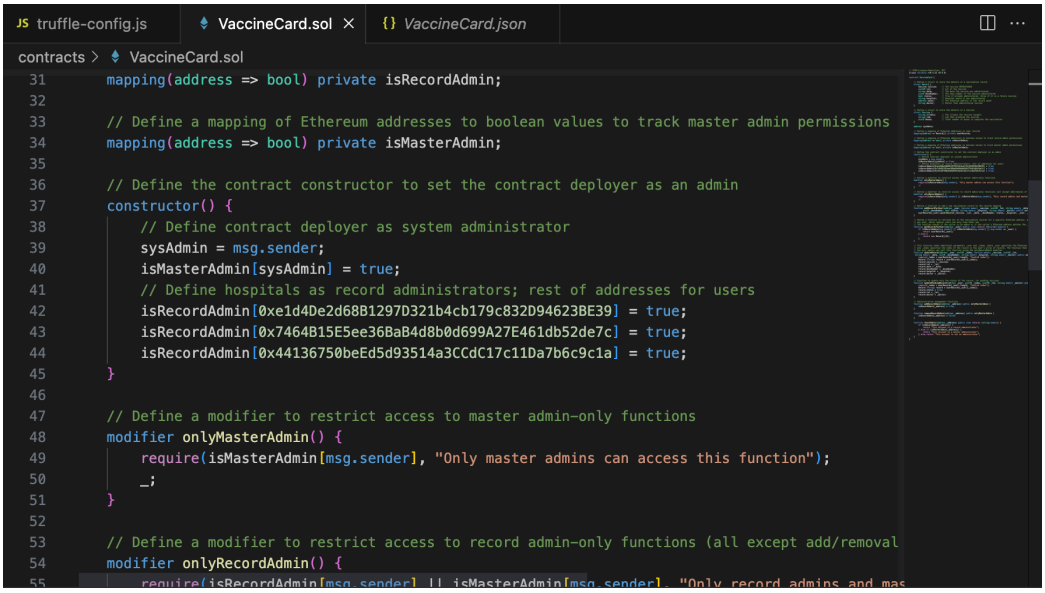
Figure 40: Contract deployment



Source: Own elaboration

Figure 41: Steps for network creation

Note, in order to set up administrator accounts, those have to be specified in the smart contract before its deployment. Look up the addresses you want to add as administrators in Ganache and set them in the constructor of the smart contract.



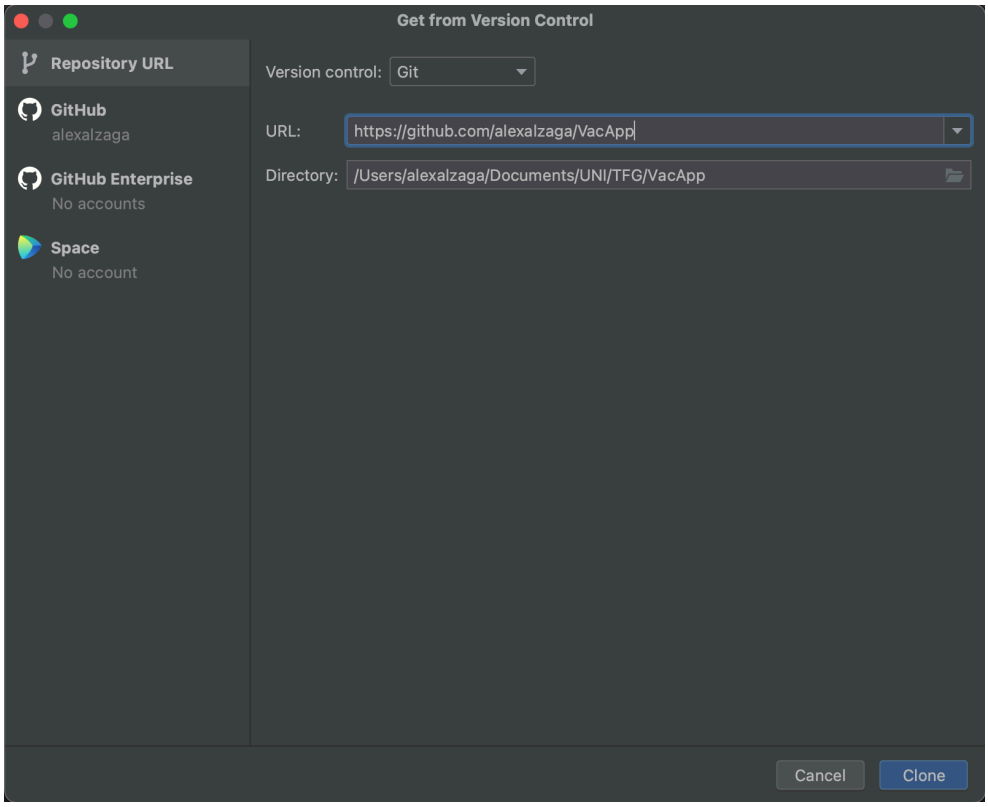
Source: Own elaboration

Figure 42: Smart contract constructor function

A.III.3 INTELLIJ IDE

This final section details the steps in order to run the application using IntelliJ and the implemented Spring Boot framework. This step requires previous installation of IntelliJ.

1. In IntelliJ, create a new project from Version Control, and introduce the URL indicated in Figure 43.

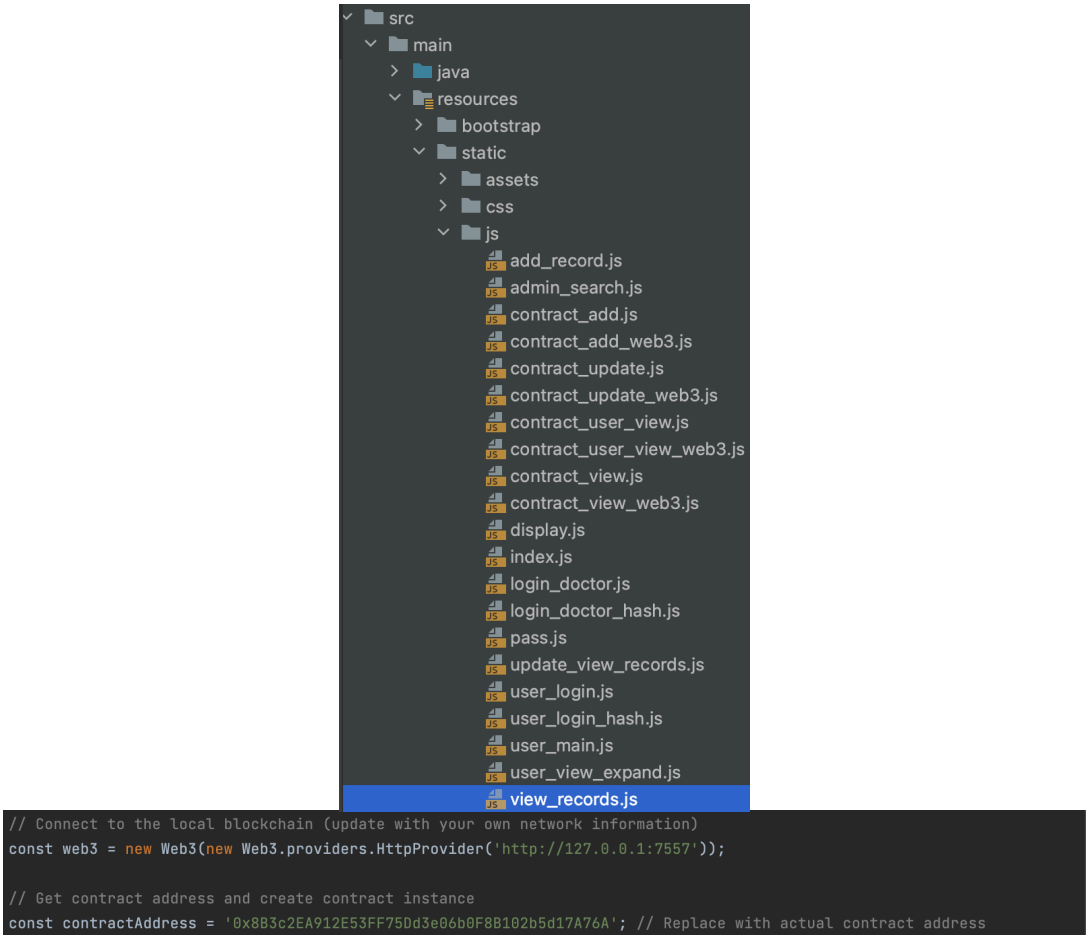


Source: Own elaboration

Figure 43: IntelliJ's project creation from Version Control view

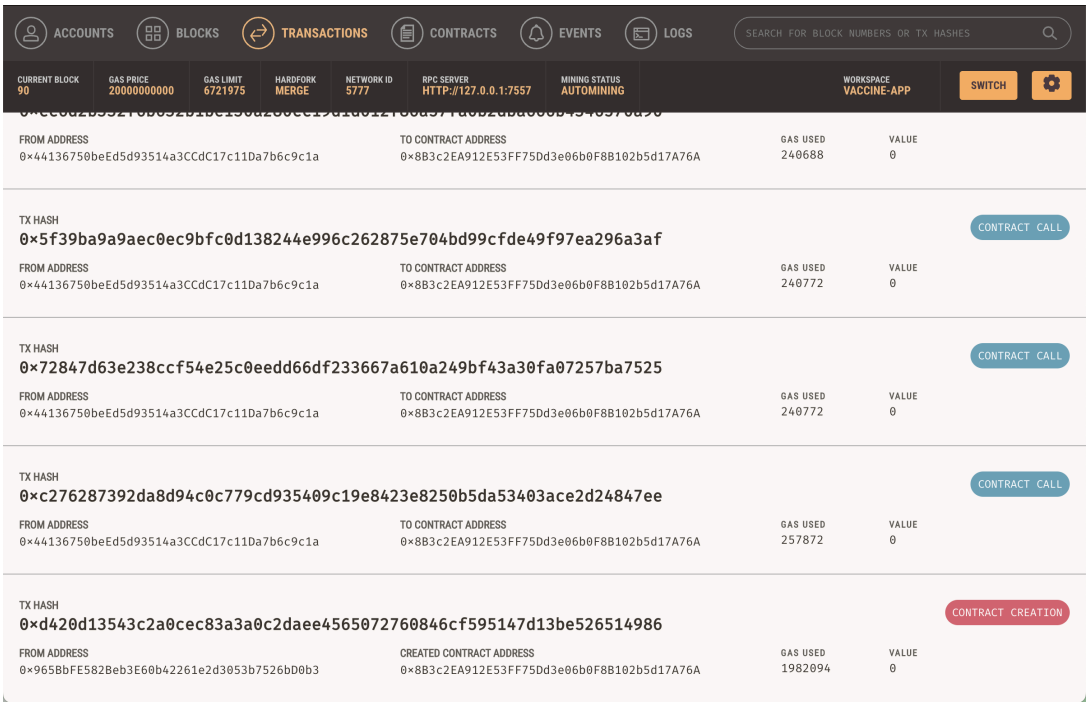
2. Before running the application, open the marked files, and change the IP and port number, and the contract address with the ones for your network (Figure 44). The contract address can be found in the Transactions tab in Ganache (Figure 45).

3. Run the file FinalApplication.java.



Source: Own elaboration

Figure 44: web3 project files and code requiring changes

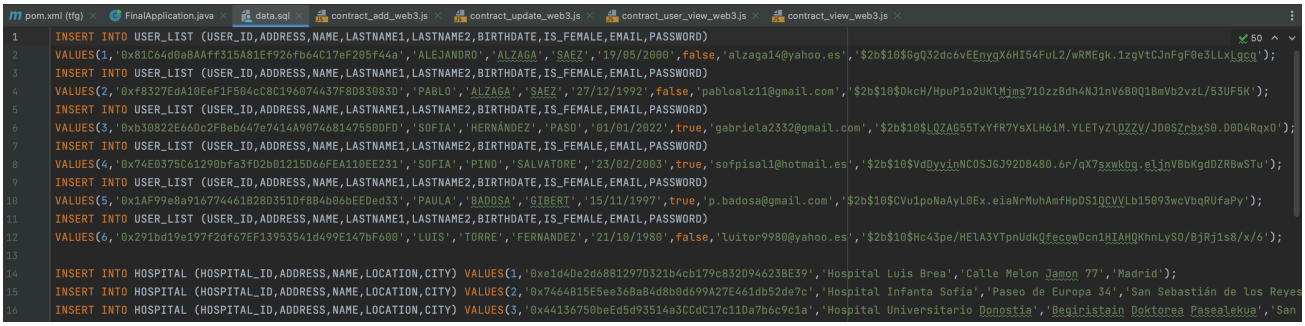


Source: Own elaboration

Figure 45: Ganache's transactions tab

Once done, the application can be open in localhost:8080 (admin view) or localhost:8080/user_login (user_view). Plain-text passwords for the included users and doctors are included in pass.txt.

Note, in data.sql you should update the addresses for the users and hospitals using the addresses generated in your Ganache private network.



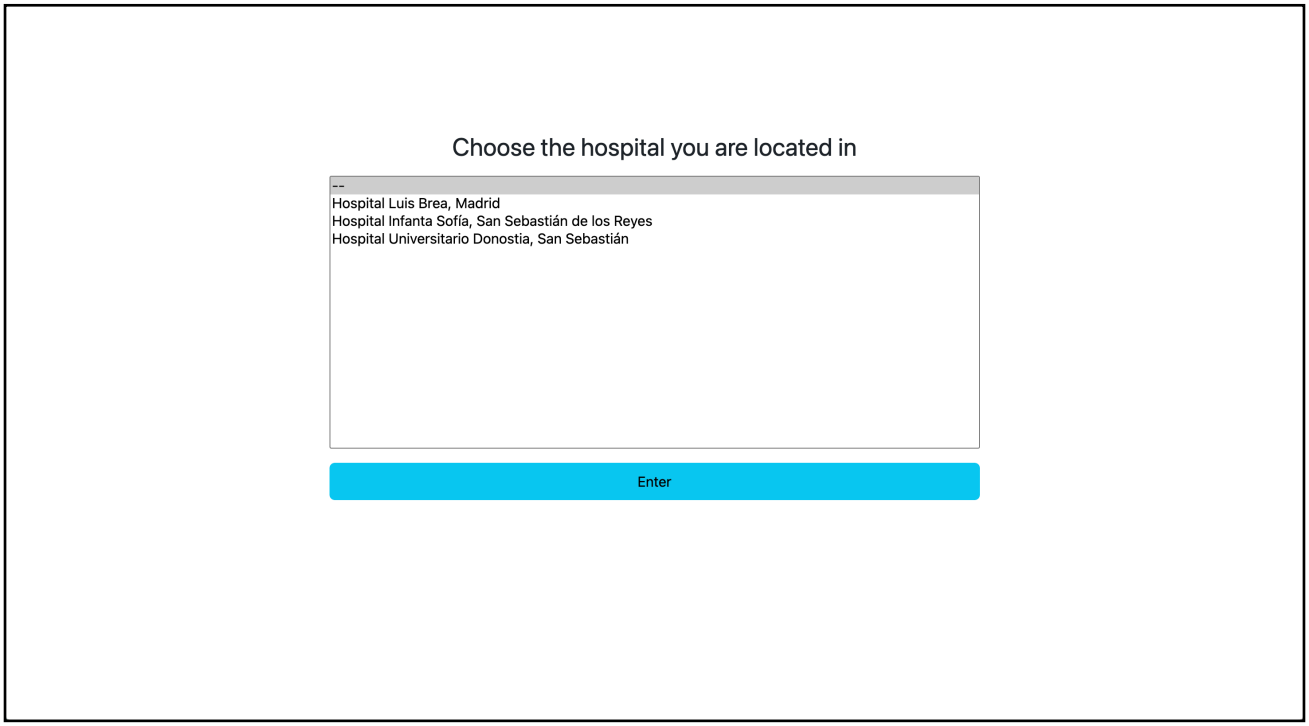
Source: Own elaboration

Figure 46: data.sql file

APPENDIX IV: USER MANUAL

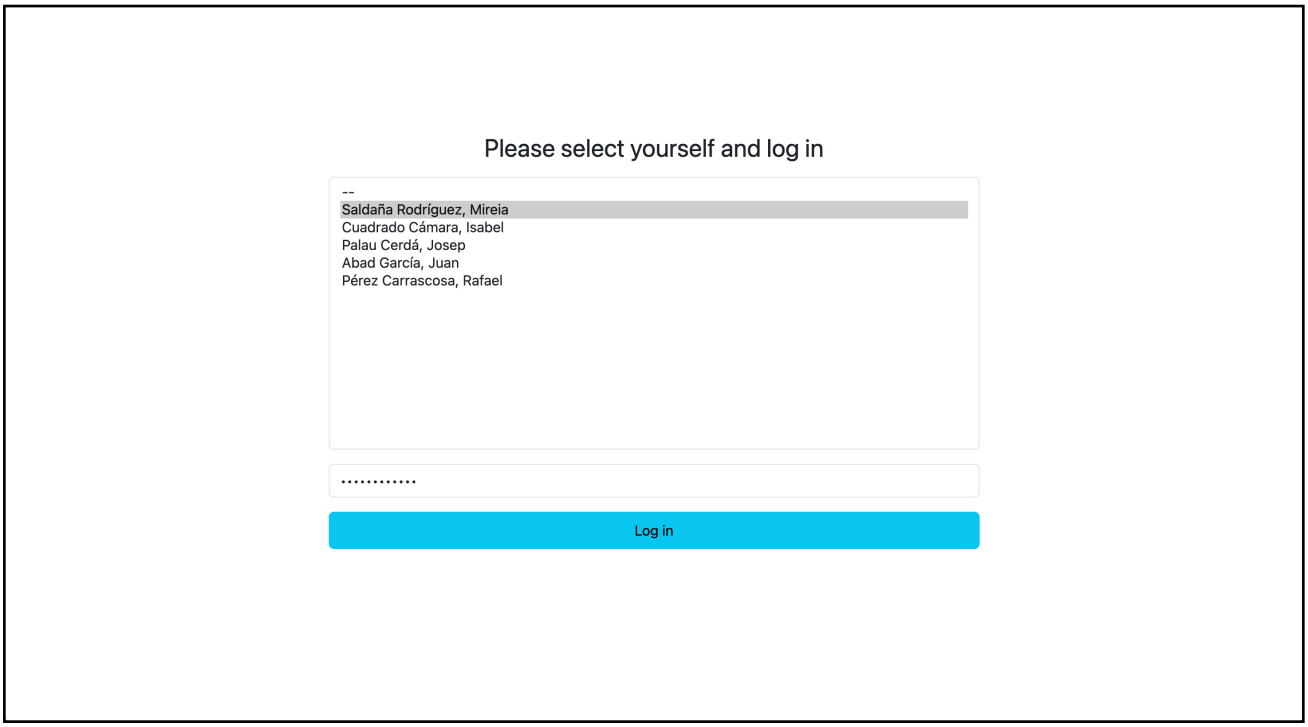
This section details and illustrates the user experience from the user and doctor perspectives, showing the functionalities of the application

Upon accessing the application, the doctor is greeted with the view shown in Figure 47. Here they will select the center they work in and progress to the proper login page (Figure 48), where they logging using their password. The main menu of the application shows the two options: 'Add' and 'View'. Both options, upon selecting them, reveal the user search view in Figure 49, where users can be searched with any combination of name and last name/s.



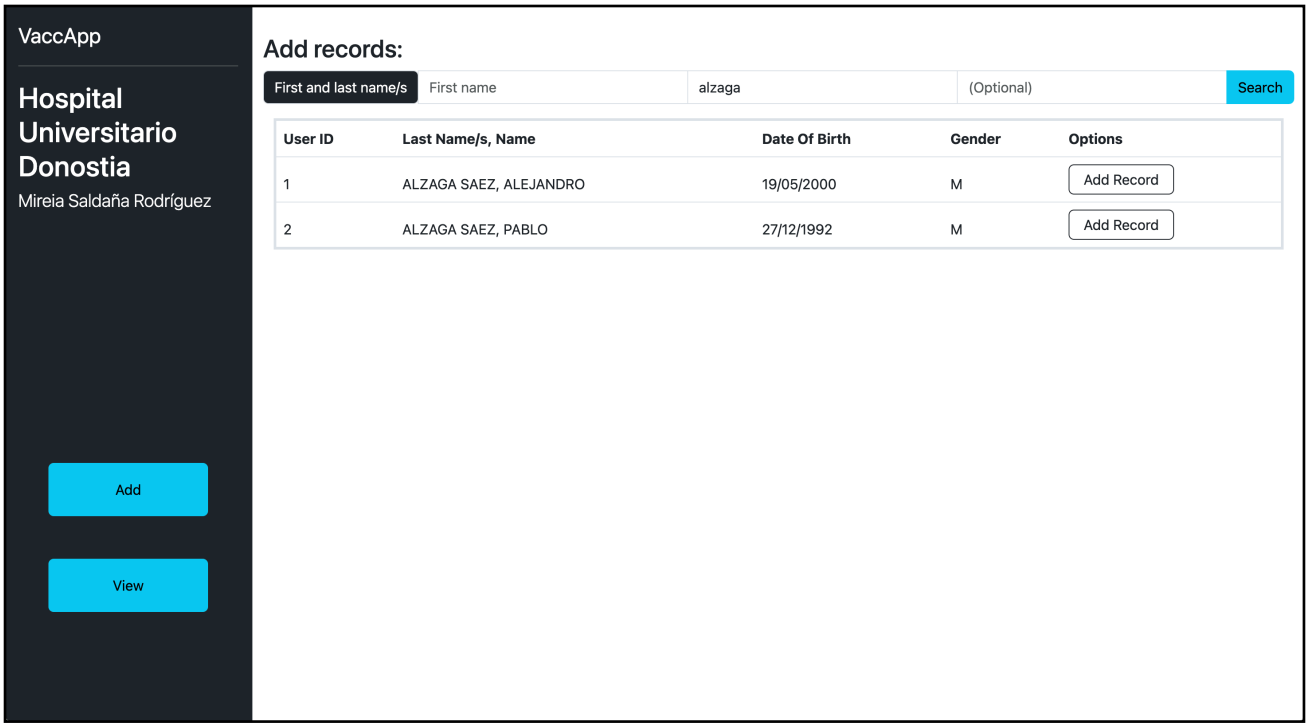
Source: Own elaboration

Figure 47: Hospital login view



Source: Own elaboration

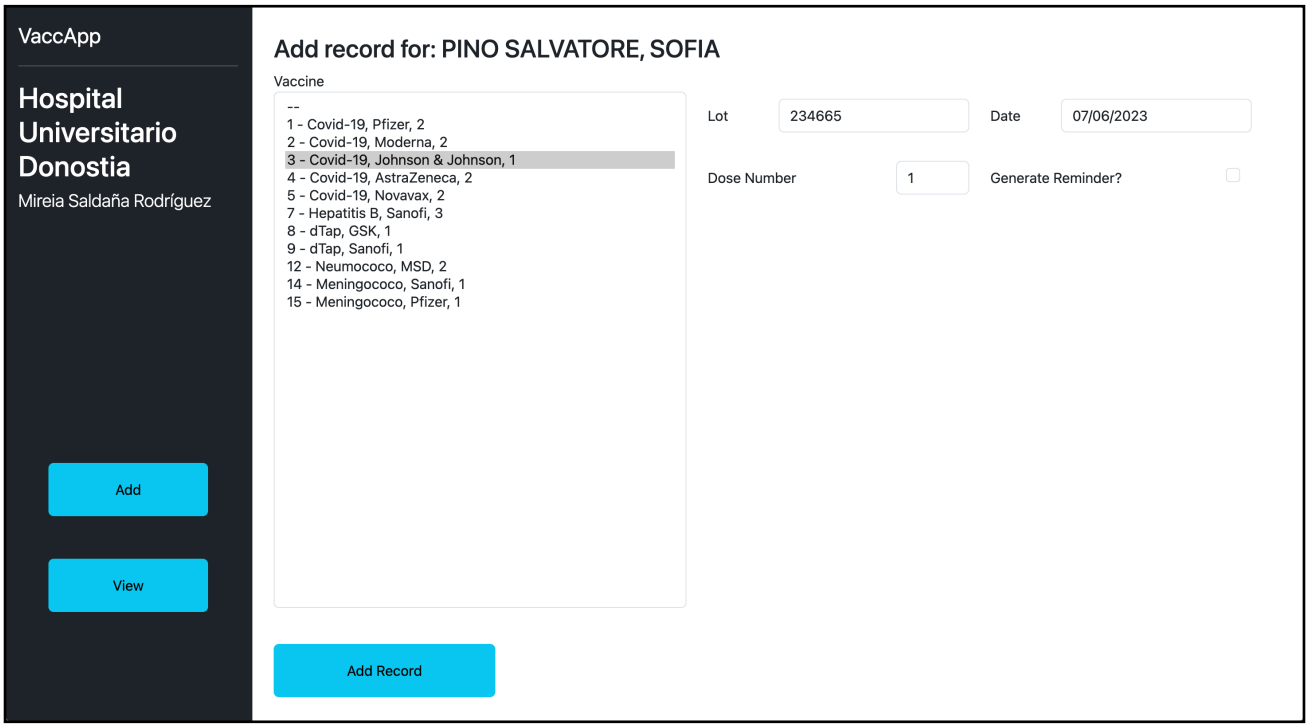
Figure 48: Doctor login view



Source: Own elaboration

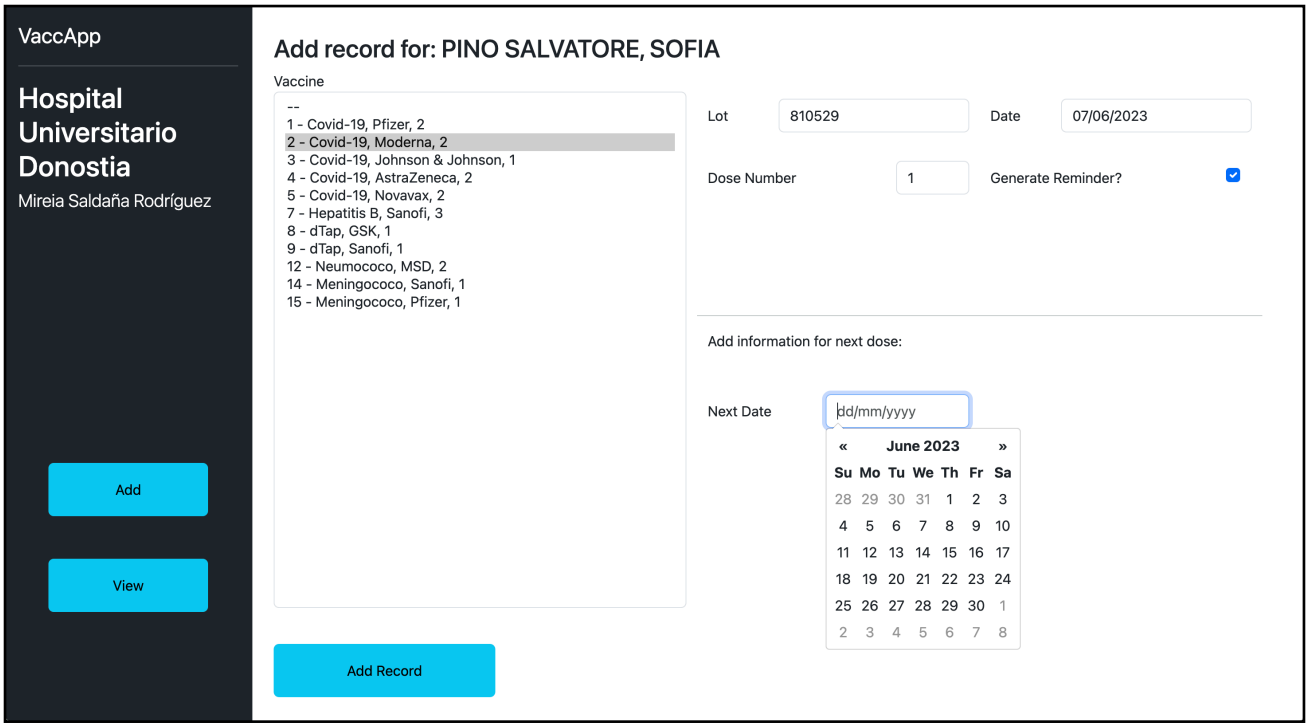
Figure 49: User search view

The ‘Add’ option is shown in Figure 50. The doctor has access to a form where they input the data to be stored in the record, which will also store the hospital and doctor’s name. Doctors have the option to generate future appointments as well by selecting the ‘Generate Reminder’ option, as shown in Figure 51. Upon adding the record the doctor receives feedback on whether or not the transaction has been successful (Figure 52).



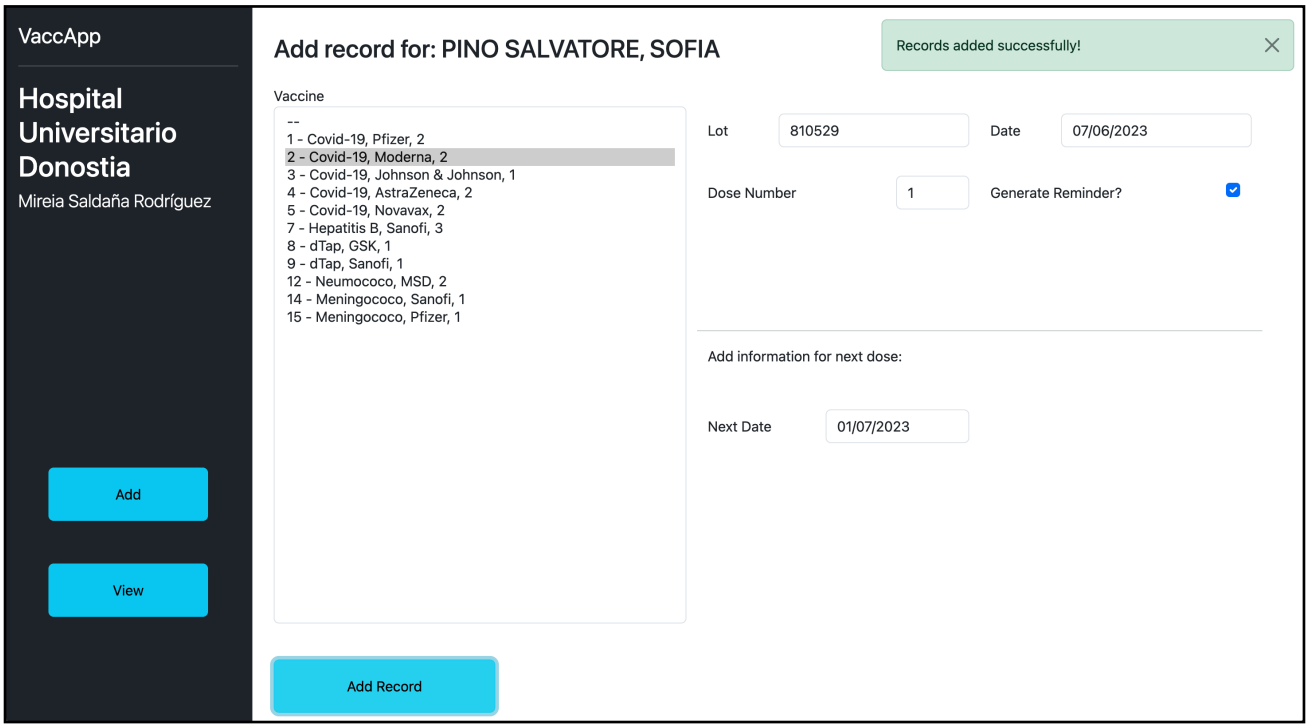
Source: Own elaboration

Figure 50: Add record view



Source: Own elaboration

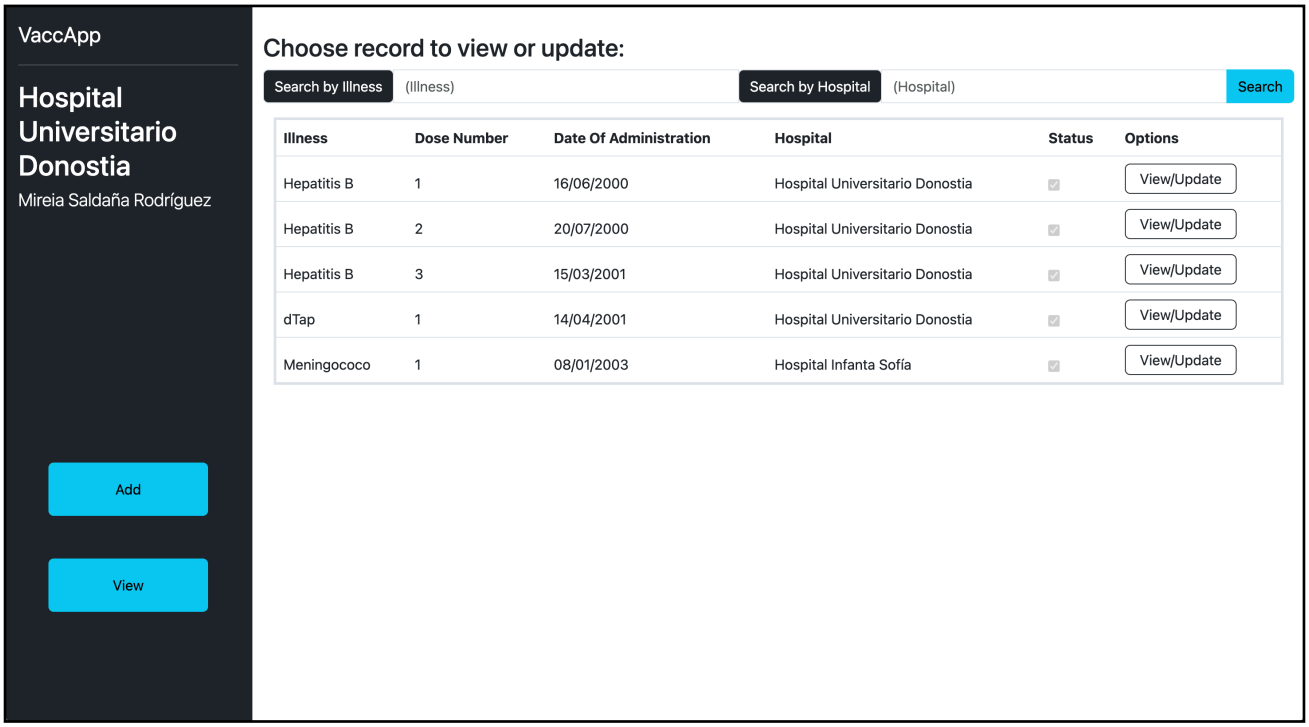
Figure 51: Add record and reminder for future appointment view



Source: Own elaboration

Figure 52: Successful record addition view

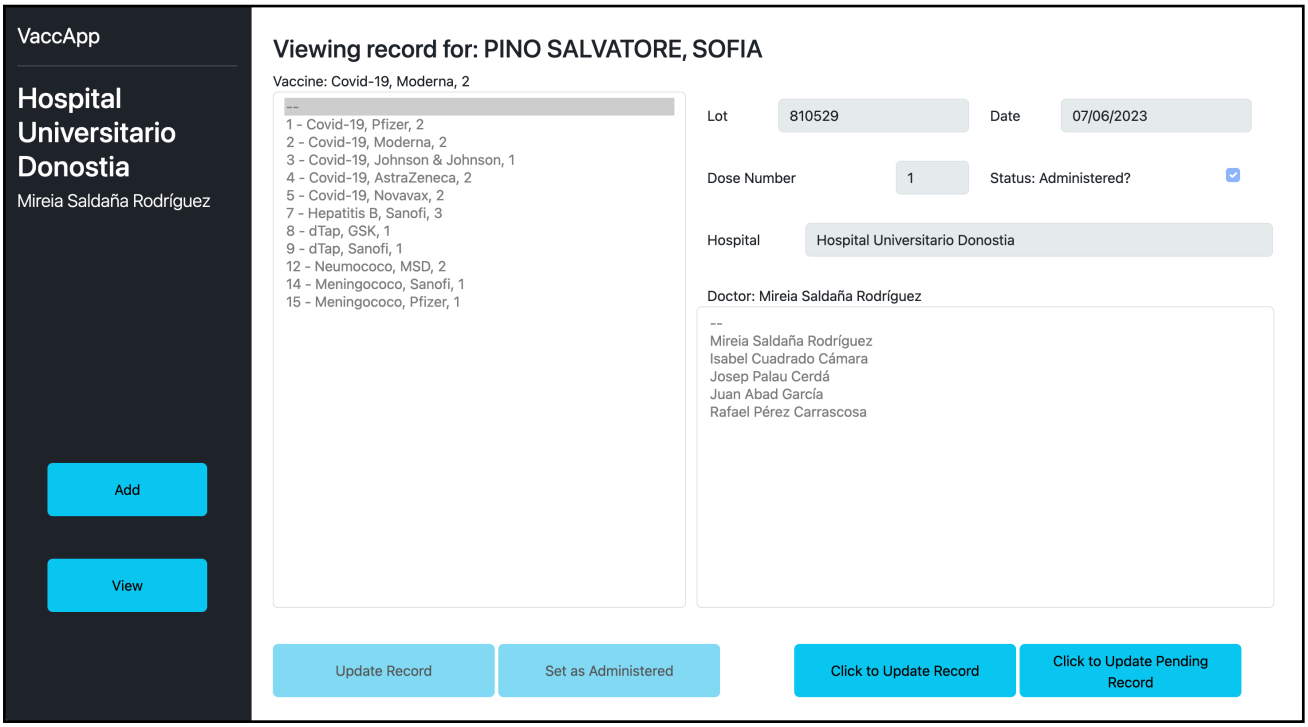
The ‘View’ option shows the view exhibited in Figure 53 after selecting the desired user. All records are shown, and can be filtered using the top search bars by either the hospital where the vaccine was administered, or the illness the vaccine targets.



Source: Own elaboration

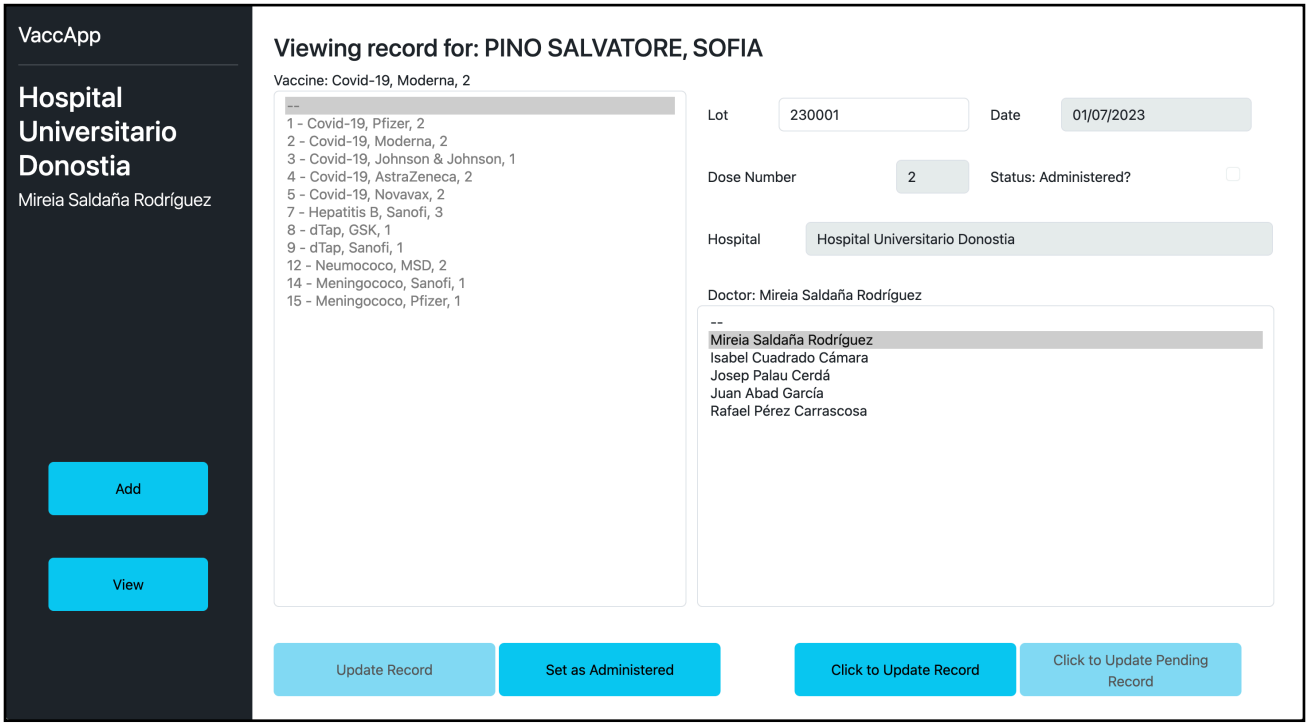
Figure 53: View records view

Every record has the option to view expanded information, which shows all information stored in the record (Figure 54). Here, doctors have the option to update a record as well, either a full update that allows the doctor to change any field (pressing ‘Click to Update Record’), or a pending record update (pressing ‘ Click to Update Pending Record’), where the doctor can only change a certain number of fields (Figure 55). Clicking ‘Update Record’ or ‘Set as Administered’ respectively completes the update transaction. Note vaccine and doctor fields have to be selected from the dropdown menu to complete a valid transaction.



Source: Own elaboration

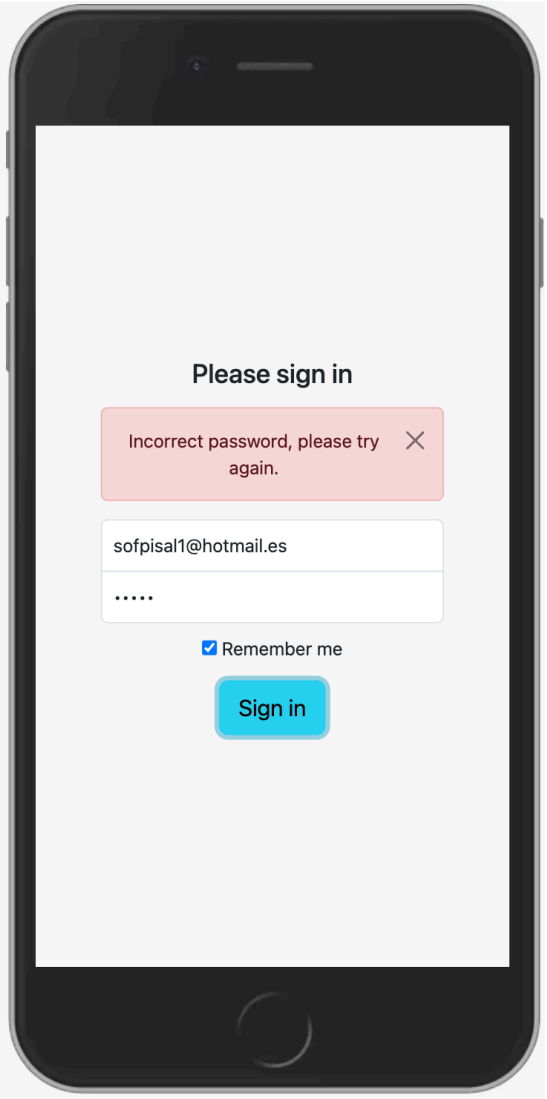
Figure 54: View expanded record information view



Source: Own elaboration

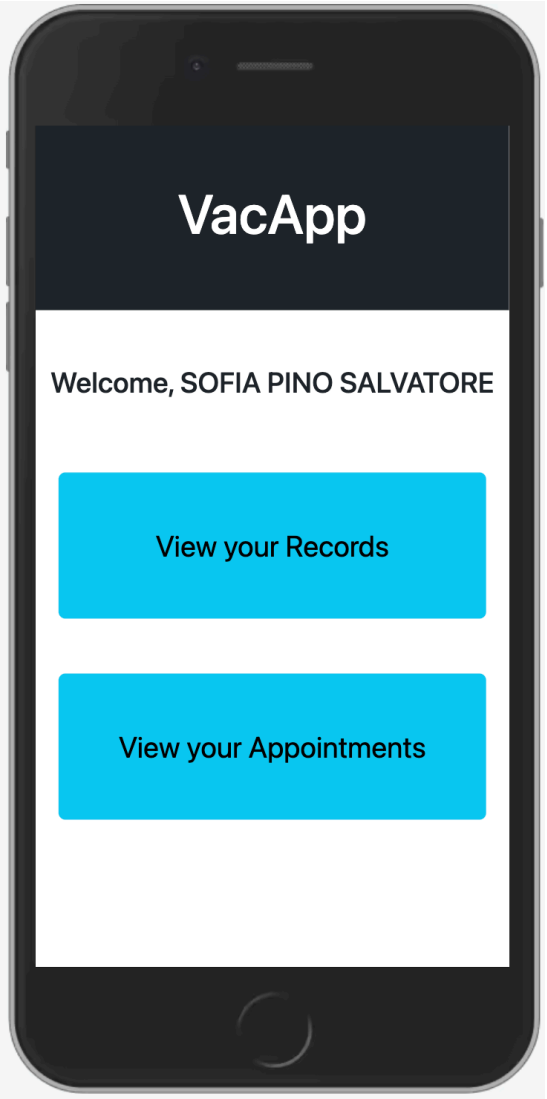
Figure 55: Update pending record view

The user is similarly greeted by a login page at the time of opening the app (Figure 56). Once the user enters their credentials correctly, the main menu reveals, shown in Figure 57.



Source: Own elaboration

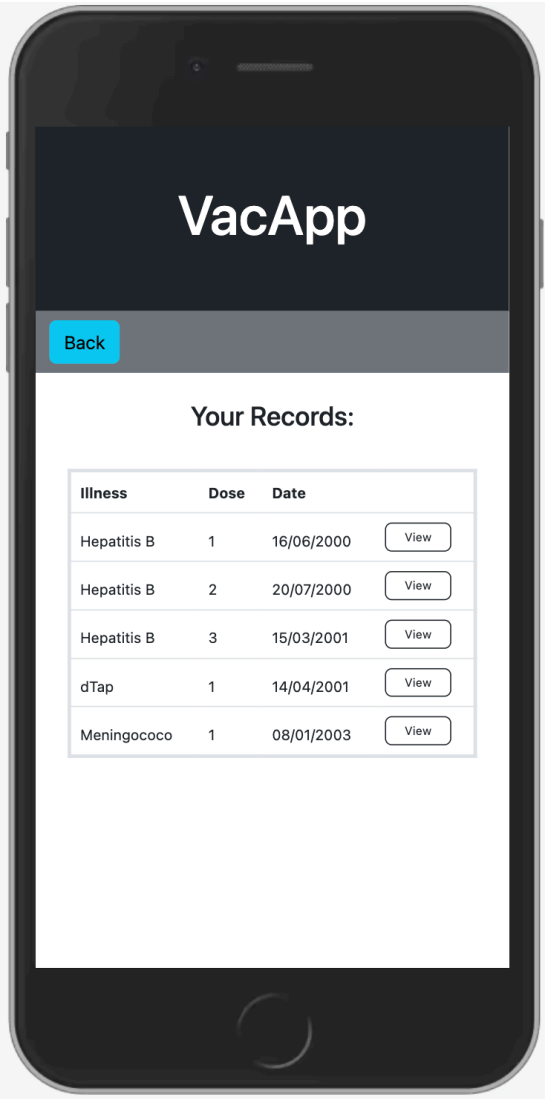
Figure 56: User login view



Source: Own elaboration

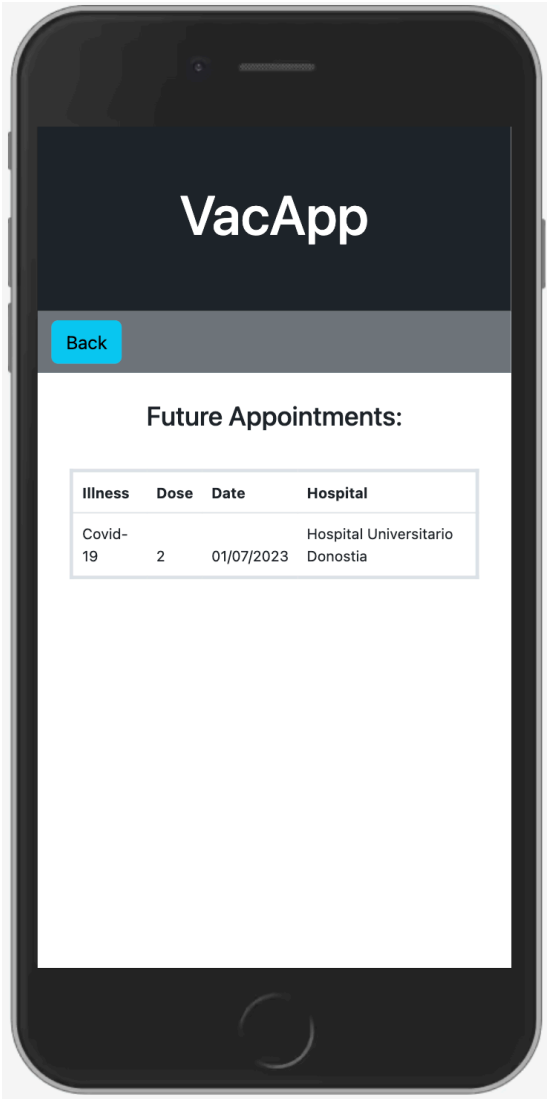
Figure 57: User main menu view

The user can either view their own records, which shows only the records associated with vaccines already applied, or future appointments, that shows records of pending vaccines. These views are shown in Figures 58 and 59 respectively. Finally, the user can also view a particular record's information in detail if they wish to do so; that view is included in Figure 60.



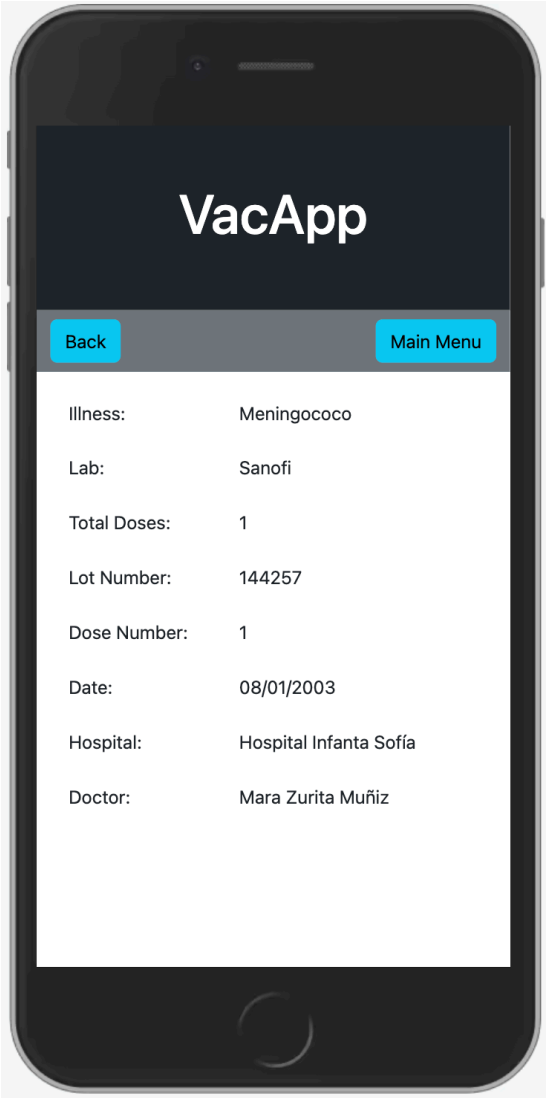
Source: Own elaboration

Figure 58: View user's own records view



Source: Own elaboration

Figure 59: Future appointments view



Source: Own elaboration

Figure 60: Extended record information view