electronics

MDPI

Article

BRAIN: Blockchain-Based Record and Interoperability Network

Sergi López-Sorribes [1,*], Josep Rius-Torrentó [2] and Francesc Solsona-Tehàs [1]

[1] Department of Computer Science, University of Lleida, 25001 Lleida, Spain; francesc.solsona@udl.cat
[2] Department of Economics, University of Lleida, 25001 Lleida, Spain; josep.riustorrento@udl.cat
* Correspondence: sergi.lopez@udl.cat

Abstract: In today's highly competitive landscape, effective information sharing is crucial for distinguishing successful entities from the rest. However, achieving seamless collaboration in this digital era remains challenging due to distrust among entities and the absence of a secure data exchange medium. These limitations hinder interactions, reducing competitiveness, especially for resource-constrained entities. To address these challenges, we introduce the Blockchain-based Record and Interoperability Network (BRAIN), a modular framework utilizing blockchain technology. BRAIN offers an innovative solution to foster collaboration, trust, and competitiveness. By leveraging blockchain's cryptographic techniques, it ensures secure and transparent data exchange, eliminating the need for intermediaries and thereby enhancing productivity and collaboration. BRAIN serves as a configurable foundation for domain-specific applications. BRAIN's key features include flexibility, efficiency, security, robustness, and scalability. It adapts to diverse applications, integrates seamlessly with existing systems, and ensures secure information sharing. In conclusion, BRAIN provides a foundational platform for entities to assert data ownership and grant access permission. Built on blockchain technology, it offers access traceability and data security and eliminates the need for trusted third parties. Its flexibility and scalability make it a valuable tool for enhancing data sharing. Future enhancements aim to address weaknesses and expand its use cases, promising a robust and versatile solution for secure and efficient data exchange.

Keywords: blockchain; hyperledger fabric; open source; scientific collaboration; modular; applicaton

Citation: López-Sorribes, S.; Rius-Torrentó, J.; Solsona-Tehàs, F. BRAIN: Blockchain-Based Record and Interoperability Network. *Electronics* **2023**, *12*, 4614. https://doi.org/10.3390/electronics12224614

Academic Editor: Mehdi Sookhak

Received: 11 October 2023
Revised: 4 November 2023
Accepted: 9 November 2023
Published: 11 November 2023

1. Introduction

In today's highly competitive world, the ability to share information effectively between entities has become a critical factor that sets successful competitors apart from the rest [1]. However, achieving seamless collaboration in this digital era remains a formidable challenge, mainly due to the prevailing distrust among entities and the lack of a trusted and controlled medium for secure data exchange. These limitations prevent interactions between entities, ultimately resulting in reduced competitiveness, particularly for those that lack adequate resources to facilitate such collaborations. By addressing these challenges and establishing secure data exchange platforms, entities can fully unlock the potential of effective collaboration and sustain their competitiveness in the long run.

To address these pressing concerns, we present a Blockchain-based Record and Interoperability Network (BRAIN), a sophisticated and modular framework that utilizes blockchain technology. BRAIN provides an innovative and smart solution for fostering collaboration, trust, and competitiveness amongst entities within a highly interconnected world. Our innovative approach, backed by blockchain technology, leads the way towards a secure and efficient data exchange platform that streamlines information sharing among entities. By leveraging blockchain's advanced cryptographic techniques, our system provides a high level of security and transparency in data exchange. Our solution removes the need for intermediaries, which simplifies and speeds up the sharing process, ultimately improving productivity and collaboration between entities. The blockchain-based platform

that we present in our investigation paper represents a significant step forward in the pursuit of secure and seamless data exchange.

In the present day, many researchers and developers have presented blockchain-based solutions to address specific domains, such as a decentralized software repository [2], a meteorological data service [3], a decentralized virus database [4], or a patent registration and trading system [5]. However, to the best of our knowledge, no one has yet presented a solution that serves as a configurable foundation for the development of such domain-specific applications.

BRAIN provides the following features without requiring three parties: flexibility, efficiency, security, robustness, and scalability. These are also the fundamental features that a global solution must offer.

One of the key strengths of our framework is its flexibility, designed to cater to a diverse spectrum of applications, and making it a highly adaptable solution that can effectively address various information-sharing needs. Whether entities are seeking to collaborate on research projects, securely exchange sensitive data, or simply streamline their internal processes, our system can be customized to meet their specific requirements.

Another strength is its efficiency in information sharing, ultimately leading to enhanced competitiveness and an improved performance. Entities can seamlessly integrate BRAIN into their existing systems without incurring significant disruptions related to high computation times and resource needs. Entities asynchronously upload relevant metadata onto the blockchain, thereby making other participants aware of the existence of these valuable data and prompting them to request access. Upon acceptance of the request by the data owner, the requester gains access to a designated access address on the blockchain. A peer-to-peer connection is subsequently established, validating the requester's credentials, thus ensuring data security, and thus providing secure information sharing between entities.

The distributed nature of our system ensures that service interruptions and global system crashes are prevented, resulting in a highly robust platform. However, while our system is designed with a focus on robustness, its availability can be impacted when nodes fail, as there is no built-in process for node recovery. Additionally, due to security restrictions, the system does not automatically duplicate information, which places the burden of data recovery on individual entities. As a result, it is important for entities to take proactive steps to ensure data recovery in the event of a node failure, as the system itself does not provide this capability.

Scalability is a key concern when it comes to data exchange, as the number of entities involved and the volume of data being shared per entity can grow exponentially. Our solution addresses this by utilizing a highly scalable architecture that can accommodate an unlimited number of entities and a virtually unlimited amount of data. This enables entities to share data without limits, and ensures that our system can continue to meet the changing needs of any organization, regardless of its size or complexity.

The rest of the paper is organized as follows: Section 2 offers a brief background of the evolution of blockchain technology and its key characteristics, as well as the most relevant topics and current legal considerations in data exchange. In Section 3, we present the architecture of BRAIN, along with its software and human workflows. Section 4 illustrates a generic experimental case to show BRAIN's adaptability to various environments. In Section 5, we employ the experimental case to discuss its usage and limitations. Finally, in Section 6, we conclude the article by addressing its limitations and potential future work.

2. Related Work

The concept of blockchain was introduced in 2009, alongside the advent of the first decentralized cryptocurrency, Bitcoin [6]. This revolutionary currency was built on blockchain technology, which enabled a decentralized control mechanism based solely on software-defined rules, rather than relying on traditional banking institutions or governmental

oversight. Three pivotal virtues of blockchain technology emerged: elimination of transaction intermediaries and immutable record-keeping of all transactions, allowing transparent validation and retrieval and resistance to third-party attacks, enhancing overall security.

At the core of Bitcoin's decentralized infrastructure lies a collaborative network of nodes [6,7].

Nodes in the Bitcoin network perform crucial tasks to ensure the integrity, security, and efficiency of the system. These tasks include facilitating, validating, and organizing transactions and maintaining the ledger [6,7]. Nodes must facilitate the transmission of digital currency between participants by verifying the authenticity and accuracy of each transaction. Nodes are also responsible for organizing and validating new transactions. This process involves verifying that each transaction meets the necessary criteria and grouping them into blocks that are then added to the existing blockchain. Finally, the nodes store copies of the entire blockchain, including all previous transaction blocks, over time. This redundancy ensures that the network has a secure and immutable ledger, preventing any individual or group from tampering with the transaction history.

The Bitcoin blockchain serves as a global ledger, recording only the bitcoins (hashcodes) of each user and their respective transactions. Consequently, all bitcoins hold the same practical value without differentiation.

The emergence of ColoredCoin [8] in 2013 marked a turning point. This initiative aimed to endow Bitcoin nodes with additional functionality, allowing for the existence of bitcoins that were clearly distinct and held a superior value compared to conventional bitcoins. ColoredCoin achieved this by associating these bitcoins with external assets, such as land or artwork, effectively allowing nodes implementing the ColoredCoin protocol to distinguish them unequivocally. This innovation empowered physical/virtual assets to be exchanged as if they were a native currency, while retaining all the aforementioned virtues. ColoredCoin thus laid the foundation for what could be referred to as a decentralized notary system.

The same year (2013), Russian–Canadian researcher Vitalik Buterin presented the initial concept of Ethereum [9], a blockchain that inherited and improved upon the ideas of its predecessors. His vision was to enhance existing blockchains, which hitherto served as ledgers and notaries, by creating a blockchain where individuals could develop decentralized programs. This vision became a reality with the official launch of Ethereum in 2015. In addition to preserving the characteristics of its predecessors and improving upon them, Ethereum introduced a unique feature, enabling the creation of decentralized programs through smart contracts. These smart contracts utilized a decentralized execution system, where small code fragments were executed based on predetermined rules. This groundbreaking development opened doors to extensive research beyond the realm of cryptocurrencies.

We focus on the field of information sharing. Several authors have recognized the considerable potential of technology in this area. For instance, the authors of [10] emphasized the necessity for users to retain control over their own private data, independently of third parties, due to the rising incidents compromising sensitive information and security. To achieve this goal, they proposed a protocol based on blockchain technology that automates access control without requiring third-party intervention. This protocol represents a step forward, improving protection and security in the context of sensitive information sharing. Azaria et al. presented in 2016 the MedRec [11], a decentralized electronic medical records management system based on blockchain technology, aimed to transfer control of personal data from medical institutions to patients. In 2018, Zaghloul et al. [12], following the same line of research, proposed an attribute-based distributed data sharing scheme to streamline data exchange processes among medical centers and grant control of the data to patients. In 2021, the authors of [13] highlighted the serious confidentiality and privacy issues faced by attribute-based distributed storage systems, and presented an access control system for resources based on blockchain, where only the data owner has the right to distribute the access key. The authors of [14] elucidate the inefficiencies of traditional

patent registration systems regarding their security and susceptibility to manipulation, among others, and advocate the utilization of blockchain technology to address these issues. Recently, the authors of [14], to ensure the data given by the student are legitimate and to secure the student's credit information, authentication, an access model, and a storage model were made using blockchain. More importantly, the student knows who accessed their data.

These examples represent just a few of the numerous scientists who have shown interest in technology focused on information sharing.

In recent times, the sharing of data has become an integral part of various industries, leading to the establishment of several regulations aimed at governing data-sharing practices. These regulations are crucial for protecting individual privacy, ensuring data security, and promoting responsible data usage. Let us delve into some notable examples of recent data-sharing regulations with relevant citations. The European Union's General Data Protection Regulation (GDPR) [15] is one of the most comprehensive data privacy regulations to date. The state of California in the United States introduced the California Consumer Privacy Act (CCPA) [16] to grant California residents increased control over their personal information. The CCPA allows consumers to know what data businesses collect about them, request the deletion of their data, and opt-out of the sale of their personal information (California Office of the Attorney General, n.d.). In the healthcare sector, the Health Insurance Portability and Accountability Act (HIPAA) [17] is a significant regulation that governs the sharing of protected health information in the U.S. The HIPAA establishes standards for the security and privacy of PHI to safeguard patients' sensitive medical data (U.S. Department of Health and Human Services, n.d.).

As one can observe, the legal landscape surrounding data exchange and privacy has significantly tightened in recent years, making it a pivotal consideration in software solution design. Consequently, numerous researchers have introduced diverse approaches to ensure compliance. For instance, in [18], the authors propose employing Egocentric Image Captioning instead of directly storing videos to safeguard patient privacy. In [19], the authors suggest an originality tracking system to prevent plagiarism. Another example can be found in [20], where the authors address the issues of the outdated privacy-preserving multiauthority attribute-based encryption (PPMA-ABE) schemes and propose a new decentralized solution to resolve them, to name a few.

3. Methods

In this section, we will provide an overview of the BRAIN system's functioning, including various design decisions and their rationale. We will elucidate its architecture, outlining the diverse technologies employed in its development. Finally, we will expound upon the distinct workflows from both a functional and user perspective.

You can find the BRAIN code available for download at this [21] GitHub repository.

3.1. Overview

BRAIN has been designed as a controlled information-sharing tool among entities, and its entire design is centered around being easily scalable and adaptable through programming to accommodate the data, technologies, and specific requirements of these entities.

3.1.1. Functionality

BRAIN employs straightforward guidelines for data sharing. Entities input the data they wish to share into BRAIN. Subsequently, the software uploads an identifier, a description, and an endpoint for data retrieval to the blockchain. Both the identifier and the description are provided by the entities themselves. When other entities query the blockchain, they can observe the presence of these data and request access by submitting their access request to the blockchain. If the data owner grants access, they change the status of the request in the blockchain. At this point, the requester can access the data by

making a request to the specified endpoint, which will be signed with their identifying certificate used in blockchain requests. The workflow described is visualized in Figure 1.

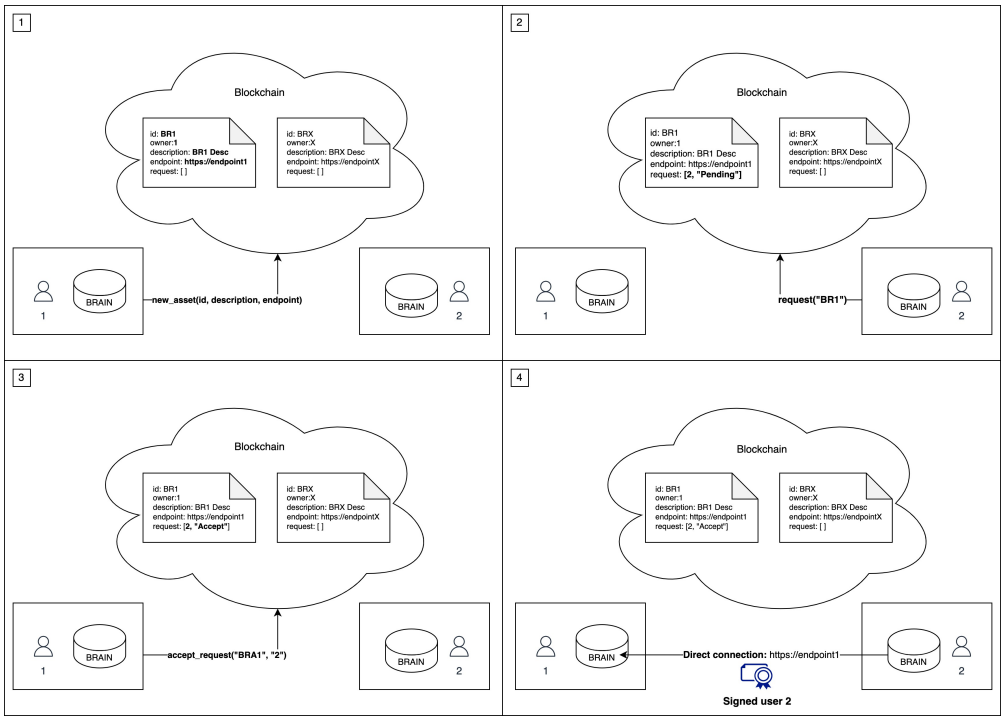

Figure 1. BRAIN functionality. [1] Metadata relevant to the information intended for sharing is uploaded onto the blockchain by entity 1. [2] Entity 2 make an access request for the aforementioned data within the blockchain. [3] Entity 1 grants approval for the request. [4] Entity 2 establishes a connection to the endpoint of entity 1, where the data is situated, and endorses the request using its certificate.

3.1.2. Design Decisions

In the development of the tool, three crucial decisions were made which are worth mentioning.

The first decision pertains to data sharing, where the choice was made to upload only metadata and access request control to the blockchain, relegating the bulk of the data to be served by an endpoint located on nodes that grant access only when previously authorized. This decision was made for several reasons. Firstly, scalability is a major consideration. Since the tool is designed to be a generic solution adaptable by entities from vastly different domains, it is understood that the volume of data may vary significantly. Uploading all these data to the blockchain, considering that they are replicated on each node, could substantially increase the storage requirements of the nodes. By limiting the upload to an identifier, description, and endpoint, we greatly mitigate this issue. Moreover, the fact that only metadata are uploaded promotes data diversity. From the blockchain's perspective, everything is managed uniformly, but the exposed data can be diverse and disparate since they are handled on each node as the entity prefers. Uploading only metadata also minimizes concerns related to security and legality that could be compromised by direct upload to the blockchain. This also eliminates the need to devise obfuscation or encryption mechanisms within the blockchain. Lastly, by dealing exclusively with metadata, we reduce the complexity of the smart contract to the minimum. This is essential, given that programming errors in smart contracts come at a high cost [22], depending on the blockchain technology used behind the scenes.

The second decision pertained to the architectural design, where it was decided to structure the application's functionalities into clearly distinct, interconnected layers. This division allowed the application to be easily adaptable to the needs of the entities, whether

they be related to data architecture, the entity's technological infrastructure, UI presentation, or required security. Modification would only be necessary in the layer interacting with the specific requirement, thus promoting flexibility and ease of adaptation.

Lastly, a decision was made to make how the application communicates with the blockchain asynchronous. While it was feasible to achieve acceptable response times for some private blockchains, BRAIN's goal of adaptability to various blockchain types, whether private or public, necessitated consideration of the slower consensus times of these blockchains. Therefore, it was decided that all communication with the blockchain should be entirely asynchronous. This approach allows users to interact with the application without any delays, with their actions subsequently reflected in the blockchain through an asynchronous process. This enhances the overall user experience.

3.2. Architecture

The application is based on a layered architecture, Figure 2, where each layer can be adapted or swapped according to the use case. These layers are blockchain, the connection interface, the asynchronous synchronizer, the administration application, endpoints, and the database.

Our implementation was primarily built using Python along with the Django framework, except for the part of the communication interface with the blockchain, where the GO language was employed for programming the smart contracts.

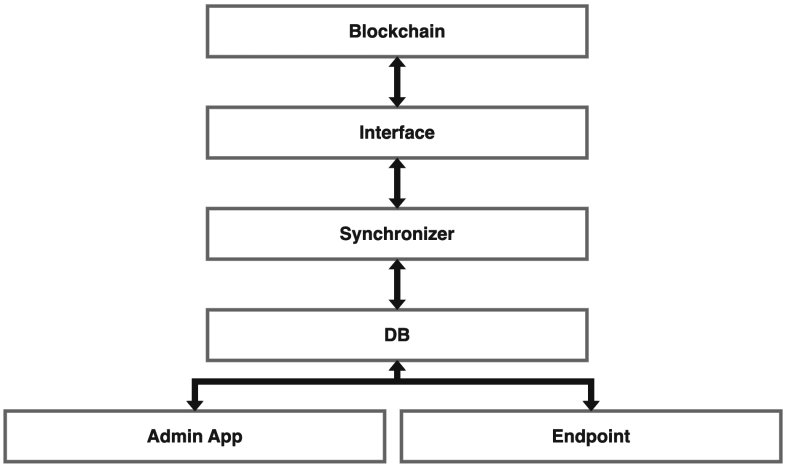

Figure 2. Architecture.

3.2.1. Blockchain

This layer serves as the core of the entire application, where a distributed record of which datum each participant has published and who requests or has access to those data are stored transparently for all participants. Thanks to the properties of blockchain technology, we ensure two important points. First, that this information will not be altered, and second, that there will be an exact traceability of who has had access to these data at all times. To enable the blockchain to store and secure these data, a logic is required, which will be implemented through smart contracts. These smart contracts implement the following functionalities:

- Store the required data structure for sharing—Table 1.
- Verify the existence of an asset.
- Retrieve metadata for all assets.
- Retrieve metadata for a specific asset.
- Publish an asset.
- Request access to an asset.
- Accept access requests to one's own assets.
- Deny access requests to one's own assets.

- Update metadata for one's own assets.
- Unpublish one's own asset.

In all of these functionalities, each asset has an owner, and only the owner of a given asset can modify, unpublish, or manage access requests to their assets.

Table 1. Share date required structure.

Field	Type	Description
Requests	map [string] [string]	Register the identifiers of the request owner and its status (Pending, Accepted, Denied).
Endpoint	string	Specify the location to make the request for obtaining that information.
ID	string	Unique identifier of the shared resource.
Owner	string	Unique identifier identifying the owner of the resource.
Description	string	Metadata of the object, used to enable people to search for the information.

In our implementation, this layer was developed using Hyperledger Fabric technology [23]. Hyperledger Fabric is an open-source software designed for developing applications and solutions based on blockchain technology. Its main features include a modular and versatile architecture. The choice of this technology was driven by the fact that Fabric is a private blockchain with a lighter and faster consensus mechanism compared to what we might find in a public blockchain, where the risk of data manipulation attempts is higher. This makes it ideal for the daily workflow of many entities that require a quick response time. Moreover, being a private blockchain, collaboration between multiple entities is more controlled and selective, providing an added layer of security and privacy.

In order to establish a connection between BRAIN and another blockchain, compatibility with smart contracts is imperative. It is necessary for the system to be capable of generating a smart contract endowed with the functionalities outlined in this section. Furthermore, a formal method should be applied to validate these functions, thereby mitigating the risk of security breaches [24].

3.2.2. Interface

This layer acts as an intermediary between the blockchain and the rest of the application. It is responsible for defining the user's identity within the blockchain and understands how to interact with and invoke the smart contracts defined in the previous section. In our implementation, this layer is written in Python and is designed to work with Hyperledger Fabric by reading the necessary parameters from the node's environment variables. If the blockchain layer were to be replaced with another one, a custom implementation would be required to enable the connection with the new blockchain while ensuring the established functions are maintained to preserve interoperability with the other layers.

3.2.3. Synchronizer

The synchronizer is the layer responsible for managing all asynchronous tasks of the application. Its primary functions include locally registering and updating the information exposed in the blockchain, as well as communicating the user's local changes to the blockchain, such as publishing new data and responding to access requests. In our implementation, this layer is written in Python, utilizing Django's data models. It is designed to work asynchronously, enhancing the user experience by avoiding response times that could be present in certain blockchains with control mechanisms that may slow down the process.

3.2.4. Database (DB)

This layer functions to locally record the metadata exposed in the blockchain, access requests to own data, own shared data, and the entities that have access to the data. To achieve this, it utilizes the following data models, Figure 3:

- GlobalData: This data model is used to store all identifiers along with their metadata exposed in the blockchain, along with the sharing status with the node.

- ABSSharedData: This model serves to share information with the rest of the participating entities. It is the model that should be extended to share different types of data.
- ExternalRequests: Requests from other entities to access the data are recorded in this model, along with their acceptance/denial status.

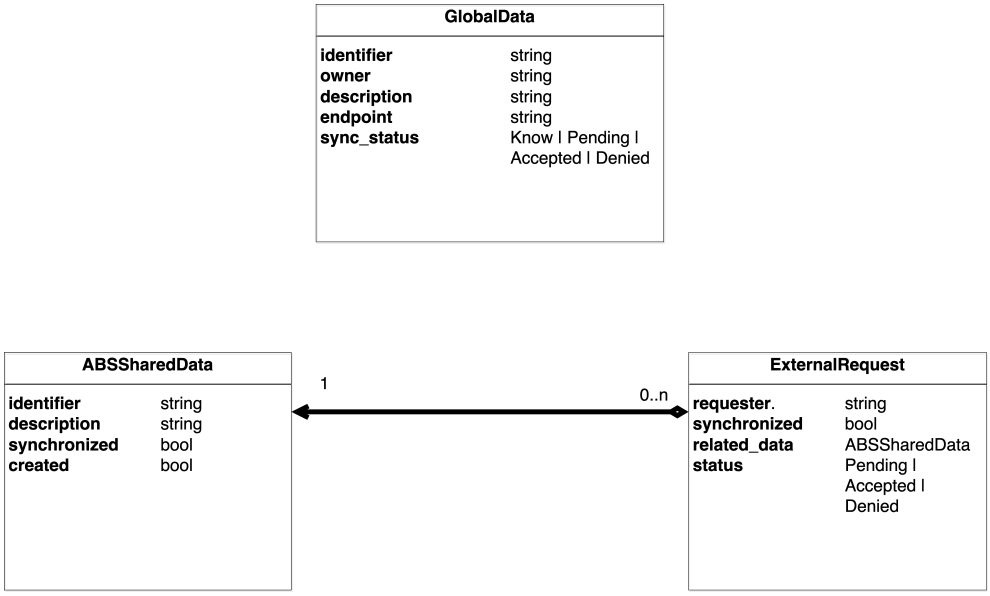

Figure 3. Database structure.

In our implementation, we use the SQLite database, a self-contained open-source relational database management system. We opted for SQLite due to its ease of use, but for a production implementation, it is recommended to switch to another database system, whether relational or non-relational, as long as the mentioned data structures are maintained.

3.2.5. Admin App

This layer comprises the visual interface through which the user interacts with the application, consisting of four menus. In the "All data" menu, all currently announced data in the blockchain can be queried, accessed if permitted, or access can be requested otherwise. The "External request" menu displays all requests received from other users, seeking access to data published by the user. It allows the user to either accept or deny these requests. The "Share data" menu is adaptable, containing a form for each distinct data model intended for sharing. Lastly, the "Own data" menu enables the user to modify the published data intended for sharing. This layer directly interacts with the database, and the synchronizer is responsible for reflecting these changes in the blockchain. In our implementation, this visual interface is developed using the built-in template functionality of Django, based on a model–view–template system. It is important to note that the provided software is built on a generic model, resulting in the data insertion form displaying only essential fields, such as "ID" and "Description", as they are shared with other users. Additional fields will be accessible only after access permission is granted. The software is designed for easy extensibility, allowing for the straightforward addition of custom forms, using Django forms.

3.2.6. Endpoint

This layer provides an Application Programming Interface (API) through which other users can access shared data. Its primary function is to manage data access requests, distributing the data only if the requesting entity has been identified as valid and has been granted access to the requested data. In our implementation, we have developed this layer

using the Django REST Framework technology and tailored it to work with Hyperledger Fabric certificates as a means of identifying the requests. The API responses return shared data in JSON format, which can be adapted based on the requested data model. These customized adaptations based on data models can be achieved using the serializable classes provided by Django REST Framework.

3.3. Workflow

In the application, we can identify several workflows, and for clarity, we will distinguish them from both the user's perspective and the operational perspective.

3.3.1. Workflow User Point of View

In the administrative section, the user will have access to four main actions: The first action is to query all published records, where the user can view the metadata published by other users and the type of access they have to those records. Depending on the status, the user can perform different actions, as there are four states:

- Know: this state is the initial state for any synchronized data for which no action has been taken, and it allows the user to request access.
- Pending: this state is visible when the user has requested access to certain data, but the other party has not responded to the request yet.
- Accepted: when the other party has accepted the access request, this state will appear, and clicking on the data will trigger the system to retrieve the data from the other node.
- Denied: if the other party has declined the access request, this state will be displayed, and it only allows the user to request access again.

The second action is data uploading. For each data model intended to be shared, the application provides a form that, upon saving, will synchronize the data with the rest of the users. The third action involves accepting or denying access requests to one's own data. Finally, the fourth action allows editing existing data of which the user is the owner.

3.3.2. Workflow Functional Point of View

If we look at the workflow from a more functional perspective, we can distinguish four main steps:

- Creation/Modification of Data: when the user utilizes a form to either share or modify data, the system locally records that the metadata of this model need to be synchronized and delegates this task to the synchronization layer.
- Querying Foreign Data: when access to foreign data is granted, the system makes a signed GET request with its identifying certificate to the endpoint layer of the other node, which, having granted prior access, serves the request.
- Requesting Data Access: when the user requests access to foreign resources, the system locally marks those data for the request and delegates the task to the synchronization layer.
- Acceptance/Denial of Requests: the received requests from the synchronizer are displayed, and when a user takes action on them, this action is locally marked and the communication task is delegated to the synchronizer. Additionally, if any requests are accepted, they are recorded to allow access to the endpoint layer.
- Synchronization: This flow, completely asynchronous to the user's actions, is responsible for synchronizing all the previously described actions to be replicated on the blockchain. It uploads or updates data in ownership, notifies accepted or denied requests, and updates the information, according to the blockchain, of foreign data.

4. Results

In this section, we will illustrate the system's adaptability by presenting a use case. We will explore the specific problem that needs to be addressed, how the system will be customized to solve it, and the operational procedures of various entities once the adaptation is complete. To ensure alignment with the paper's objectives, we will establish the following predefined assumptions for the use case:

- Institutions do not completely trust each other.
- There is a desire for cooperation among these institutions to enhance their efficiency and effectiveness.
- A Hyperledger Fabric network has been set up among them, with each institution having at least one node connected to the blockchain.
- The nodes have BRAIN installed, with only the code modification required to tailor it to their specific needs.

4.1. Context

In a locality, several hospitals have decided to collaborate by sharing information to provide more efficient patient care. The information to be shared includes the medical records of their patients. This way, if a patient treated at Hospital A seeks treatment at Hospital B, Hospital B can have advanced knowledge of the patient's existing data at Hospital A. This allows them to request access and avoid unnecessary tests or duplicative procedures.

4.2. Data

The data contained within medical records exhibit significant diversity, often extending to considerable lengths. Additionally, various models of medical records may exist. In this example, we will model the one corresponding to the following JSON structure, Listing 1. To model other record formats or expand their scope, one simply needs to replicate the steps outlined for the additional models.

Listing 1. Medical record.

```
{
    "Name": "John␣Doe",
    "DateOfBirth": "1980-05-15",
    "Gender": "Male",
    "SocialSecurityNumber": "123456789",
    "Address": "123␣Main␣Street,␣City",
    "Phone": "555-555-5555"
    "MedicalHistory": {
        "LastVisit": "2023-08-15",
        "Diagnosis": "Hypertension",
        "Medication":
            {
                "Name": "Losartan",
                "Dosage": "50mg",
                "Frequency": "Once␣a␣day"
            },
        "Allergies": "None"
    }
}
```

4.3. BRAIN Adaptation

In this section, we will outline the necessary program modifications required for adaptation. To guide us through this process, we will use Listing 2 as a reference, depicting the project's structure as of the time of writing this article.

Listing 2. Files to modify.

```
exchange_system
|
|-core
|   |- models.py
|   |- ...
|-endpoint
|   |- views.py
|   |- ...
|-frontend
|   |- forms.py
|   |- ...
|/
```

4.3.1. Models: Allow System to Recognize Your Data

To enable BRAIN to recognize the data models for sharing, they would be modeled within the code using the format employed by the Django framework in the "exchange_system/core/models.py" file. In this instance, we will model the "Expedient" model, Listing 3. To make it valid for sharing, it should inherit from the "ABSSharedData" class. By doing so, we would introduce the required fields into "models.py" see [25] for the full code.

Listing 3. models.py.

```
class Expedient(ABSSharedData):
    name = models.CharField(max_length=30)
    #...
class MedicalHistory(models.Model):
    last_visit = models.DateField()
    #...
class Medication(models.Model):
    name = models.CharField(max_length=50)
    #...
```

4.3.2. Endpoint/Serielizers: Allow Data Sharing with Other Members

It is possible that not all stored data should be visible, which is why in BRAIN, you need to define how your data will be communicated once you accept an access request. To achieve this, we will modify the file "exchange_system/endpoint/views.py" to add a serializer following the guidelines of Django REST Framework, Listing 4. Here is an example of the code, see [25] for full code:

Listing 4. views.py—ExpedientSerializer.

```
class ExpedientSerializer(serializers.ModelSerializer):
    class Meta:
        model = Expedient
        #...
```

Then, we would add the serializer to the mapping dictionary for the "Expedient" class, Listing 5.

Listing 5. views.py—Mapping dictionary.

```
serializers_by_class = {
    ABSSharedData: GenericSerializer,
    Expedient: ExpedientSerializer
}
```

4.3.3. Form: Allow User to Introduce Data to the System

In BRAIN, you have the option to define forms to facilitate the input of data to be shared by the end user. While this step is not strictly necessary as data can be entered directly into the database, we will illustrate it in case someone finds it useful. To achieve this, you would modify the file located at "exchange_system/frontend/forms.py". In this file, you can create a form using Django's FORMS guidelines that represents the data you want to input. You have to extend the "ABSSharedDataForm" class and add your class on the "CUSTOM_FORMS" list. Here is an example for your reference, Listing 6. See [25] for the full code:

Listing 6. views.py—ExpedientForm.

```
class ExpedientForm(ABSSharedDataForm):
    friendly_name = 'Expedient'
    medical_history_last_visit = forms.DateField(label="last_visit
        ")
    #...

CUSTOM_FORMS = [ABSSharedDataForm, ExpedientForm]
```

4.4. BRAIN Use

Once the code has been customized for the specific use case, users can access the local node's URL to perform searches (Figure 4), upload new data (Figure 5), request access to data from other hospitals (Figure 6), manage requests from other hospitals (Figure 7), and query data to which they have access (Figure 8). All of these operations are orchestrated by BRAIN and supported by the blockchain.

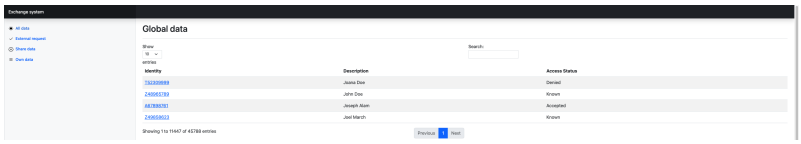

Figure 4. Frontend: Perform searches.

Figure 5. Frontend: Upload new data.

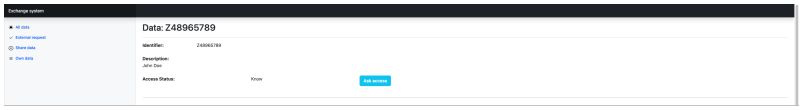

Figure 6. Frontend: Request access.

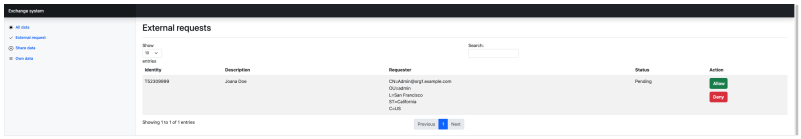

Figure 7. Frontend: Manage requests.

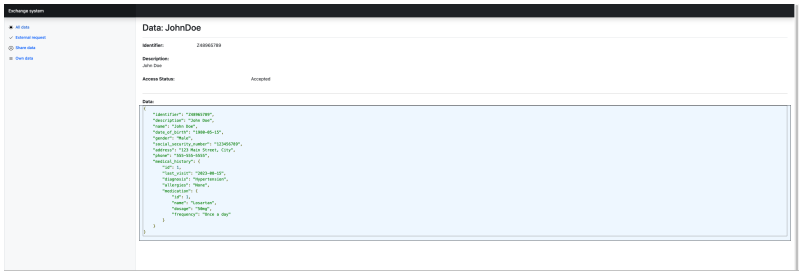

Figure 8. Frontend: Query data.

5. Discussion

The BRAIN framework has been designed and implemented with the intention of serving as a foundational platform for subsequent projects. This framework is available for free download [21], exhibits ease of adaptability to diverse data types, and, due to its modular structure, can be seamlessly integrated into infrastructures different from its original design.

In the results section, BRAIN has been used to implement an experimental use case for sharing medical records that significantly enhances operational efficiency within hospitals. The meticulous and secure tracking of this data sharing is achieved through the use of blockchain technology. This technology guarantees transparency in accessing the information, rendering any tampering attempts futile. This solution serves as an illustration of how BRAIN can be employed to provide a swift response to the presented problem. However, it is crucial to acknowledge that this solution may need additional requirements, such as the need for access control to the web interface of the nodes or legal requirements stemming from the scope of application, i.e., the medical domain in this case. As previously emphasized, this software is fundamentally designed to serve as a foundational framework for subsequent software solutions. Accordingly, it encompasses only the essential and minimal functionalities necessary for its core operations, deliberately deferring specific features to be tailored and implemented according to the specific use cases of the entities that choose to adopt it.

We are aware that there are many other blockchain solutions, such as the one proposed in [2], which suggest the use of blockchain for implementing a software repository. This addresses the issue of maintaining conventional repositories, often relying on limited donations, and provides us with a functional example of blockchain where they migrate the entire PyPi catalog. We also acknowledge the contributions of Nicolas Lopez and their colleagues [3], who have developed a solution aimed at sharing meteorological data at no cost, built on a public blockchain to facilitate free access to these data. Additionally, Ahmed Lekssays et al. [4] proposed a blockchain-based malware containment framework for the Internet of Things (IoT) as a countermeasure against malware exploiting the low-security measures of IoT devices, claiming it achieves a drastic reduction in malware propagation on the IoT network. These are just a few examples of similar solutions, to name a few.

However, to the best of our knowledge, we have not found any open-source software that can serve as a solid and modular foundation for implementing a data sharing solution

with security, traceability, and tamper resistance, as BRAIN does, ensuring the properties of flexibility, efficiency, robustness, and scalability.

6. Conclusions

The application presented in this study represents a foundational platform where multiple entities can assert data ownership and grant access to other parties. This framework is built upon blockchain technology, offering several advantages such as access traceability, data manipulation security, and the elimination of the need for a trusted third party.

BRAIN emerges as a versatile foundation that allows entities to easily tailor it to their specific requirements, thanks to its independently layered structure. This flexibility even extends to the ability to switch between various technologies within the initial solution, as blockchain or database, without impacting the overall system. As demonstrated in the results section, BRAIN seamlessly adapts to real-world challenges, exemplified by its successful application in streamlining the sharing of medical records among multiple local hospitals, thereby enhancing operational efficiency.

Looking ahead, our development team envisions leveraging BRAIN's layered architecture to introduce an alternative blockchain layer and interface capable of interacting with a public blockchain, such as Ethereum. This prospect holds great promise, as public blockchains enable the exploration of use cases beyond the scope of private blockchains, such as Hyperledger Fabric.

We are also committed to enhancing the endpoint layer. While it is currently functional, it represents a potential weak point in the system, as a node failure could render all the information hosted by that node inaccessible. We acknowledge the importance of addressing this limitation based on specific use cases. Additionally, we plan to offer an optional configuration to work with the InterPlanetary File System [26] (IPFS), thereby mitigating the risk of a single point of failure.

Furthermore, we are considering the addition of an optional security layer that would encompass the entire web interface of the application. This enhancement would cater to developers who find common security measures sufficient for their use cases and prefer not to implement bespoke security solutions.

Our ongoing efforts will focus on decoupling data ownership from the nodes where data originate. This will enable the publication of data under different identities from a single node, a functionality not currently supported by the system.

Finally, after addressing the aforementioned aspects, BRAIN will be ready for deployment in real-world scenarios to showcase the system's adaptability.

Author Contributions: Conceptualization, S.L.-S.; Methodology, S.L.-S.; Software, S.L.-S.; Validation, J.R.-T. and F.S.-T.; Investigation, S.L.-S., J.R.-T. and F.S.-T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministerio de Ciencia e Innovación under contract PID2020-113614RB-C22 funded by MCIN/AEI/10.13039/501100011033. Furthermore, the authors are members of the research group 2017-SGR363, funded by the Generalitat de Catalunya.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code for BRAIN can be accessed at https://github.com/slopez1/exchange_system (accessed on 8 November 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, S.; Li, Y.; Li, G.; Yu, H.; Hao, B.; Song, J.; Fan, J. An Improved Data Provenance Framework Integrating Blockchain and PROV Model. In Proceedings of the 2020 International Conference on Computer Science and Management Technology (ICCSMT), Shanghai, China, 20–22 November 2020; pp. 323–327. [CrossRef]

2. Costa, F.Z.D.N.; De Queiroz, R.J.G.B.; Bittencourt, G.P.; Teixeira, L. Distributed Repository for Software Packages Using Blockchain. *IEEE Access* **2022**, *10*, 112502–112514. [CrossRef]
3. Lopez, N.; Agbu, A.; Oloyede, A.; Essien, E.; Eze, A.; Mhambe, C. Software tool to store IoT device data onto a blockchain. *Softw. Impacts* **2023**, *16*, 100511. [CrossRef]
4. Lekssays, A.; Carminati, B.; Ferrari, E. MalCon: A blockchain-based malware containment framework for Internet of Things. *Comput. Netw.* **2023**, *233*, 109853. [CrossRef]
5. Hu, J.; Zhu, P.; Qi, Y.; Zhu, Q.; Li, X. A patent registration and trading system based on blockchain. *Expert Syst. Appl.* **2022**, *201*, 117094. [CrossRef]
6. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: http://www.bitcoin.org/ (accessed on 8 November 2023).
7. Narayanan, A.; Bonneau, J.; Felten, E.; Miller, A.; Goldfede, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*; Princeton University Press: Princeton, NJ, USA, 2016; p. 336.
8. Rosenfeld, M. Overview of colored coins. *Comput. Sci.* **2012**, *41*, 94.
9. Wood, G.; Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Proj. Yellow Pap.* **2014**, *151*, 1–32.
10. Zyskind, G.; Nathan, O.; Pentland, A.S. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 180–184. [CrossRef]
11. Azaria, A.; Ekblaw, A.; Vieira, T.; Lippman, A. MedRec: Using Blockchain for Medical Data Access and Permission Management. In Proceedings of the 2016 2nd International Conference on Open and Big Data (OBD), Vienna, Austria, 22–24 August 2016; pp. 25–30. [CrossRef]
12. Afifi, M.H.; Zaghloul, E.; Li, T.; Ren, J. UBNB-PPDP: Utility-Boosting Negotiation-Based Privacy Preserving Data Publishing. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [CrossRef]
13. Wang, X.; Zhou, Z.; Luo, X.; Xu, Y.; Bai, Y.; Luo, F. A Blockchain-Based Fine-Grained Access Data Control Scheme With Attribute Change Function. In Proceedings of the 2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI), Atlanta, GA, USA, 18–21 October 2021; pp. 348–356. [CrossRef]
14. Maheswari, M.; Prasath, S.; Rajesh, R.; Ramalakshmi, D. Blockchain-based Access Control Model for Student Academic Record with Authentication. In Proceedings of the 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 17–19 May 2023; pp. 1411–1414. [CrossRef]
15. *Regulation (EU) 2016/679*; Technical Report, European Union General Data Protection Regulation (EU GDPR). European Union: Maastricht, The Netherlands, 2016.
16. *California Consumer Privacy Act (CCPA)*; Technical Report; California Office of the Attorney General: Sacramento, CA, USA, 2016.
17. U.S. Department of Health and Human Services. Available online: https://www.hhs.gov/ (accessed on 8 November 2023). .
18. Qiu, J.; Lo, F.P.W.; Gu, X.; Jobarteh, M.L.; Jia, W.; Baranowski, T.; Steiner-Asiedu, M.; Anderson, A.K.; Mccrory, M.A.; Sazonov, E.; et al. Egocentric Image Captioning for Privacy-Preserved Passive Dietary Intake Monitoring. *IEEE Trans. Cybern.* **2023**, 1 –14. [CrossRef] [PubMed]
19. Zhu, P.; Hu, J.; Li, X.; Zhu, Q. Using Blockchain Technology to Enhance the Traceability of Original Achievements. *IEEE Trans. Eng. Manag.* **2023**, *70*, 1693–1707. [CrossRef]
20. Han, J.; Susilo, W.; Mu, Y.; Zhou, J.; Au, M.H.A. Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 665–678. [CrossRef]
21. López Sorribes, S. BRAIN: Blockchain-Based Record and Interoperability Network. 2023. Available online: https://github.com/slopez1/exchange_system (accessed on 8 November 2023).
22. Abdellatif, T.; Brousmiche, K.L. Formal Verification of Smart Contracts Based on Users and Blockchain Behaviors Models. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–5. [CrossRef]
23. Hyperledger Fabric: Open Source Blockchain Technologies. Available online: https://github.com/hyperledger/fabric (accessed on 8 November 2023) .
24. Krichen, M.; Lahami, M.; Al–Haija, Q.A. Formal Methods for the Verification of Smart Contracts: A Review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022; pp. 1–8. [CrossRef]
25. López Sorribes, S. BRAIN: Blockchain-Based Record and Interoperability Network (Medical Example). 2023. Available online: https://github.com/slopez1/exchange_system/tree/medical (accessed on 8 November 2023).
26. Benet, J. Ipfs-content addressed, versioned, p2p file system. *arXiv* **2014**, arXiv:1407.3561.