Informatics

Datensouveräne und sichere Verarbeitung personenbezogener Daten durch den Einsatz von Solid in Datenplattformen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

066 645 Data Science

eingereicht von

Tobias Hajszan, BSc
Matrikelnummer 11776172

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Dr.techn. Mag. Tomasz Miksa
Mitwirkung: Dipl. Ing Moritz Staudinger

Wien, 31. März 2025

Tobias Hajszan   Tomasz Miksa

Using Solid in Secure Data Platforms to support Data Sovereignty and Privacy Preserving Computation of Personal Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

066 645 Data Science

by

Tobias Hajszan, BSc
Registration Number 11776172

to the Faculty of Informatics

at the TU Wien

Advisor: Dr.techn. Mag. Tomasz Miksa
Assistance: Dipl. Ing Moritz Staudinger

Vienna, March 31, 2025

Tobias Hajszan   Tomasz Miksa

Erklärung zur Verfassung der Arbeit

Tobias Hajszan, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 31. März 2025

Tobias Hajszan

Danksagung

Ein großes Danke gebührt Dr.techn. Mag. Tomasz Miksa für seine Anleitung, sein umfangreiches Fachwissen und seine große Geduld mit mir. Seine Ideen und Feedback haben die Richtung und Klarheit meiner Arbeit deutlich geprägt.

Ein ganz besonderer Dank gilt Moritz, meinem Co-Betreuer und engen Freund, der mich während dieses Prozesses mehr unterstützt hat als jeder andere. Es gelingt ihm auf bemerkenswerte Weise, konstruktive Kritik so zu formulieren, dass ich (meistens) motiviert war weiter zu machen und genau verstanden habe, was verbessert werden musste, und vor allem warum. Obwohl er einer der beschäftigtsten Menschen ist, die ich kenne, hat er sich dennoch immer Zeit genommen, jede meiner Fragen zu beantworten, auf jede Nachricht prompt zu reagieren und generell jede Bitte zu erfüllen – dafür bin ich ihm sehr dankbar.

Mein Dank gilt auch Sotiris, meinem griechischen Gott und Teil des WellFort-Teams, der mir großzügig beim Aufbau von Experimenten geholfen und wertvolle Einblicke in die Architektur ihrer Plattform gegeben hat. Danke für deine Hilfsbereitschaft, ohne Zögern oder Umstände.

Marlene, meiner Lebensgefährtin, danke ich dafür, dass sie in den Höhen und Tiefen dieser Reise stets an meiner Seite stand. Deine emotionale Unterstützung, deine Geduld und deine Motivation gaben mir die Kraft, weiterzumachen, als es schwierig war. Du hast mich mehr als einmal aus der Verzweiflung herausgeholt. Die Herausforderungen des Lebens ohne dich zu bestreiten will ich mir gar nicht vorstellen.

Ich danke außerdem meiner Familie (die doch aus recht vielen Menschen besteht) und meinen Freund:innen für ihren anhaltenden Zuspruch und dafür, dass sie mich immer wieder daran erinnert haben, während dieser intensiven Zeit auch mal abzuschalten. Shout-Out an euch!

Abschließend möchte ich die Rolle von generativer KI bei der sprachlichen Verbesserung dieser Arbeit anerkennen. Die Technologie diente als zusätzliches Werkzeug zur Verbesserung der Klarheit und Kohärenz des Textes und trug so zur Gesamtqualität der Arbeit bei.

Acknowledgements

I would like to express my sincere gratitude to Dr.techn. Tomasz Miksa for his invaluable guidance, expertise, patience and support throughout the course of this thesis. His feedback and critical insights helped shape the direction and clarity of my work.

A very special thanks goes to Moritz, my co-supervisor and close friend, who supported me more than anyone else during this process. He manages to be incredibly helpful and provide critique in a way that (mostly) left me motivated and clear on what needed to be improved and why. And although this man is one of the busiest humans I know, he always found time to respond to every question and fulfill every request I had, an effort I deeply appreciate.

I would also like to thank Sotiris, my Greek god and member of the WellFort team, for generously assisting me with experiment setups and providing helpful insights into their architecture, always willing to help without hesitation or delay.

To Marlene, my partner, thank you for standing by my side through the ups and downs of this journey. Your emotional support, patience, and motivation gave me the strength to keep going when things got difficult. You've pulled me out of despair more than once, and your support made all the difference.

I would also like to thank my family (which are quite a lot of people) and friends for their ongoing encouragement and for reminding me to find balance during this lengthy period. Your belief in me has meant more than words can express.

Finally, I acknowledge the role of generative AI in refining the text of this thesis. The technology served as an additional tool for enhancing the clarity and coherence of the content, thereby contributing to the overall quality of the work.

Kurzfassung

Datenanalysen müssen das Bedürfnis nach einer sicheren und sinnvollen Verarbeitung sensibler personenbezogener Daten mit dem Bedürfnis nach Kontrolle über die eigenen Informationen sorgfältig in Einklang bringen. *Solid* (Social Linked Data) ist ein offenes Protokoll, mit dem Nutzer ihre Daten in geschützten, persönlichen Pods speichern und verwalten können. *Solid* als dezentralen Datenspeicher in bestehende Infrastrukturen für datenschutzfreundliche Berechnungen zu integrieren, bleibt jedoch weitgehend unerforscht. Diese Arbeit untersucht, wie *Solid* effektiv in solche Plattformen integriert werden kann, um die dezentrale Datenfreigabe voranzutreiben und gleichzeitig den technischen Anforderungen an datenschutzbewusste Forschung gerecht zu werden.

Wir stellen als Lösung zur Integration das *Solid Gateway* vor, einen Vermittler, der in bestehenden Analyseumgebungen einen zustimmungsgetriebenen Zugriff auf *Solid Pods* ermöglicht. Das *Solid Gateway* implementiert anfragespezifische Authentifizierung und Autorisierung, verwaltet Zugriffsrechte und koordiniert den Abruf der Daten, die zur Erfüllung von Datenanfragen notwendig sind. Eine neuartige, granulare Datenfreigabestrategie, die Nutzerdaten in minimale, anfragebezogene Teilmengen umstrukturiert, ist der zentrale Bestandteil dieses Ansatzes. So werden überflüssige Datenübertragungen verringert und die Offenlegung irrelevanter Informationen begrenzt. Dadurch bleibt es den Datengebern möglich, die Kontrolle über ihre Daten zu wahren, während Analysen auf ihre gespendeten Daten durchgeführt werden können.

Unsere Evaluation bekräftigt, dass *Solid Pods* mit unserem *Solid Gateway* erfolgreich in bestehende Analyseumgebungen eingebunden werden können. Die Resultate belegen, dass die Datenexposition erheblich reduziert und die Performance beim Laden und Verarbeiten der Daten im Vergleich zu etablierten Verfahren verbessert wurde. Darüber hinaus zeigen wir, dass die von uns vorgeschlagene Lösung gegenüber der ursprünglichen WellFort-Architektur eine konkurrenzfähige Leistung beim Laden von Daten aus *Solid Pods* bietet und die Verarbeitungseffizienz erheblich verbessert. Die Evaluation stellt eine reproduzierbare Basis für künftige Studien und praktische Umsetzungen dar. Wir stellen ein konkretes, erweiterbares Design sowie Richtlinien zur Integration von *Solid* in bestehende Datenplattformen bereit und eröffnen Perspektiven für zukünftige Forschungen, wie etwa zur SPARQL-Integration, Validierung mit etablierten Datensätzen und der Anwendung von FAIR-Prinzipien innerhalb von *Solid*.

Abstract

Privacy-preserving data analysis must carefully balance the need for secure, meaningful computation on sensitive personal data with the fundamental rights of individuals to retain control over their information. *Solid* (Social Linked Data) presents an open protocol where users store and manage their data in personal, access-controlled pods. However, its potential for integration as a decentralized data store into existing infrastructures for privacy-preserving computations remains underexplored. This thesis addresses the question of how Solid can be effectively integrated into such platforms to support decentralized data sharing while meeting the technical requirements of privacy-aware research.

To address this, we propose the *Solid Gateway*, a mediator that facilitates consent-driven access to *Solid Pods* within existing analysis environments. The *Solid Gateway* introduces request-specific authentication and authorization, manages access permissions, and orchestrates the retrieval of only the data necessary to fulfill individual data requests. Central to this approach is a novel granular data-sharing strategy, which restructures user data into minimal request-specific subsets, thus reducing unnecessary data transfers and limiting the exposure of irrelevant information. This ensures that contributors retain sovereignty over their data, while allowing privacy-preserving analysis to operate on decentralized sources.

Our experimental evaluation, conducted on controlled artificial datasets, confirms the feasibility of our integration. The results demonstrate a significant reduction in data exposure while achieving improved data retrieval performance compared to existing approaches. Also, we compare our proposed solution against the WellFort architecture and demonstrate that our approach offers competitive fetch performance and significantly improves processing efficiency. Although the controlled nature of the evaluation limits comparability with existing platforms, it provides a reproducible foundation for future studies and practical deployments. This work contributes a concrete, extensible design for combining *Solid* with privacy-preserving computation, identifies key trade-offs between privacy, performance, and system complexity, and opens pathways for future research into SPARQL integration, validation with established datasets, and the application of FAIR principles within *Solid*.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement and Motivation

The success of machine learning and data-driven applications relies heavily on access to large, high-quality datasets. In sensitive domains such as healthcare and finance, however, the collection and use of personal data introduce complex ethical and legal challenges [LS22]. Data platforms have emerged as essential infrastructures to manage, store, and process such data. They provide controlled environments for data access and analytics, offering technical solutions to address concerns of privacy and security.

Despite their advantages, many existing data platforms adopt centralized architectures, which often operate as opaque data silos. These platforms typically limit the involvement of data owners, providing neither transparent insights into how personal data are processed nor mechanisms to understand or influence how analytical results are derived [JSYH22]. As a result, the user does not have direct oversight over the specific pieces of their data used or has a clear understanding of the results of the training process.

The CARE Principles (Collective Benefit, Authority to Control, Responsibility, and Ethics) propose a framework for more ethical data governance, emphasizing individual and community empowerment [CGFR+20]. Although initially developed in the context of Indigenous data sovereignty, these principles can be extended to broader communities, such as patients sharing medical data, who expect not only privacy and data protection, but also a say in how their data are used and how they might benefit from research outcomes.

This lack of transparency, sovereignty, and benefit underlines the need for more user-centric approaches in data computation, where individuals have the ability to influence and understand how their data are used, what the results will be, how they will be applied, and how they benefit from sharing to make informed decisions about contributing their data to third parties.

A possible way to solve this problem is the *Solid* (Social Linked Data) Project[1]. First, instead of storing data in a data silo, the data are stored decentralized in the so-called *Pods* (Personal Online Data Stores). Pods are personal storage on the Web where all its data is owned and controlled by the individual, be it a person, an organization, or an application. Pods can be hosted anywhere and by anyone; there is no limit on how many Pods one owns and where they are located.

Secondly, *Solid* also has built-in mechanisms for authentication via WebID[2] and authorization through Web Access Control (WAC). As a pod owner, one can give and revoke precise access to any slice of data as needed and decide which persons, applications, or organizations can access what part of data at any time. Privacy and consent are honored by design, thus increasing trust and fostering responsible data practices.

This thesis addresses the problem of limited control and transparency for the data owner on traditional data platforms, particularly how personal data is utilized for analytical processing. The core issue revolves around the existing systems' inability to provide users with sufficient insight into, and control over the specific use of their data, as well as a clear understanding of the derived analytical results and their use. This lack of transparency and control calls for a more user-centric approach in data management, where individuals have a definitive say in how their data are used and can actively decide to participate or opt out of sharing their data. The purpose of the thesis is to explore how the integration of Solid with these data platforms can enhance privacy, transparency, and data sovereignty, addressing this critical gap in current data handling practices.

1.2 Research Questions and Objectives

To address the problem of traditional data platforms failing to offer users insight and control over how their data is used and the outcomes of its analysis, the following primary research question has been defined.

Main Research Question: *What is a proper solution to combine Solid with privacy-preserving analysis tools in a way that avoids centralized data storage while still meeting the functional requirements of data platforms?*

> This research question aims to design a novel data platform that leverages user-contributed data through Solid Pods, moving away from traditional centralized data repositories. To achieve this, we will assess state-of-the-art data platforms to conceptualize the development of a framework that meets the comprehensive functional requirements of a data platform.

As this research is expected to produce a conceptual framework for a data platform that uses *Solid* for data sharing, subsequent research questions will explore the technical

[1] `https://solidproject.org`

[2] A WebID is a URI/IRI that uniquely identifies a person, company, organization, or other agent.

aspects of integrating *Solid* with privacy-preserving data platforms, focusing on important qualities such as performance, information exposure, and privacy protection.

Research Question 1: *In what way can Solid be integrated into privacy-preserving computation platforms to minimize the performance impact?*

Fetching data from *Solid Pods* instead of local databases certainly brings performance trade-offs. To establish a baseline for retrieval performance, we evaluate a standard approach, similar to the one used in *TIDAL* [SOSD23], where all available data are fetched to be filtered locally. We then compare it with a granular retrieval strategy that fetches only the necessary data, which is prepared by a pre-processing step, which we implement via a user-facing *Solid* application. We analyze how the two strategies impact the fetch and process performance of our system.

We start by analyzing the storage aspects of a traditional data platform to find a solution to replace data retrieval with *Solid*. The existing infrastructure will then be extended to allow user-centric and privacy-preserving data retrieval. As part of this step, we develop a process for analysts to query data in a request-based system from data owners participating in this platform. This mechanism should be designed to provide researchers with straightforward access to the data they need, creating an environment that promotes scientific discovery while also respecting the privacy interests of data owners.

Then, we establish a baseline by analyzing data retrieval using the approach used by *TIDAL* [SOSD23] and assess how a more granular strategy influences performance. This evaluation focuses on measuring the retrieval time, the processing time, and the number of HTTP requests required. By conducting these tests with different numbers of *Data Contributors* and datasets, we gain insight into how our proposed strategy behaves in terms of scaling.

As a further step, we compare our *Solid*-based strategies to the performance of *WellFort* [EEM+21], which uses a centralized repository. By contrasting our decentralized approach against the local repository, we evaluate whether *Solid* can be a viable and scalable alternative.

In a final step, we will evaluate how parallelization will optimize the more efficient data fetching approach even further. Therefore, we implement a mechanism in our proposed system to allow multiple workers to obtain and process data from *Solid Pods* simultaneously. This evaluation will demonstrate the extent of optimization that can be achieved through parallelization.

By establishing this performance baseline, we provide a foundation for optimizing data retrieval in platforms with *Solid*-based data retrieval. The dataset used during our evaluation will be made available, enabling further research to refine our benchmarks and extend our proposed retrieval strategies.

Research Question 2: *To what extent can we reduce the information exposure from individuals contributing to the privacy-preserving computation platforms by using Solid?*

While *Solid* offers data owners benefits such as data control and transparency, the decentralized nature of *Solid Pods* requires data transit through uncontrolled environments, such as the public Internet, thereby posing potential risks to data security. An important aspect to investigate when using *Solid* as personal data store, is data exposure. This is due to the fact that data in *Solid* are accessed on a file basis, meaning that even if only a subset of the data of a file is needed, the whole file will be transferred.

In order for the data platform to fetch the data from the user pods, the reading permissions on the files containing the information have to be set. The goal is to minimize both the data that must be authorized and the data that must be transferred for a data request so that no unnecessary data is leaked. We show, through the use of our strategy, how only the minimum data points necessary for a data request need to be authorized and transmitted.

We propose an evaluation against state-of-the-art data platforms in which we demonstrate that our novel implementation effectively reduces data exposure. Therefore, we create datasets within the *Solid Pod*, that we transfer to the data platform. Subsequently, we will evaluate and compare the levels of data exposure of our proposed granular strategy compared to the currently used approaches. We measure the extra data authorized and sent when responding to data requests using the number of additional triples as unwanted data leakage.

Based on the results of the experiment, our strategy to minimize data transfer between *Solid Pods* and the data platform will be recommended.

1.3 Aim of the Work

The objective of this thesis is to explore the feasibility of integrating *Solid*, as a decentralized storage alternative, into data platforms to enhance data sovereignty and support privacy-preserving computation. Personal data is often stored and managed by third parties, such as hospitals, research institutions, or service providers. Data owners may give their consent for data processing, but often lose control over how their data are accessed, processed, and used. This thesis investigates whether a solid-based approach can offer an alternative that allows data owners to actively manage access to their data, ensuring greater data sovereignty.

More specifically, this thesis aims to develop and evaluate a framework that enables structured, request-specific data access with *Solid Pods*. Instead of broad, pre-approved data sharing, the proposed approach ensures that only minimal necessary data subsets are retrieved for computational tasks.

This work does not propose a replacement for existing data platforms, but rather an extension that introduces decentralized user-controlled data access through *Solid*. The feasibility of this approach is evaluated through an experimental evaluation that focuses on functional requirements, as well as the performance overhead and data exposure. The result determines whether *Solid* can be practically integrated into privacy-preserving computing frameworks.

Contributions

As part of this thesis, the following components are expected to be delivered:

- A design of a request-specific and consent-driven data sharing model based on *Solid*.
- A reference implementation of a mediator that manages controlled access to data hosted in *Solid Pods* using request-specific WebIDs.
- An evaluation of performance trade-offs introduced by *Solid*, comparing different retrieval strategies to analyze fetch times, computational efficiency, and information exposure.
- Guidelines for integrating data retrieval via *Solid* into existing data platforms.

1.4 Methodological Approach

In our research, we follow the methodological framework of Design Science by Peffers et al. [PTRC07], which provides six steps that lead from the initial identification of the problem to the communication of the results.

- **Problem Identification and Motivation**
  We conduct a detailed and thorough review of the literature following [Fin13] to discover the existing implementations and limitations, where our primary sources are peer-reviewed articles and case studies.
- **Objectives of a Solution**
  We define the functional requirements for our solution based on the literature research. Our aim is to create a data platform that uses user-contributed data through *Solid* rather than requiring a centralized data store. We want to propose an alternative to current state-of-the-art data platform implementations by providing a more user-centric and privacy-preserving mechanism for data sharing. Therefore, we further want to optimize our implementation by focusing also on non-functional qualities such as performance, information exposure, and privacy protection.
- **Design and Development**
  In this phase of our thesis, we will employ an agile development methodology to craft our artifacts. This approach will allow us to initially focus on delivering

a functional prototype, ensuring that all core capabilities are addressed. Once functionality is assured, we will enter a phase of iterative optimization, where our attention will shift to enhancing non-functional aspects of the artifact.

- **Demonstration**
  For the demonstration, we will bring our artifact into the context of a state-of-the-art data platform such as [EEM+21, SOSD23]. By simulating common scenarios, we will validate that our platform aligns with functional requirements under practical conditions. This demonstration will serve as a verification of the system's ability to handle privacy-preserving computation using data retrieved from *Solid Pods*, ensuring that our solution is robust and ready for evaluation.

- **Evaluation**
  Through testing, we will evaluate our proposed solution against the functional requirements extracted from existing solutions. Efficiency will be measured by the time needed to collect and process data, focusing solely on the temporal performance of data collection from *Solid Pods*. Effectiveness will be evaluated by the degree to which our implementation minimizes data exposure compared to the current approach in data platforms such as [EEM+21, SOSD23], measuring unwanted data leakage through the metric of the absolute number of unnecessary data points authorized and transferred.

- **Communication**
  Our work aims to lay a clear and accessible path for others to follow, providing guidelines to our integration process of incorporating *Solid* into a data platform. By sharing our experience and findings, we contribute to the broader conversation on privacy and user-centric data control, helping other researchers and platform providers apply these methods in their infrastructures.

1.5 Structure of the Thesis

This thesis explores how *Solid* can be integrated into existing data platforms to improve data sovereignty.

Chapter 2 reviews related work, including existing privacy-preserving data platforms, federated learning, distributed analysis, data sovereignty and the *Solid Project*. This establishes the foundation for understanding the challenges and opportunities in the current landscape.

In Chapter 3, the design and development of the solution are described. This includes the definition of stakeholders, relevant use cases, functional and non-functional requirements that the system must fulfill, and the rationale behind these decisions.

The conceptual architecture of the proposed solution is detailed in Chapter 4. This includes the key components, workflows, and data management strategies necessary to support decentralized user-centric data sharing based on *Solid Pods*, while ensuring privacy and compliance through controlled data access.

Chapter 5 presents the evaluation of the system, including its implementation, functional verification, performance analysis, and its effectiveness in minimizing data exposure. Furthermore, it proposes a set of guidelines for integrating *Solid Pods* into existing data platforms in Section 5.5 to support future implementations.

Finally, Chapter 6 provides the conclusion, revisiting the research questions, summarizing the results, discussing the limitations of our current work, and outlines future research directions.

CHAPTER 2

Related Work

2.1 Data Sovereignty

Data sovereignty traditionally refers to the right of individuals, communities, or nations to exercise full authority and control over their own data, analogous to political sovereignty over land and governance. In the context of digital data, this means that data owners have the power to decide how their data are collected, used, and shared [CGFR$^+$20]. This concept of data sovereignty is a key factor that motivates the design of our system.

The CARE principles provide a framework that focuses on the aspects of human rights and ethical considerations. Caroll et al. [CGFR$^+$20] stress a people- and purpose-oriented approach in data practices. They advocate that these principles not only promote community wellbeing but also underscore the need to empower data subjects with control over their information. In our system, this empowerment is achieved through the self-hosting of personal data, accessible consent mechanisms, and granular access controls to third parties.

Moreover, as Braud et al. [BFRLG21] suggest, data sovereignty is the key to building trust in data-driven ecosystems by breaking down traditional information silos and enabling cooperative data services. Our implementation follows this vision by leveraging the decentralized data storage solution *Solid* as an alternative data source to central data repositories. In doing so, we aim to minimize unnecessary exposure of personal data and address the challenges of excessive data collection highlighted in [Dat19].

The European Union uses the term digital sovereignty to refer to greater control over digital infrastructures, technologies, and data flows within and across its borders [Kom21]. By imposing legislative frameworks such as the *General Data Protection Regulation* (GDPR), the EU embeds the ideals of data sovereignty into its pursuit of digital sovereignty, ensuring that data concerning European persons are protected by European laws. While data sovereignty focuses on the freedom to control data, digital sovereignty extends this concept to cover entire digital ecosystems. This broader scope is relevant to our work because we aim to enable interoperable data exchanges across diverse digital infrastructures that may span borders. Our design aims to follow the EU's vision of digital sovereignty by providing a framework that respects individual control.

2.2 Data Platforms

Data platforms serve as foundational infrastructures for managing and extracting value from data-driven ecosystems. They incorporate a combination of technical, organizational, and business components that enable the collection, storage, and processing of data for various stakeholders. With the rapid expansion of the digital economy, data have become a strategic asset, often described as the "new oil," highlighting its central role in innovation, commerce, and governance [LS22].

However, the governance models of the data platforms vary significantly between regions. In North America, for example, dominant digital platforms tend to prioritize the interests of platform providers, often leveraging network effects to consolidate data control within a few major corporations. This centralized model has raised concerns about monopolization, lack of transparency, and limited user control over personal and organizational data [JSYH22]. The regulatory environment in the United States has traditionally favored a market-driven approach, allowing companies to establish proprietary data ecosystems with minimal restrictions on data access, portability, or interoperability.

In contrast, Europe has taken a structured, regulation-driven approach to data governance, emphasizing data sovereignty and fair data access as fundamental principles. As stated by the European Commission President Ursula von der Leyen in her political guidelines for the 2019–2024 Commission, Europe must "balance the flow and use of data while preserving high privacy, security, safety, and ethical standards" [TL22]. The European *Strategy for Data*, introduced by the European Commission, envisions a framework in which data flows freely but remains subject to European values and legal protections, including GDPR and the *Data Governance Act* [NL22].

In Article 5[1] of the GDPR, the term *data minimisation* is introduced and mandates that "personal data shall be adequate, relevant and limited to what is necessary in relation to the purposes for which they are processed". This applies for one, to the quantity of personal data collected, but also involves decreasing the 'personality' of the collected data [Eur20] using measures such as pseudoanymization to reduce the ease with which data can be linked to specific individuals.

By shifting storage towards *Solid*, our system is designed to overcome the limitations of centralized data platforms. We improve transparency and user control by ensuring that data are stored in a way that inherently aligns with and respects measures, promoted in the European regulatory frameworks, such as *data minimization* and sovereignty.

[1]`https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434`

2.3 Reference Implementations

In this section, we present reference implementations that illustrate contrasting architectural approaches to privacy-preserving data platforms, selected for their relevance in combining secure data analysis with user-centric control and data sovereignty. We will evaluate their shortcomings and strengths in regard to privacy-preservation and data sovereignty to propose a novel approach on data retrieval from *Solid Pods*.

2.3.1 WellFort

The *WellFort* architecture proposed by Ekaputra et al. [EEM$^+$21] combines privacy preservation mechanisms with auditability features to create a data platform that addresses the challenges of managing and analyzing personal and sensitive data. The architecture consists of three main parts: the *Secure Repository*, the *Trusted Analysis Environment*, and the *Audit Box*.

The *Secure Repository* manages user data, storing sensitive information with explicit user consent and metadata. The raw data of a data donor are tagged with a persistent identifier (PID) using the Invenio framework [2]. High-level descriptions of the data and user consent are extracted and stored in a Triplestore, thus decoupling metadata from the raw data, which is important to find valid candidates for a data request without actually loading any of the sensitive data. The *Usage Policy Compliance Check Engine* ensures that only data with the right user consent are accessed using semantic reasoning. [EEM$^+$21]. Although this approach takes measures to load personal data only when necessary into other parts of the infrastructure such as the *Trusted Analysis Environment*, the centralized data storage model still conflicts with the principles of *data minimization*.

The *Trusted Analysis Environment* isolates data analysis tasks from raw data access. The *Analysis Server* is implemented with Opal[3], a data management system that supports secure import, transformation and export of various types and formats of data. The *Analysis Database*, a PostgreSQL database, temporarily stores session-specific data. The *Analysis Interface* uses RStudio Server to perform analysis with DataSHIELD[4], a library that wraps common statistical methods with safeguards that prevent analysis tasks from revealing sensitive information. We will follow the approach of the *Trusted Analysis Environment* and also use DataSHIELD as a privacy-preserving computation layer in our proposed solution.

The *Audit Box* is responsible for collecting, storing, and providing provenance data. Trails are collected throughout the architecture at all stages. Audit trails can be retrieved using self-written or pre-defined SPARQL queries in the GraphDB user interface. We will also integrate thorough provenance tracking within our solution. Each access attempt to an external resource inside a *Solid Pod* , even if unsuccessful, will be recorded, together with

[2]https://inveniosoftware.org
[3]https://www.obiba.org/
[4]https://datashield.org/

a timestamp, HTTP return code, and hash of the result. In this way, we can provide a clear provenance trail when it comes to a web-based decentralized storage solution like **Solid**.

The feasibility and effectiveness of the *WellFort* approach have been demonstrated through a prototype implementation and evaluation within the medical domain, highlighting its potential to facilitate secure and compliant data analysis in sensitive fields.

The *WellFort* architecture, while robust in its mechanisms for privacy preservation and auditability, is based on a centralized data storage model that conflicts with the principles of *data minimization*. The current architecture requires initial broad consent to store personal data. Once data is stored, any change or withdrawal of consent requires the data donor to ask the platform for redaction. We have identified two shortcomings with this approach. Firstly, broad consent may not be aligned with the principle of data minimization mandated by *GDPR* Article 5, as it enables the collection and use of more personal data than is strictly necessary.

Secondly, users lose direct control and oversight over their data when they submit them to *WellFort*. From the user's perspective, the system functions as a black box, encapsulating data handling and analysis tasks proceeding without further user input or updates. Users are no longer included in permission requests or informed about ongoing uses, which centralizes control within the system instead of empowering the data donor.

To address these shortcomings, we propose the shift towards a decentralized storage solution. In this model, personal data would remain under the control of the data owner, ideally stored locally on personal devices or on a personal cloud storage service. Data would only be transmitted to research environments when absolutely necessary and only for specific consented purposes, minimizing the amount of data stored centrally.

2.3.2 TIDAL

This strategy is explored by Sun et al. in their proposed ciTIzen-centric DatA pLatform platform *TIDAL* [SOSD23]. The platform is designed to give individuals control over their personal data while allowing secure and privacy-preserving data analysis for health research. They focus on the data sovereignty of data owners and, therefore, do not store any sensitive data in their architecture; instead, they leverage personal data stores.

These data stores are based on *Solid* and allow users to store their personal data in decentralized storage pods where they have fine-grained control over who can access their data and for what purpose.

In *TIDAL*, researchers create participation requests that are stored as digital consents in their own pods. These requests specify the type of data needed, the purpose of the research, and the analysis methods to be used. Participants can view these requests and decide to give access to their data directly through the *TIDAL* platform anytime.

The analysis itself is executed in an open source temporary sandbox container that pulls data from the participating data pods. This sandbox is a docker container that analyzes the data of the participants with a predefined analysis script and only passes the result on to the researchers pod. The container is destroyed after the process. In the participation request, the container used is linked and accessible for inspection and ensures that the proposed method is transparent for the data donors.

Although the use of short-lived containers ensures that data is processed strictly according to the specifications in the data request, it also imposes significant limitations on flexibility. In this model, every analysis step must be predetermined and submitted in advance, leaving little room for interactive data exploration. As a result, data consumers face challenges in dynamically refining or adapting their analytical approach, since there is no mechanism to directly engage with the data during the analysis process. In contrast, our system provides researchers with an interface to interact with the data directly, enabling iterative and exploratory analyzes that can be refined in real time while still maintaining robust privacy safeguards.

[SOSD23] also integrates ontologies such as SNOMED CT[5] and DPV[6] to ensure that personal health data are structured and interoperable. In our proposed solution, DPV will also be used to create data requests and consent, following the approach of Sun et al.

TIDAL also does not comply with *data minimization*, as in their proposed algorithm, they transfer whole data files to the temporary analysis environment. This is due to the fact that with *Solid Pods*, data are always accessed on a file basis. The platform, upon recognizing the required data for a data request within the data file of the pod, authorizes the complete data file to be transferred. Although the data are then filtered for only the requested data attributes inside the analysis containers, we see a clear shortcoming in this regard, as more data than necessary are authorized and transferred. Our proposed solution will build on the concept followed in TIDAL and optimize the data extraction process in relation to *data minimization*.

2.3.3 Personal Health Train

A significant portion of valuable health data remains trapped within the organizational limits of hospitals and clinics, largely due to ethical considerations. The Personal Health Train (PHT) architecture was proposed by Brian et al. in [BCvS+20] and is designed to enable secure and privacy-preserving data analysis while ensuring that data remain within their original location while analysis algorithms, represented as "*trains*," travel to the data instead. This federated learning architecture is used in many different contexts, such as integrating the architecture in the cloud as in [MPS+21] [SPM+22], optimizing the deployment through containerization [HGP+22], prediction modeling with radiomics [SZT+19], or using the infrastructure to run high-volume studies such as [DDO+20]. In this architecture, data are stored in independent "*data stations*", each manages access, and is responsible for making data accessible while ensuring compliance with relevant regulations. "*Trains*" are essentially containers that carry analysis algorithms, metadata

[5]https://bioportal.bioontology.org/ontologies/SNOMEDCT
[6]https://w3c.github.io/dpv/2.1/dpv/

specifying the requester, the requested data, and a description of the analysis to be performed. Trains travel to data stations where they perform their tasks locally. Data controllers can manage their data through the "*Personal Gateway*", a mediator between data stations and controllers used to manage access permissions and monitor data usage across data stations. Data controllers are typically organizations such as hospitals, research institutions, or healthcare providers that oversee large collections of personal health data.

Therefore, data owners, typically patients, lack direct oversight, control, or active participation in decisions about the use of their data, affecting their autonomy over personal health information. This centralized control contrasts with the *data minimization* principle, as it involves accumulating extensive datasets in institutional data stations, regardless of immediate research needs.

In contrast, our work explores an alternative decentralized approach using *Solid Pods*. *Solid* is proposed primarily as a storage solution that empowers data owners by allowing them to control access to their data in a distributed manner. Unlike *data stations* in the PHT architecture, *Solid Pods* do not support in-pod computation [SMH+16]. This means, that while *Solid* enhances data sovereignty and minimizes unnecessary data accumulation, computational tasks are not part of the current specification[7].

2.3.4 DataSHIELD

DataSHIELD[8] focuses on secure federated analytics, sharing only aggregated results. Its architecture operates within a client-server model, where sensitive individual-level data remain securely on the data custodian's servers, as shown in Figure 2.1. These data nodes, represented as "Server 1," "Server 2," and "Server 3," are hosted by institutions or organizations responsible for the storage and protection of sensitive information. The analyst interacts with the system via the DataSHIELD client, often using the R programming environment, to issue standardized analysis commands. These commands are sent to the DataSHIELD Client, a centralized middleware, which forwards commands to each data server, ensuring computations are executed locally without raw data leaving the servers. This process, as visualized in Figure 2.1, is designed to produce only aggregated and non-disclosive output, such as low-dimensional summary statistics, which are returned to the analyst. With these safeguards in place, compliance with privacy standards is enforced, and the risk of data leakage or re-identification is minimized.

[7]https://solidproject.org/TR/protocol
[8]https://datashield.org/

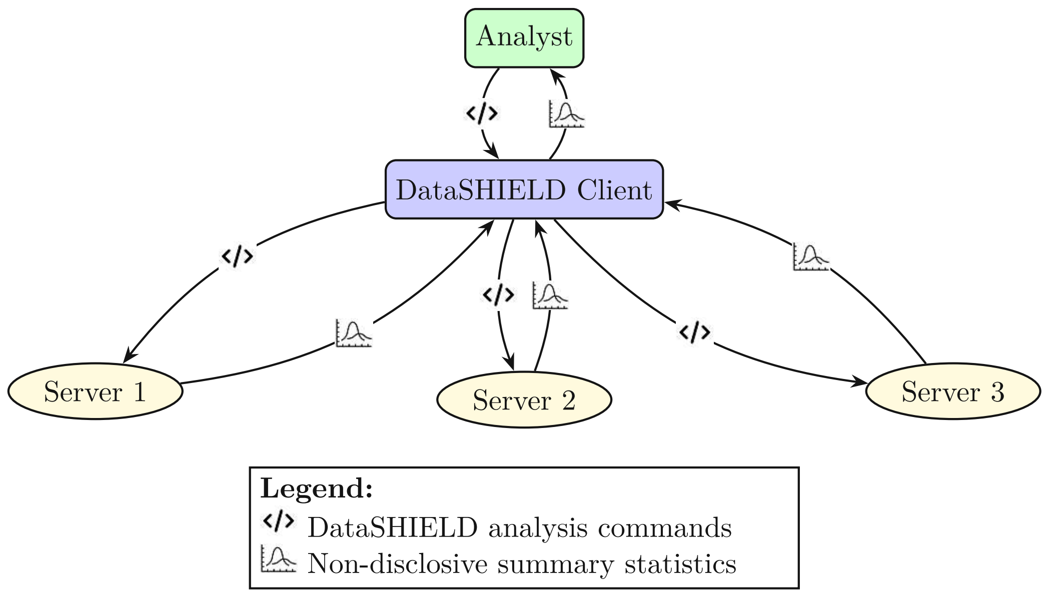
Analyst
DataSHIELD Client
Server 1
Server 2
Server 3
Legend:
DataSHIELD analysis commands
Non-disclosive summary statistics

Figure 2.1: DataSHIELD Architecture

DataSHIELD is widely adopted across diverse research domains and settings. Its versatility is demonstrated by its application in federated analyzes ranging from multicohort studies in pediatric obesity [DCC+23] to neurooncological investigations employing DNA methylation analyses [SSN+24] and biomarker comparisons in COPD research [ZHM+24]. The wide use of DataSHIELD underscores the platform's capacity to manage various types of data while enforcing privacy standards through their proposed infrastructure.

Moreover, the DataSHIELD framework is inherently extensible and can be adapted through the development of additional coding packages or add-ons. Tools such as those described by [EMMA+23] facilitate user-friendly access for non-technical users, while the dsSynthetic package [ST22] enables synthetic data generation for script testing and debugging in federated environments. We choose DataSHIELD, as these developments show the flexibility, a data platform provider has, when leveraging DataSHIELD as a privacy layer.

However, integrating DataSHIELD directly with individual *Solid Pods* as independent data sources would lead to a complex and inefficient infrastructure due to the decentralized nature of *Solid* and the high overhead associated with managing multiple remote data sources. Instead, our approach uses DataSHIELD as a privacy-preserving interface for analysts, just as [EEM+21].

In our proposed system, data from *Solid Pods* are temporarily downloaded and aggregated into a central temporary data store. This store contains only the minimal request-specific attributes and personal data needed for analysis, adhering strictly to *data minimization*. DataSHIELD is then used as the analytical interface to provide secure, privacy-preserving access to this curated dataset. By operating in a temporary central repository, our system avoids the latency and complexity of interfacing with numerous individual *Solid Pods* while still maintaining robust privacy features. In this way, our system bridges the gap between user-controlled data and high-quality, secure data analysis.

2.4 PDS

Building on the idea of data sovereignty, *Personal Data Stores PDSs* provide users with control over their own data, ensuring that third parties can only access or process these data with explicit consent or action. More fundamentally, it represents a change from traditional centralized data processing models in which organizations collect, store, and use data within their own infrastructures, often limiting individuals' ability to understand or influence the handling of their data. The goal of *PDSs* is to decentralize both data storage and processing to increase data sovereignty [JCS20].

PDSs typically incorporate a variety of technical features that together empower people with control over their data. First, they enable local data capture and secure storage directly on the user's device, unifying information from multiple sources, such as smartphones, wearables, and online platforms. Second, they support on-device computation, allowing analytical processes to be executed locally so that raw data do not need to leave the secure environment. Third, *PDSs* include mediated data transfer mechanisms, by which any sharing of data or computed results with external parties is strictly controlled and only occurs under user-specified constraints. Finally, they offer robust transparency and control measures, including real-time monitoring tools, configurable access controls, and data encryption, to ensure that users remain informed and can effectively govern how their data are accessed and processed [JCS20].

Recent research by Fallatah et al. [FBP23] provides a comprehensive overview of the *PDS* landscape, which has emerged as an alternative to centralized data architectures. Within this landscape, a variety of platforms have been developed with different design philosophies and targeted use cases. For example, *MyDex*[9] [PSTW15], is a cloud-based *PDS* platform that aggregates personal data using their commercially developed platform with its own proprietary MRD[10] API. In contrast, *Personal Data Vaults* (PDVs) are implemented on phones and therefore focus mostly on sensitive information, such as location, images, and health data, directly on the user's phone.[FBP23].

Other PDS implementations, such as *Databox* [CLC+18] take a more localized or edge-based approach and propose a designed for domestic environments, where it collects and processes data from various Internet of Things (IoT) devices locally. Meanwhile, the *Hub of All Things*[11] (HAT) offers decentralized micro-servers that function as personal data hubs, allowing individuals to store and manage their data and engage in data exchange, but only within the realm of proprietary software provided by Dataswyft[12] [FE23].

[9]https://mydex.org/
[10]https://dev.mydex.org/mrd-api/mrd-overview.html
[11]https://www.hubofallthings.com/
[12]https://www.dataswyft.io/

In general, while the diverse range of *PDS* platforms, spanning cloud-based solutions, mobile vaults, and edge computing devices, demonstrate potential to improve data sovereignty and user control, they either are tailored to specific contexts or employ proprietary software [FE23]. These challenges motivate the exploration of a new alternative architecture: The Solid Project, an open protocol that builds on open web standards to offer a robust, decentralized, and highly interoperable solution for personal data management.

2.5 The Solid Project

The Solid Project[13] founded by Sir Tim Berners-Lee, the inventor of the World Wide Web, introduces the concept of *Personal Online Data Stores* (Pods). *Pods* are essentially private data containers on the Web, where all its data are owned and fully controlled by the individual, be it a person, an organization, or an application. This concept inherently supports the principles of data sovereignty and privacy, empowering users with control over their personal data [SMH+16]. For our work, *Solid* provides the foundational storage layer, ensuring that data remain under the control of its owner and aligning with our goal of user-centric data management.

Solid introduces the Solid Protocol[14] specification which describes how *Solid* components such as servers and clients can be interoperable using Web communication protocols, global identifiers, authentication and authorization mechanisms, data formats and shapes, and query interfaces. This specification is a significant advantage over other PDS solutions as it provides a thought-out and structured framework that can be universally adopted and implemented, much like other established Internet protocols.

2.5.1 Authentication

Instead of using generic identifiers like "Bob123", *Solid* assigns each user a WebID. A WebID is an HTTP-based identifier (for example, `https://verysolid.de/Bob123-pod/profile/card#me`) that is used to uniquely represent an entity, such as a person, organization, or software and that can be de-referenced to retrieve an RDF profile document containing the user's information. In order to access a *Solid Pod* or a resource inside a pod, users and agents must first authenticate.

Solid therefore uses *Solid OpenID Connect*[15] (Solid-OIDC) as the primary authentication mechanism, which relies on widely adopted standards from *OAuth 2.0*[16] and *OpenID Connect*[17] to authenticate users through trusted Identity Providers (IdPs).

[13] `https://solidproject.org`
[14] `https://solidproject.org/TR/protocol`
[15] `https://solid.github.io/solid-oidc/`
[16] `https://datatracker.ietf.org/doc/html/rfc6749`
[17] `https://openid.net/developers/how-connect-works/`

```
@prefix acl: <http://www.w3.org/ns/auth/acl#>.
<#authorization1>
    a acl:Authorization;
    acl:accessTo <https://verysolid.de/Bob123-pod/diagnoses>;
    acl:mode acl:Read, acl:Write, acl:Control;
    acl:agent <https://verysolid.de/Bob123-pod/profile/card#me>.

<#authorization2>
    a acl:Authorization;
    acl:accessTo <https://verysolid.de/Bob123-pod/diagnoses>;
    acl:mode acl:Read;
    acl:agent <https://health.de/doctor/profile/card#me>.
```

Listing 2.1: Access Control List Example diagnoses.acl

2.5.2 Web Access Control

Web Access Control (WAC) in *Solid* is a standardized mechanism for managing access to resources stored in *Solid Pods*, as defined by the Solid WAC specification[18]. WAC uses Access Control List (ACL) files associated with each resource to specify which users, identified by their WebIDs, are granted specific permissions such as read, write, append, or control. For example, consider a scenario where Bob stores a document named "diagnoses" in her *Solid Pod*. To allow the Doctor to view this document, Bob extends the ACL file named "diagnoses.acl" by a new `acl:Authorization` object as seen in Listing 2.1. The resource is denoted by `acl:accessTo`, in this case the diagnoses file. The mode of access is denoted by `acl:mode` which can be a combination of `acl:Read`, `acl:Write` and `acl:Control`. Finally, the `acl:agent` specifies the WebID of the Doctor (`https://health.de/doctor/profile/card#me`).

This explicit rule ensures that, besides Bob, only the Doctor can access "diagnoses", while all other users are denied access. By enforcing such granular, rule-based permissions, WAC maintains the security and privacy of user data. This allows for scenarios as visualized in Figure 2.2. Bob, the owner of the *Solid Pod* has full access and control over his data. He saves his diagnoses, prescriptions, and blood tests in his pod. In addition, he has a heart monitor which also actively writes the measurements in the pod. This can be achieved by allowing the WebID of the heart monitor to write to a file or a whole folder inside Bob's pod. The write access for the heart monitor and the read access for the doctor are actively managed by Bob himself through the creation of ACL files inside his pod.

We will make use of this mechanism to manage access to specific data points in *Solid Pods* similar to [SOSD23], but propose a solution on *data minimization*. We will automatically create WebIDs, that represent data requests and build a *Solid* application that manages data extraction and ACL generation to streamline a request-based data sharing model that can be integrated into existing data platforms.

[18] `https://solidproject.org/TR/wac`

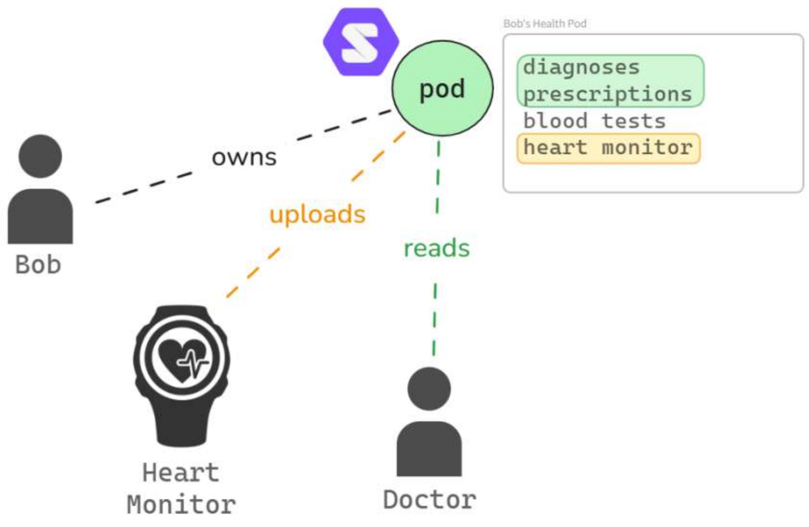

Figure 2.2: Interaction of different Entities with a Solid Pod

2.5.3 Implementations

Hochstenbach et al. [Hoc23] suggests that *Solid Pods* could serve as personal digital libraries, where researchers maintain control over their scholarly outputs while ensuring accessibility and interoperability. Similarly, *Solid* has been explored to manage health data, allowing individuals to store and share fitness and medical data securely while granting fine-grained access to applications [Pau20a]. [Pau20b] provides a decentralized approach for recording and managing personal health and fitness activities using Google Fit APIs to store health data in *Solid Pods* according to the HL7 FHIR RDF specification[19].

Extensions of *Solid*, such as Libertas, aim to enable privacy-preserving computation using Secure Multi-Party Computation (MPC) and Differential Privacy (DP) [ZGA+23]. It uses encryption agents that retrieve the data from *Solid Pods*, encrypt it, and distribute secret shares to computation agents, which execute the MPC computation collaboratively. Although Libertas demonstrates their infrastructure protect data during computation, our approach does not deploy an MPC infrastructure. Instead, we rely on DataSHIELD to provide privacy-preserving analysis capabilities.

A recent contribution to the Solid ecosystem is the ESPRESSO framework, which addresses the challenge of enabling efficient and privacy-preserving search across distributed *Solid Pods*. ESPRESSO introduces a decentralized architecture with components such as Brewmaster (for local pod indexing), CoffeeFilter (for searching within pods) and Barista (as the user interface), all connected through a federated overlay network [RSM+23]. However, ESPRESSO introduces additional complexity, storage, and synchronization overhead due to its distributed indexing approach and may even expose sensitive meta-

[19]https://www.hl7.org/fhir/rdf.html

data, such as keyword occurrences, to search participants. These challenges further justify our design choice to use *Solid*'s secure storage with well-tested privacy-preserving analytical tools such as DataSHIELD.

Despite the aim of providing user-controlled data pods, *Solid* has been criticized for its lack of the cryptographic foundations necessary to ensure robust security. Instead, the model places trust in the administrators of the servers storing the pods, which poses the risks of potential surveillance and data breaches. The dependence on policy-based security solutions, rather than cryptographic protections, casts doubt on *Solid*'s privacy guarantees, particularly when third parties maintain control over the infrastructure. The system's reliance on URL-based identification and access control, which depends on the centralized DNS system, is contrary to the claims of decentralization. [Hal22]

2.6 Summary

The idea of data platforms is not new, but is receiving substantial tailwinds. The *European Strategy for Data* [TL22] establishes a robust framework to drive innovation, transparency, and trust in data management throughout Europe. The Data Governance Act[20], for example, is designed to promote data sharing while protecting the rights of data owners, and the broader push toward digital sovereignty [Kom21], and ensures that personal data are managed under strict privacy and security standards. We will focus to improve on *data minimization* as part of our proposed solution.

In our exploration of contemporary data platforms, two concepts, namely *TIDAL* [SOSD23] and *WellFort* [EAK+19], stood out due to simplicity and focus on data sovereignty. *TIDAL* distinguishes itself by demonstrating a proof-of-concept with a strong focus on data sovereignty and control over personal data through decentralized data hosting using *Solid*, granular consent mechanisms and temporary privacy-preserving data analysis environments using Docker.

WellFort, on the other hand, offers a more traditional data platform infrastructure with a centralized data store, serving as a robust foundation from which to advance and develop toward decentralized systems such as *Solid*. *WellFort* excels in areas important for effective data management, including regulated access, comprehensive governance, and detailed traceability implemented through semantic web technologies. These features are complemented by privacy-preserving mechanisms using DataSHIELD that enable analysts to interact with data confidentially. Together, these capabilities make *WellFort* an ideal model to demonstrate how traditional data platform frameworks can evolve to incorporate *Solid* as an alternative to centralized storage and enhance data sovereignty for data donors.

We aim to combine the strengths of *WellFort* and *TIDAL* to create a new approach for data platforms that combines a reliable and secure foundation with strong privacy controls with request-based data sharing and decentralized data management. By merging

[20]https://digital-strategy.ec.europa.eu/en/policies/data-governance-act

these features, we propose a blueprint for modernizing existing data systems to be more user-driven and respect data sovereignty using *Solid*.

The PHT, with its federated learning approach, offers a promising approach for privacy-preserving computations, but its integration within the *Solid* ecosystem would mean a fundamental change of the design principles of *Solid*. It is envisioned as a decentralized storage platform, with well-defined protocols and standards for data retrieval and access control, rather than a computational framework. Introducing a federated learning module would require extending these protocols to support distributed computations, a paradigm shift that *Solid* is not currently developed for. This additional layer of complexity would involve the introduction of new interoperability standards and computational models that go beyond the intended scope of *Solid*, potentially undermining the simplicity and focus of the system. Given these challenges and considering that *Solid's* robust access control mechanisms already provide a transparent and controlled means of data sharing, our approach focuses on using *Solid* as a storage platform.

CHAPTER 3

Design and Development of the Solution

3.1 Stakeholders

Miksa et al. [EEM+21] identify three primary stakeholders in a data platform context: *User*, *Analyst*, and Auditor. The *User* is the individual who provides their personal data and consents to its analysis. The *Analyst* is the researcher or data scientist who accesses and processes these data for specific research or analytical purposes. The *Auditor* is an external entity responsible for reviewing and verifying the processes and results of data usage, ensuring compliance with ethical and legal standards.

In contrast, the TIDAL platform, as described by Sun et al. [SOSD23], identifies stakeholders such as *Participants*, who are data subjects maintaining control over their personal data in *Solid Pods*, *Researchers*, who request access to these data pods for analysis, and a *Trusted Party*, which then performs the data retrieval and analysis. Similarly to the *Trusted Party* in the TIDAL platform, we propose the *Solid Gateway*, a mediator application to help data platforms transition towards more user-centric data sources.

Building on these frameworks, we introduce six distinct stakeholder roles, as seen in Figure 3.1, that can be categorized into two groups: users and providers. *Data Contributors*, *Data Consumers*, and *Auditors* are users, actively engaging with the infrastructure to manage, share, analyze, or audit data. Meanwhile, *Platform Providers*, *Pod Providers*, and *Identity Providers* deliver the infrastructure and services required for these activities.

3.1.1 Data Contributor

Data Contributors are individuals who own and manage their personal data and have an intention to share subsets of these data under certain conditions. They may want to

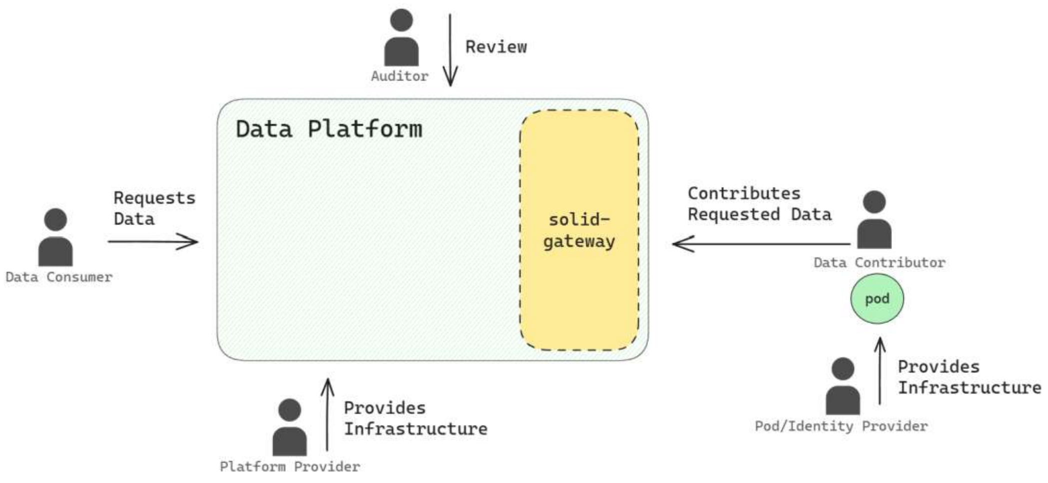

Figure 3.1: Stakeholders

support specific research by contributing data on their personal health by providing access to measurements and medical evaluations, they could share their music listening history with analysts to better understand trends in media consumption, or share sensor data of their IoT enabled devices to provide data for climate research. They use the decentralized open-source storage technology *Solid*, to store their personal data and maintain control over their data and access to it, ensuring that data sharing aligns with their consent at all times. This role is central to the data sovereignty of our proposed platform, as it does not rely on a central data silo to store user data. Therefore, *Data Contributors* are no longer passive participants but active decision-makers in the data-sharing process similar to the *Participant* in [SOSD23].

3.1.2 Data Consumer

Data Consumers are individuals, similar to the roles *Analysts* and *Researcher* introduced by [SOSD23, EEM+21] who use the platform to gather data that aligns with their research goals, such as analyzing data points to produce insights, create anonymized datasets, or train statistical models. Unlike the approach followed by [EEM+21], where *Analysts* query a central database in which users have already donated their data, we follow the same approach Sun et al. in [SOSD23], namely the creation of access requests that must be actively consented to by the *Data Contributor*.

3.1.3 Auditor

An *Auditor* is an external stakeholder who seeks to review and verify the processes and results of data analyses conducted on the platform. By examining detailed records and logs maintained by the platform, *Auditors* can trace how analysis results were derived, including which data subsets were used and the methods applied. This capability is essential for answering specific audit questions that may arise in various contexts, such as litigation cases, regulatory compliance checks, or internal reviews.

A special form of *Auditor* may be a *Data Contributor* wishing to understand how and when their personal data was utilized. This role underscores the platform's attention to data sovereignty, as it allows *Data Contributors* to verify that their data have been accessed and used in accordance with their consent and the platform's policies. Auditing relies on the platform's robust logging and tracking mechanisms.

3.1.4 Platform Provider

We further introduce the role of the *Platform Provider*, responsible for developing, maintaining, and managing the data platform's infrastructure. The stakeholder also aims to provide up-to-date and relevant data for analysis to remain competitive in the evolving space of data providers. Integrating technologies such as Solid is one possible strategy to connect new data sources and become more attractive for data sharing.

3.1.5 Pod and Identity Provider

Pod Providers and *Identity Providers* are integral to the *Solid* ecosystem's decentralized approach to data storage and user control. *Pod Providers* host *Solid Pods*, storing personal data and building on WebIDs for access control. These WebIDs are managed by *Identity Providers*, who handle user authentication and authorization, ensuring that only authorized users can access data. Although both types of providers can be hosted by the same entity or under the same domain name, they serve distinct roles within the ecosystem. Notably, a *Data Consumer* can be his own *Pod Provider*, *Identity Provider*, or both.

3.2 Use Cases

In Section 3.1 we identified 6 stakeholders, the use cases will be mainly relevant to the 3 roles that represent the users of the proposed solution. There are four use cases that are relevant to the users:

1. **UC1 Contribute Data:** In this use case, a person wants to participate in research by providing access to his personal data. He is motivated to support research and help finding insights and knowledge about specific topics, but wants to stay in control over his data sharing and be sure, the data is only used for research purposes he agrees to.

2. **UC2 Consume Data:** In this use case, there is an analyst that wants to research on a particular topic and needs data to run his experiments. He uses the proposed data platform to request access to the needed data. He provides information on

the data, usage, and goal of the research and publishes his data request on the platform.

3. **UC3 Change Consent:** In this use case, the *Data Contributor* has already accepted one or more data requests, but decided that he no longer wants to share his data to a specific case and revokes his consent. The data and access controls created for the specific request are removed from the users pod and data access is no longer possible for the data platform.

4. **UC4 Audit Analysis:** In this use case, an auditor checks the chain of events for a research outcome. Having a complete lineage with rich metadata helps to understand and verify the entire process from data request to final analysis. The *Auditor* ensures that all data used in research adheres to the specified consent parameters and checks compliance with both internal policies and external regulations. The *Auditor* uses tools integrated into the platform that allow for tracking and verifying every transaction and data access, ensuring transparency and accountability in the research process.

3.3 Functional Requirements

Based on the requirements of [EEM$^+$21] and [SOSD23], we consolidated a list of functional requirements, presented in Table 3.1, that data platforms should meet.

Table 3.1: Functional Requirements of the Proposed Data Platform

ID	Description
FR1	The system shall allow *Data Consumers* to query and analyze data from participating *Data Contributors*.
FR2	The system shall enable data analysis to be performed within isolated, temporary environments, ensuring that raw data remains protected and only aggregated or anonymized results are shared.
FR3	The system shall log all data access and processing events with provenance information.
FR4	The system shall allow *Data Contributors* to only share data, that is relevant in the context of a accepted study.
FR5	The system shall ensure that only the minimum necessary data required for a specific analysis is authorized for transfer from data contributors, in line with the principle of *data minimization*.
FR6	The system shall use standardized data formats and ensure interoperability across different platforms and systems.
FR7	The system shall allow *Data Contributors* to have full transparency over which data elements are shared, with whom, and for what purposes, at all times.
FR8	The system shall allow *Data Contributors* to withdraw their consent at any time, and the system shall enforce such withdrawal immediately.

FR1 - Data Query & Analysis
The core function of any data platform is to facilitate data sharing between *Data Contributors* and *Data Consumers*. Drawing inspiration from the TIDAL platform described by Sun et al. in [SOSD23], which allows researchers to access data through decentralized storage pods, our platform will provide a similar mechanism. *Data Consumers* should be able to specify their data needs so that they can perform targeted analyses on relevant datasets.

FR2 - Isolated Analysis Environment
Our platform will facilitate data analysis within isolated, yet not immediately transient environments. This approach is an improvement over the model used by [SOSD23], where data is processed in ephemeral Docker containers that are destroyed immediately after the analysis is complete. Changing parameters or changing other details in the analysis process means that a new version of the Docker image used must be published and a new data request must be issued. This method works for quick one-time analyses, but restricts the ability to have a deeper, ongoing interaction with the data.

In contrast, our platform utilizes DataSHIELD within isolated environments, enabling extended and detailed analyses without compromising the privacy of sensitive data. DataSHIELD operates under a client-server architecture, ensuring that individual personal data remain secure on the server side, managed by the data platform, and never directly accessed or viewed by *Data Consumers*, which interact with the data through predefined privacy-preserving functions. This approach not only supports more complex and iterative analysis processes, but also adheres to data protection standards by only returning non disclosive summary statistics to the client. Thus, while our environments allow for prolonged data retention to facilitate comprehensive analyses, they maintain robust privacy safeguards, ensuring that data is responsibly managed and eventually deleted only after the analytical need is fulfilled.

FR3 - Provenance
Following the architecture proposed by Ekaputra et al. [EEM$^+$21], our platform integrates support for *Solid Pods* as a data source. Ekaputra et al. [EEM$^+$21] effectively tracks detailed audit trails throughout the data processing lifecycle with their proposed *Audit Box* and ensures robust compliance with GDPR standards and allows for verification of data integrity by *Auditors*. Our proposed system shall collect provenance information that documents data access and retrieval events and be able to provide this information in an machine-actionable way, so that it can be integrated into existing provenance solutions.

FR4 - Study-Specific Sharing
This functional requirement builds on the principles set by Sun et al. [SOSD23], improving the approach by incorporating specific request-based data sharing. In contrast to [EEM$^+$21], where data can be stored and accessed for potential future research without immediate necessity, our system requires data to be only shared after an explicit and approved research request. This modification, inspired by [SOSD23] ensures that *Data Contributors* have absolute control over their data, sharing only when there is a direct

and confirmed need, thus reinforcing data sovereignty. This practice ensures that all data transfers are justified and purposeful, minimizing unnecessary exposure.

FR5 - Data Minimization
We implement *data minimization* by ensuring that only the essential data required for specific analyzes are transferred, refining the approaches seen in [EEM+21, SOSD23]. In [SOSD23], entire datasets are transferred to Docker containers, where the relevant data are then extracted. This method potentially moves more data than necessary, only filtering out the excess at a stage where privacy risks have already increased. In contrast, our system implements upfront data filtering, ensuring that only the data essential for a specific analysis is transferred from the start. This proactive approach minimizes the volume of data moved and processed, reduces the risk of exposure, and aligns more closely with *data minimization* principles.

FR6 - Interoperability
We will use well-known data structures such as RDF for data saved and processed inside *Solid Pods*, building on the standard of semantic web technologies. For greater accessibility and ease of integration, the *Solid Gateway* will provide access to API endpoints in JSON format. Data retrieved from *Solid Pods*, such as the data from *Data Contributors*, and data uploaded to *Solid Pods*, such as data requests or consent files, will be in RDF format and use well established ontologies such as FHIR for data and DPV for consent management.

FR7 - Sharing Transparency
We ensure that *Data Contributors* have complete transparency over the data they share, who accesses them, and for what purpose. We will adopt the data request approach used by [SOSD23], which clearly outlines the specific data elements required for each study. This method allows contributors to see exactly which aspects of their data are being utilized, enhancing their understanding and control over data usage.

Solid's architecture ensures that access permissions are managed by the pod owner, explicitly indicating which files are accessible and to which WebID. This is complemented by detailed request files that specify the purpose of the data access, among other details, offering contributors a comprehensive view of how and why their data are being used. With our *Granular* approach, we provide a mechanism that filters out only relevant data from the *Data Contributor* and create a separate data file which is authorized for transfer. This way, *Data Contributors* can inspect what data exactly will be transferred.

FR8 - Consent Withdrawal
Data Contributors can withdraw their consent at any time, with the platform enforcing this withdrawal immediately. This characteristic underscores the importance of respecting the autonomy and control of *Data Contributors* over their personal information.

Using the Solid framework, *Data Contributors* manage their data stored in personal pods, where they can directly adjust access permissions or completely revoke them using ACL files. This immediate control mechanism ensures that *Data Contributors* can swiftly withdraw their consent, reflecting changes in real-time and giving sovereignty to the *Data Contributor*.

3.4 Non-Functional Requirements

While functional requirements ensure that the system operates correctly in terms of data access, sharing, and governance, non-functional requirements define the quality attributes that impact its usability and efficiency. In the context of this thesis, two key non-functional aspects are evaluated: performance and minimization of data exposure.

3.4.1 Performance Efficiency

Performance is a key consideration when retrieving data from decentralized storage, as the overhead introduced by *Solid Pods* compared to traditional centralized data platforms must be understood. The goal is to assess how different retrieval strategies affect the overall efficiency of the system.

To evaluate performance, we conduct an experiment comparing *Standard* retrieval strategy, which fetches all available data from a pod in a single request, with a *Granular* retrieval strategy, which selectively retrieves only the necessary data based on the data request. The evaluation focused on measuring three key performance metrics:

- **Total Fetch Time**: The time taken from initiating a fetch request until all data has been received.
- **Total Processing Time**: The time required to process and filter the received data to extract relevant information.
- **Number of HTTP Requests**: The total number of requests sent from the Solid Gateway to the *Solid Pods*.

Subsequently, we compare our results to *WellFort* [EEM+21] to evaluate how our solution compares to the original implementation using a local data repository.

In addition, we evaluate to what extent we can further improve the performance of the *Solid Gateway* through parallelization of the fetching and processing of data received from *Solid Pods*.

3.4.2 Minimization of Data Exposure

Since *Solid* stores data in RDF format, the information is structured as triples, where each triple consists of a subject, a predicate and an object. The risk of unwanted data exposure occurs when fetching more data than is required for a given request. Wagner et al. [WE18] give an overview of various privacy metrics in their proposed framework for privacy measurement. We chose metric m, *Amount of Leaked Information* which is defined as the absolute number of leaked data points:

$$m \equiv |L|$$

To assess data exposure, we perform an experiment to measure the number of triples transferred under different retrieval strategies. The primary objective was to quantify how much unnecessary data is retrieved and to what extent a more granular approach can reduce the amount of expose. We analyze the following metrics:

- **Total Number of Transferred Triples**: The total number of RDF triples retrieved from *Solid Pods*.
- **Unwanted Data Leakage**: The number of triples transferred that were not explicitly required by the data request, thus representing $|L|$.

The *Standard* strategy corresponds to the data-sharing model implemented in *TIDAL* [SOSD23], where complete files are shared if any part of them is relevant to a request. This allows us to directly evaluate how our *Granular* approach compares to *TIDAL* in terms of data exposure. Furthermore, we set our results into context with *WellFort*, where entire personal datasets are stored centrally, even before a specific request is made.

3.5 Summary

We introduce key stakeholders based on responsibilities and interactions within the system in Section 3.1, categorizing them into users: *Data Contributors*, *Data Consumers*, and *Auditors* and providers: *Platform Providers*, *Pod Providers*, and *Identity Providers*. Next in Section 3.2, we provide use cases relevant to user interactions with the platform, including contributing data, consuming data, changing consent, and auditing analysis.

In Section 3.3 we propose a set of functional requirements derived from the reference implementations [SOSD23, EEM+21]. Key aspects include the ability for *Data Contributors* to selectively share data based on explicit requests, the integration of secure analysis environments using DataSHIELD, and the implementation of robust logging mechanisms for auditing and compliance.

Finally, in Section 3.4 we define non-functional requirements, focusing on performance and minimization of data exposure. We give metrics to evaluate different data retrieval strategies, comparing standard and granular approaches in terms of retrieval time, processing overhead, and data exposure levels.

CHAPTER 4

Conceptual Design

In this chapter, we implement a prototype to establish Solid as a decentralized data store for *Data Platforms*. We base our design on [EEM+21] and [SOSD23]. We provide an overview of important processes and show at a high level how stakeholders interact with our proposed solution. We illustrate these interactions and steps using flow charts that are intentionally kept at a high level to represent the relevant business processes. These visualizations highlight the functional requirements that our system aims to fulfill more effectively than the systems on which we base our design.

4.1 Architecture

This section outlines the complete conceptual architecture of the *Data Platform* into which our proposed solution is integrated, as illustrated in Figure 4.1. Our design builds upon *WellFort*, the privacy-preserving data platform architecture introduced by Ekaputra et al. [EEM+21].

We retain the design principles of the *Trusted Analysis Environment*, which ensures that raw data access is separated from analytical operations. In our setup, the *Analysis Server* is also based on Opal, a data management system that handles the controlled import, transformation, and export of structured data. Opal provides a REST API which we use to automatically import data from the *Loader*. A temporary *Analysis Database*, implemented with PostgreSQL, stores the session-specific datasets required for analysis. The analysis itself is performed through an *Analysis Interface* built on RStudio Server, which uses DataSHIELD to execute statistical methods with embedded privacy controls that prevent the disclosure of sensitive information, forming a privacy-preserving computation layer between data and consumer.

For our implementation we modify the *Secure Repository*, as we propose a solution to replace the local data repository, which is implemented using Invenio in [EEM+21]. We color components that we introduce, such as the *Solid Gateway* and the *Request Portal* orange. Components we adapted, such as the *Loader*, *Controller* and *Experiment Setup Interface* are highlighted in purple.

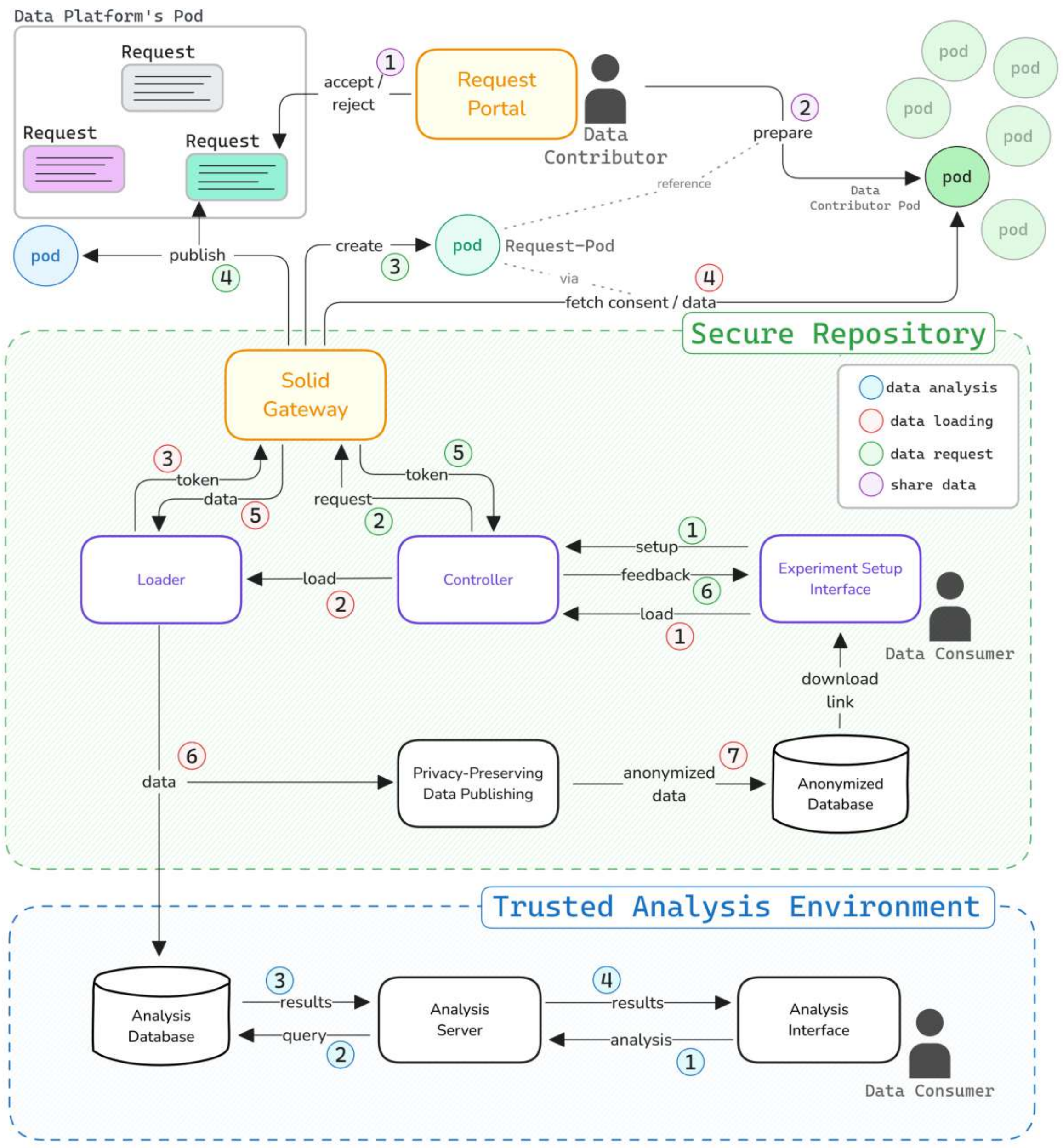

Figure 4.1: Conceptional architecture

We adapt the *Experiment Setup Interface* by adding views for the *Data Contributors* to create and overview their data requests. Further, we adapt the *Controller* to communicate with our *Solid Gateway* for request creation (green markings) and the *Loader* to fetch data via the *Solid Gateway* instead of the data repository (red markings).

As we introduce *Solid* into our architecture, we take inspiration from Sun et al. [SOSD23] as they already explored request-based data sharing in TIDAL. We are not using their code, as we were not able to run their provided application. Instead, we took their general approach and implemented the needed workflows into our our proposed *Solid Gateway* and *Request Portal*. The *Solid Gateway* is a NodeJS[1] application using Solid-Node-Client[2] for working with *Solid* and rdfjs[3], as well as n3[4] for parsing and creating RDF data. The *Request Portal* is a *Solid* application based on React[5] and uses the solid-client[6] library next to rdfjs and n3.

Specifically, we also create request-sepcific WebIDs and pods through our *Solid Gateway* so that data sharing for different requests is also distinguished by different WebIDs. For simplicity, we publish requests in one central *Data Platform Pod* using DPV which is scanned by our *Request Portal*. For the *Request Portal* we implement the *Standard* approach, which is built according to Sun et al. proposed *Algorithm 2* [SSD23]. Further, we propose our own *Granular* approach to improve *data minimization* when accepting a data request.

In summary, the proposed architecture extends the privacy-preserving design of *WellFort* by replacing its centralized *Secure Repository* with a decentralized *Solid*-based data sharing mechanism. We introduce the *Solid Gateway* and *Request Portal* as new components that facilitate request-based access and consent-driven data sharing from external *Solid Pods*. While preserving the analytical workflow of the *Trusted Analysis Environment*, our system enhances data sovereignty through dynamic request-specific sharing strategies and our novel *Granular* data sharing approach. These components together enable a user-centric architecture that aligns with *data minimization*.

4.2 Components

To contextualize the flowchart visualizations presented in the following sections, we first introduce the core components and stakeholders involved. At this stage, we narrow our focus to the *Solid Gateway* and the *Request Portal*, without delving into the detailed architecture of the complete data platform. For this reason, we refer broadly to the *Data Platform*, omitting internal components such as the *Loader* or *Controller* used in *WellFort*, as these elements may differ across various platform implementations that adopt our proposed architecture.

[1]`https://nodejs.org/en`
[2]`https://www.npmjs.com/package/solid-node-client`
[3]`https://github.com/linkeddata/rdflib.js`
[4]`https://www.npmjs.com/package/n3`
[5]`https://react.dev/`
[6]`https://www.npmjs.com/package/@inrupt/solid-client`

- **Data Platform**: As described in Section 4.1, we base our architecture on the *Data Platform* proposed by Ekaputra et al. [EEM$^+$21]. We adapt their core components inside the *Secure Repository* to use our proposed *Solid Gateway* to retrieve data from externally hosted *Solid Pods* instead of using the centralized data repository.

- **Solid Gateway**: We highlight the *Solid Gateway* separately due to its role in our proposed system. It facilitates the retrieval of data from externally hosted *Solid Pods*, effectively replacing the reliance on the internal centralized data repository following the implementation of Sun et al. [SOSD23].

- **Solid Server**: Hosts *Solid Pods* and acts as an Identity Provider for WebID creation and authentication. It can be part of the *Data Platform* or managed by a trusted third party, ensuring secure, authenticated data interactions.

- **Request Portal**: We propose an *Solid* application that operates exclusively within the user's browser, without direct connection to the *Data Platform*. Its primary functions are to format the data appropriately for later ingestion by the *Data Platform* and to guide *Data Consumers* through the process of responding to data requests, including granting and withdrawing consent to data requests. Although the *Request Portal* is currently integrated with the *Data Platform*, this functionality could alternatively be managed by a third party or directly by *Data Contributors* themselves.

- **DP Pod**: This pod is used by the *Data Platform* to store all data requests generated for *Data Consumers*. Currently, these requests are publicly accessible, though future implementations might involve more restrictive authorization, possibly limiting access to only registered contributors through specific WebIDs. This pod ensures that *Data Contributors* can access and respond to these requests.

- **Contributor Pod**: This pod is directly associated with an individual, serving as a personal data storage space. For simplicity, we assume that each user manages one pod where their personal data is stored in an RDF document. This assumption was also chosen by Sun et al. in [SOSD23].

- **Data Contributor**: Individuals who own and manage their personal data, using *Solid*, can share specific data subsets in the context of data requests from *Data Consumers*. They contribute to research by providing access to personal data and actively participating in data sovereignty by deciding how and when their data are shared.

- **Data Consumer**: Researchers or analysts who seek data for their studies or analytics, creating access requests that require explicit consent from *Data Contributors*. This approach contrasts with centralized data access, emphasizing active consent and data sovereignty, similar to the TIDAL [SOSD23] platform model.

4.3 Controlled Data Sharing through WebID and ACLs

The *Solid Gateway* acts as an autonomous agent that manages the retrieval of data from distributed *Solid Pods* in response to published data requests. For each data request incoming from a *Data Consumer*, the *Solid Gateway* generates a dedicated WebID and an associated pod. This automated creation of WebIDs is used, so that each request operates under its own distinct identity.

Data Contributors who wish to participate in fulfilling the request can share relevant data directly using ACLs by explicitly granting read permissions to the WebID linked to the data request. This means that the *Solid Gateway*, acting on behalf of the *Data Consumer*, is only able to access the precise data, authorized by each individual *Data Contributor*, and only for the scope of that particular request.

When executing the data retrieval, the *Solid Gateway* authenticates itself using the WebID linked to the request. Upon receiving the access request, the *Contributor's Pod* verifies the *Solid Gateway's* identity via WebID authentication and evaluates the corresponding ACLs. If the permissions are valid, the data are transmitted securely via HTTPS to the *Solid Gateway* for further processing.

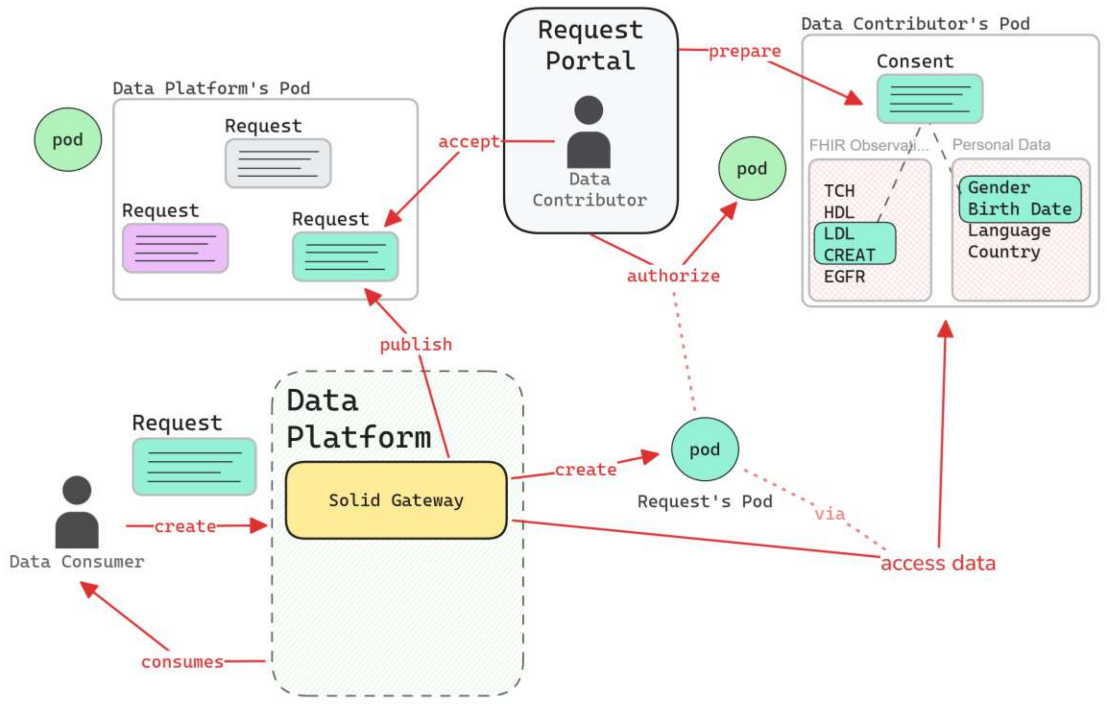

Figure 4.2: High level overview of the process of creating and accepting a data request using the *Solid Gateway* and *Request Portal*

This mechanism allows the *Solid Gateway* to orchestrate decentralized data collection across multiple pods while respecting the sovereignty and consent of individual *Data*

Contributors. In addition, each request is isolated within its own WebID and pod, which simplifies permission management, and provides a clear provenance trail that associates shared data with specific requests. *Data Contributors* can easily revoke access at any time by updating or removing relevant ACL entries, reinforcing the user-centric paradigm of this mechanism.

Figure 4.2 illustrates the high-level overview of the process of handling data requests within a Solid-integrated *Data Platform*. The workflow begins with a *Data Consumer* creating a Data Request. Upon receiving the request, the *Solid Gateway* generates a dedicated WebID and pod specifically associated with this request, which serves as the identity under which data retrieval will be performed. The WebID is added to the *Data Consumer's* request and is then processed by the *Data Platform* and published in the *Data Platform's Pod* (DP Pod) by the *Solid Gateway*, making it accessible to potential *Data Contributors*. If a *Data Contributor* chooses to accept the request, via the *Request Portal*, a consent file is created based on the request file in their personal pod, signifying their explicit consent to share the requested data. As part of this process, the necessary access control files are created, granting the WebID of the specific data request permission to access the relevant data. In the final step, the *Solid Gateway* authenticates using the request-specific WebID, fetches the authorized data from the *Data Contributor Pods*, and provides it to the *Data Consumer* in a privacy-preserving manner.

4.4 Workflows

This section outlines the key workflows of the proposed *Solid Gateway* and *Request Portal*, including request creation, consent management, and data retrieval, illustrating how the system enables user-centric data sharing in detail.

4.4.1 Creating a Data Request

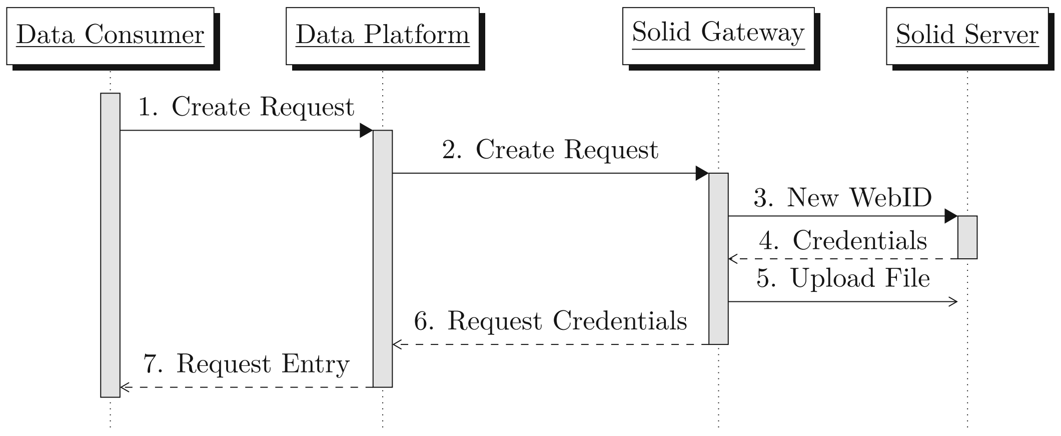

Figure 4.3: Creating a new data request

Figure 4.3 shows the steps involved in creating a new data request using our proposed system.

1. The process starts when a *Data Consumer* creates a request for data. Using an interface provided by the *Data Platform*, the *Data Consumer* creates a request by providing information on what data attributes he requests and how the data is going to be used. This request is then registered to the account of the *Data Consumer*.
2. The *Data Platform* then forwards this request to the *Solid Gateway*.
3. The *Solid Gateway* creates a new WebID from a Solid Server for this request. The credentials to authenticate as this WebID are only known by the *Solid Gateway*, effectively acting as a mediator between the *Data Platform* and *Data Contributors*.
4. The *Solid Server* returns an API key and API secret which is directly linked to the just created WebID back to the *Solid Gateway*. Using these API credentials, the Gateway can later prove authenticity when querying *Solid Pods*.
5. The *Solid Gateway* uploads the data request file to the public accessible *DP Pod*, which in this case is hosted on the same *Solid Server*.
6. The *Solid Gateway* then sends a new pair of credentials, which is directly mapped to the just created WebID, back to the *Data Platform*. With this set of credentials, the *Data Platform* can request the *Solid Gateway* to send status information and data for the request.
7. The *Data Platform* gives feedback to the *Data Consumer* by showing a new request in the overview of the *Data Platform's Request Portal*, confirming the creation of the data request.

Originally, in [EEM$^+$21], the data request is translated directly into a query that was executed immediately against the *Triplestore*.

With our proposed solution, querying data immediately after request creation is no longer possible. We introduce the constraint of giving explicit consent to each data request before allowing data retrieval, thus we have to give *Data Contributors* time to react to the published request.

4.4.2 Accepting a Request

We introduce workflows for a new component facing the *Data Contributor*. The *Request Portal* is a *Solid* application showing the created requests of the *Data Platform* and allowing the *Data Contributor* to give consent to share specific data for given data requests. The acceptance of a data request is visualized in Figure 4.4. We implemented a similar strategy as proposed in [SSD23], where upon acceptance access to the complete

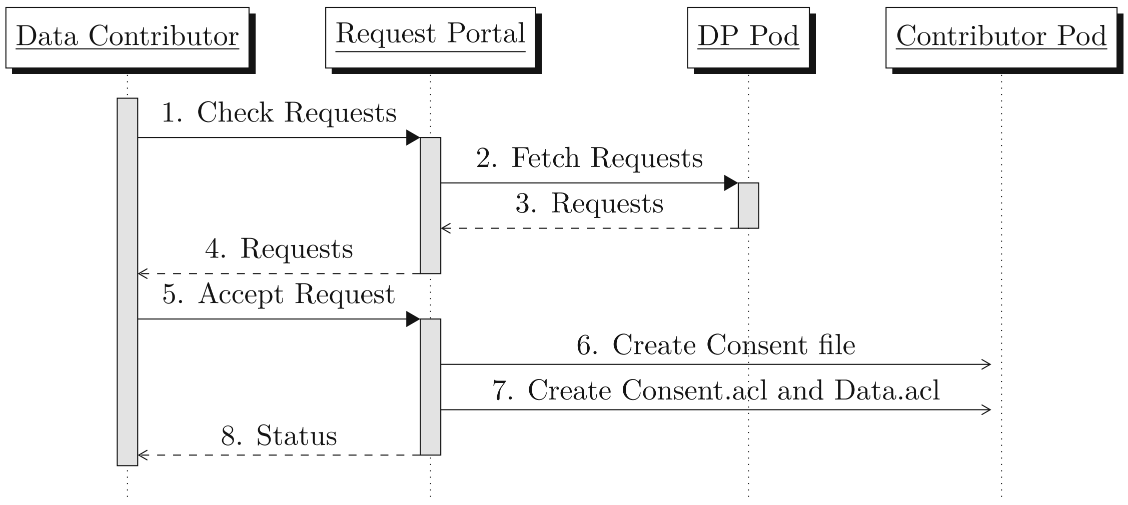

Figure 4.4: Accepting a data request with naive approach

data file of the *Data Contributor* is granted. We will refer to this baseline implementation, the *Standard* approach.

1. The *Data Contributor* uses the *Request Portal* to list all available data requests.
2. The *Request Portal* fetches the *DP Pod*, which hosts all created data requests publicly.
3. The *DP Pod* returns a list of all requests, which are essentially RDF files that contain all relevant information about the data request.
4. The *Request Portal* displays all open requests for the user to inspect.
5. The *Data Contributor* gives his consent to a data request by pressing "Accept" on the *Request Portal*.
6. The *Request Portal*, which is authenticated with the WebID of the *Data Contributor*, creates a consent file within the personal pod of the *Data Contributor* that signals the acceptance of the data request.
7. Furthermore, the *Request Portal* creates an ACL file, which gives access rights to the WebID, referenced in the data request, as can be seen in Listing 5.3, for both the consent and the actual data file.
8. The *Data Contributor* then sees that the request was accepted.

As shown in Figure 4.5, the *Solid Pod* of the *Data Contributor* contains three important files: the consent file, an ACL file that grants read access to the requested data (Data.acl)

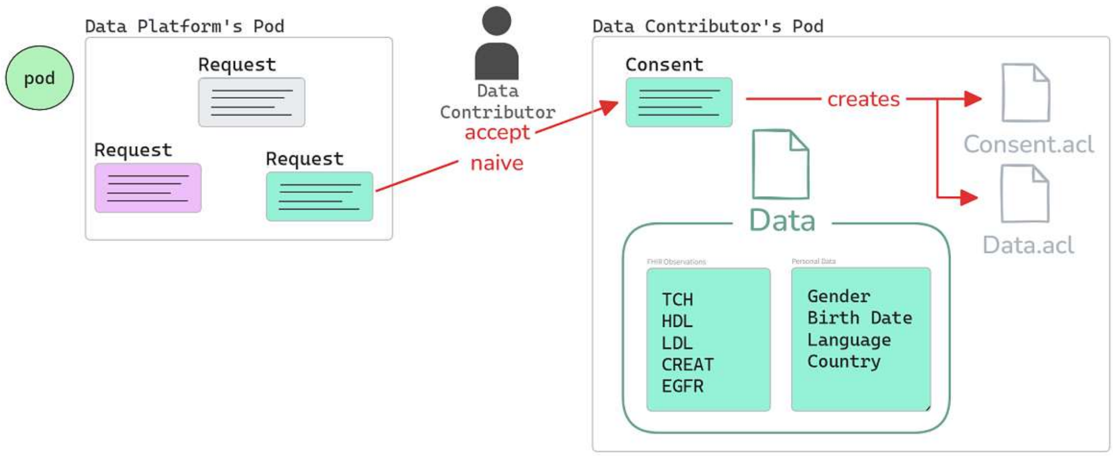

Figure 4.5: Visualization of Actions of the *Request Portal* for the *Standard* approach

and another ACL file that grants read access to the consent file (Consent.acl) for the WebID specified in the request.

The consent file created is also an RDF file that adheres to `dpv:consent`. It includes a reference `dpv:hasConsentNotice` to the original request file, an expiration time `pdv:hasExpiryTime` and the date of acceptance `schema:hasProvisionTime` as can be seen in Listing 5.7.

In *Solid*, data access takes place on a file basis, which means that the entire file is transferred to the client, which is then responsible for parsing and filtering the content. This method has a significant drawback, as the server does not deliver only the relevant data segments. The same limitation was also observed in the implementation proposed by [SOSD23].

To address these issues, we have optimized the *Standard* approach to minimize data sharing. Our goal is to reduce both the amount of data that must be authorized for access and the volume of data transferred during a request, thus preventing unnecessary data leakage. We propose the *Granular* approach, an implementation that reduces data exposure by authorizing and transferring only relevant data. While still transferring entire files, we create data files that contain only relevant data points.

Figure 4.6 shows our implementation of this approach. The first 5 steps are the same as in 4.4, thus starting with step 6.

6. The *Request Portal* checks the *Contributor's Pod* for the requested data, which is specified in the requests file. In our implementation, we check for FHIR `Patient` and `Observation` triples using rdfjs[7]. If not all requested triples can be found within the pod's data, the accept routine fails.

[7]`https://rdf.js.org/`

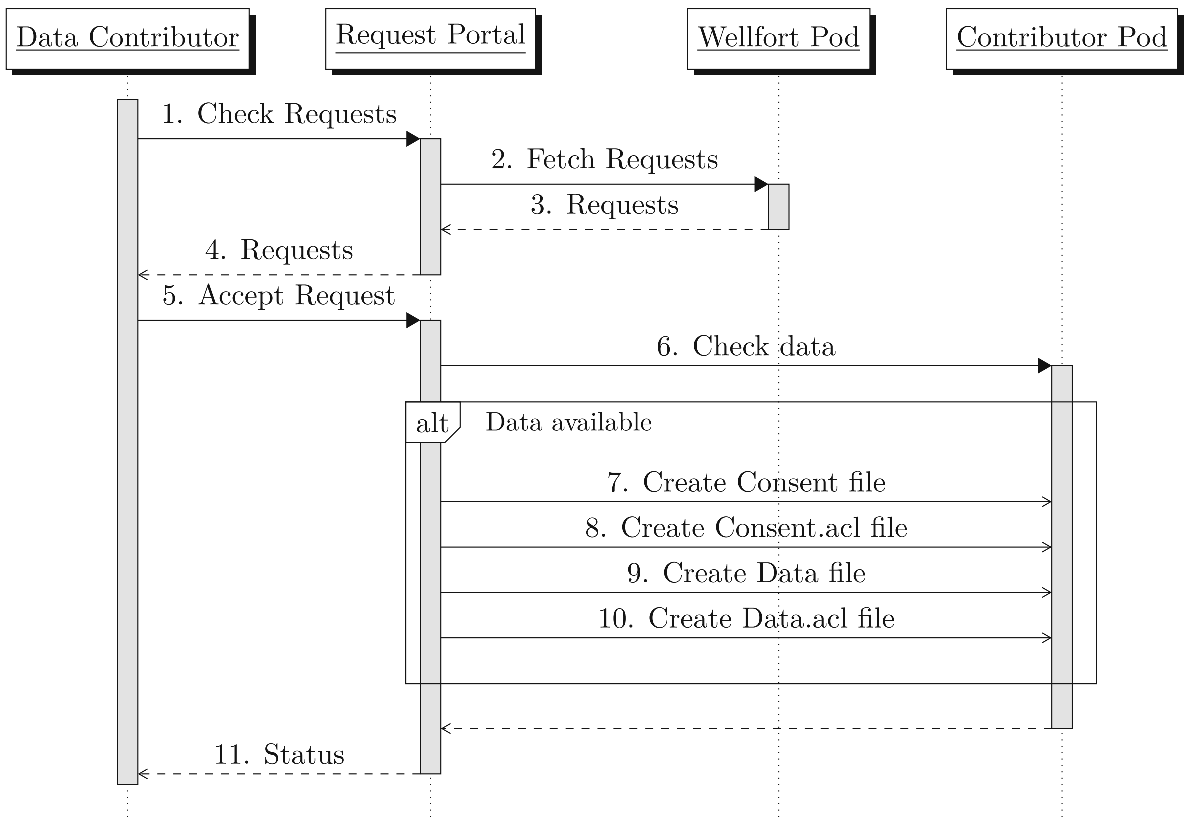

Figure 4.6: Accepting a data request with granular approach

7. A consent file is saved to the *Contributors Pod*, signaling consent to the *Data Platform*.

8. An ACL file is created, so that the WebID associated with this request can later fetch the consent file.

9. Each requested `Observation` that matches the requested `fhir:code` as well as the requested information of the `Patient`, is written to a new data file.

10. Further, an ACL file is created, giving the WebID of the request read access to the created data file.

In Figure 4.7, the *Granular* algorithm extracts the relevant data from the main data file. The *Request Portal* then creates a folder (Request) containing a new RDF file (Request/Data), containing only the relevant information from the *Data Contributor's* data, and two ACL files to allow read access to consent (Consent.acl) and data file (Request/Data.acl).

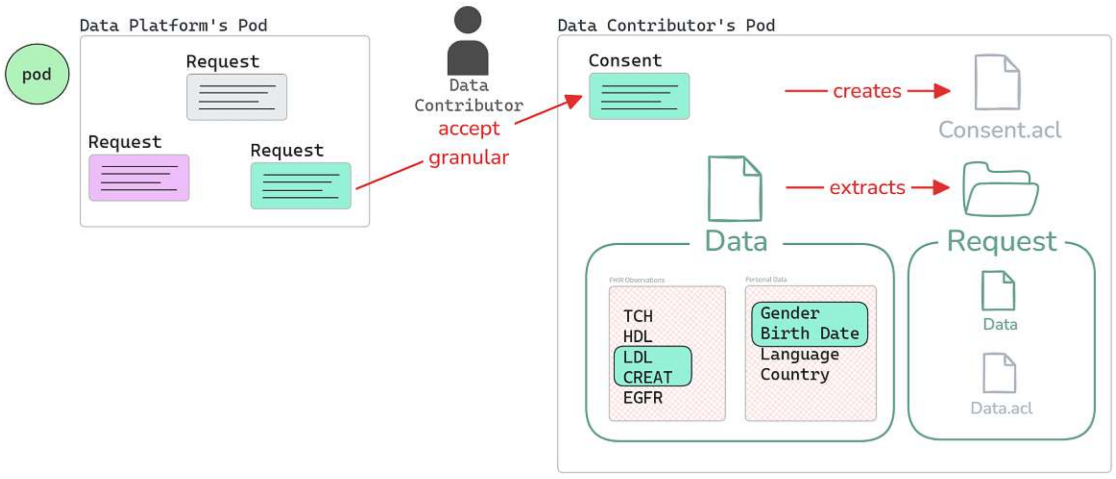

Figure 4.7: Visualization of Actions of the *Request Portal* for the *Granular* approach

As we implemented the *Request Portal* as a *Solid* application, all operations occur isolated from the *Data Platform*, the application only interacts with the *DP Pod* in order to fetch open requests and the *Data Contributors Pod* for consent and data preparation.

4.4.3 Retrieving Data

To ensure up-to-date consent, we added a mechanism within the *Solid Gateway* that verifies the number of accepted requests immediately before data retrieval. In contrast, [SOSD23] employs trigger messages—files saved to a *Solid Pod* to indicate when consent is given. However, we contend that this approach risks outdated consent records. Specifically, if a *Data Contributor* withdraws consent using another application or service that does not generate a trigger message, the stored consent information may not reflect the current status. Figure 4.8 shows the steps of our implementation of this check.

1. The *Data Consumer* triggers the process by requesting a status update on the created data request.
2. The *Data Platform* queries the *Solid Gateway* using the participant endpoint `E2` as described in Table 4.1.
3. The *Solid Gateway* goes through the registered *Data Contributors*, a list of WebIDs that is stored within the internal database of the *Solid Gateway*, as can be seen in Figure 4.12. For each WebID, the pod is queried for the consent file, which is created in a specific file location on acceptance, as described in Figures 4.4 and 4.6.
4. For every WebID, where a consent file is found, the pod returns a positive response.
5. The total amount of accepts is returned to the *Data Platform*.

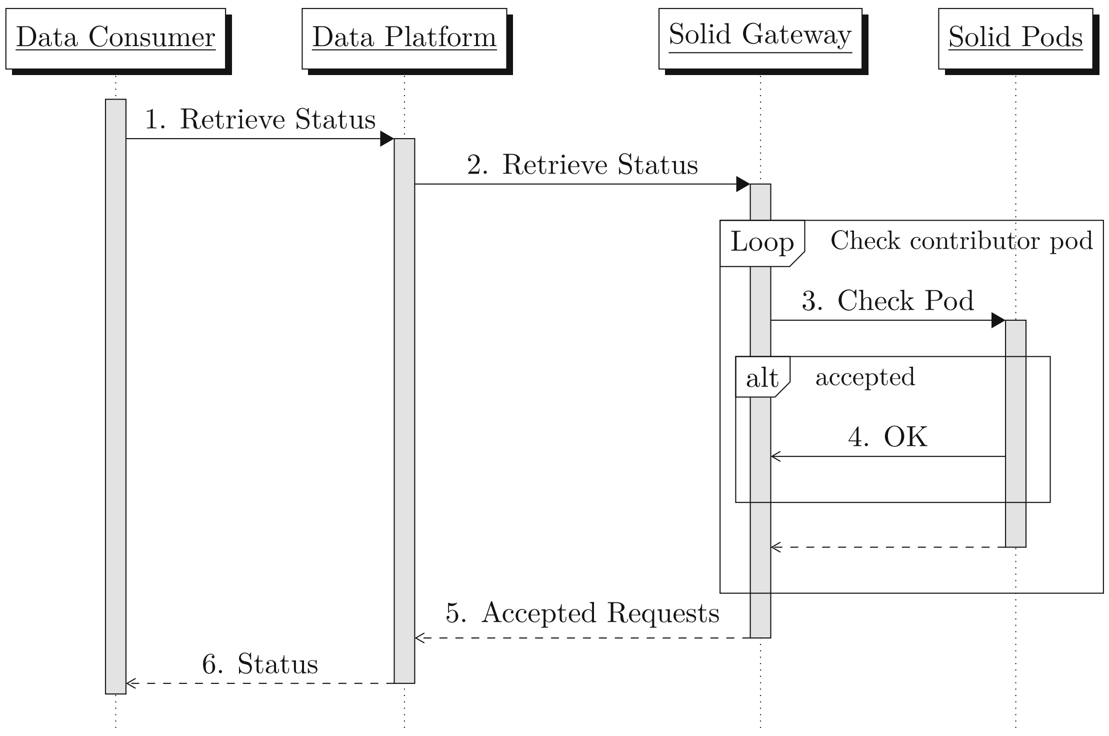

Figure 4.8: Checking the status of a request

6. The status is shown to the *Data Consumer*.

Figure 4.9 shows the sequence of actions involved in obtaining data from consented requests, which assumed a threshold of accepts, demonstrating how our system ensures that only authorized data are retrieved with respect to data sovereignty. This diagram explains the process from the perspective of a *Data Consumer* looking to access the data that has been granted through the consent mechanism described in Figures 4.6 and 4.4.

1. At any point in time, *Data Consumers* can initiate the data fetch routine for one of their requests.
2. The *Data Platform* queries the *Solid Gateway* using the data endpoint `E3` as described in Table 4.1.
3. The *Solid Gateway* performs a count of accepts by querying all *Data Contributors Pods* for the data request. Each pod with a valid[8] and accessible consent file is counted. Should the necessary threshold of accepts not be reached, the *Solid Gateway* returns an error as can be seen in Figure 4.10.
4. The *Solid Gateway* then retrieves the requested data from each pod.

[8]The consent is valid if the referenced data request matches and expiry time is not reached.

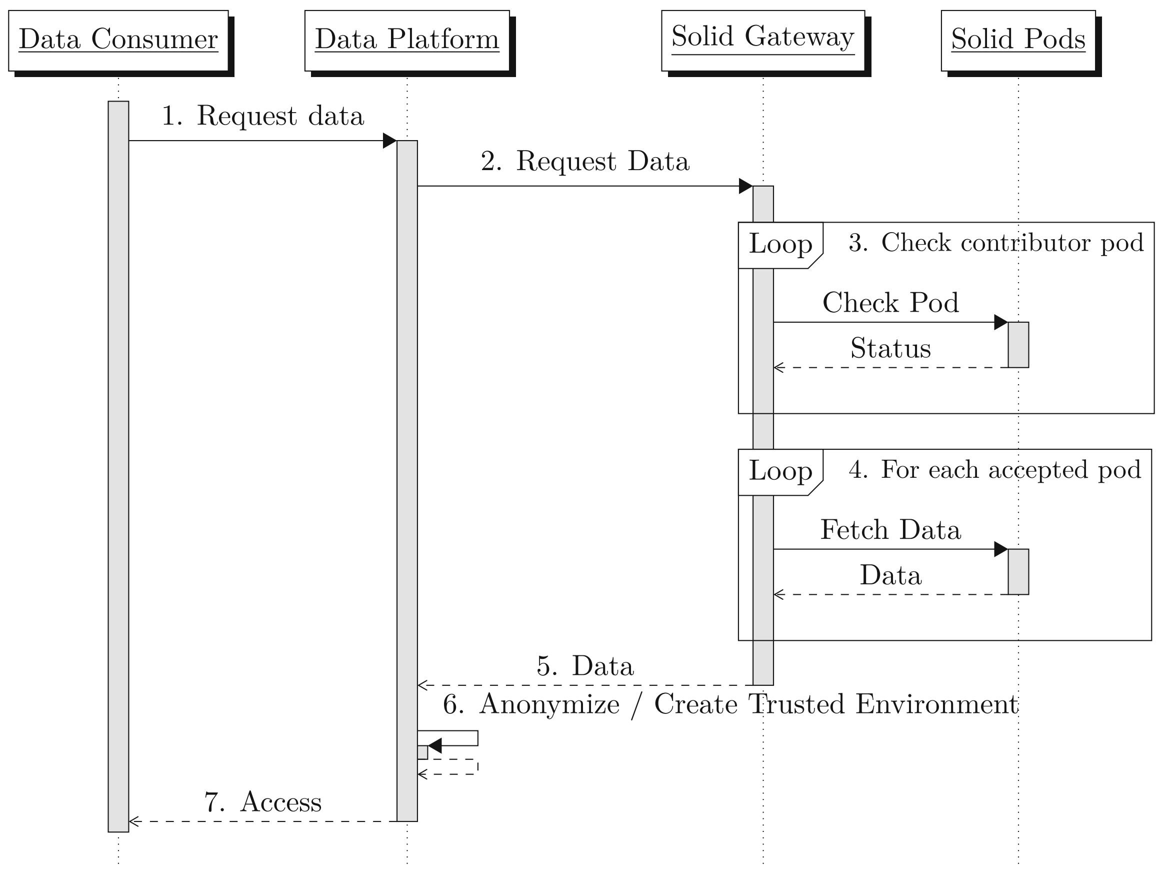

Figure 4.9: Fetching data for a request

5. The data is then passed on to the *Data Platform*.
6. Based on the *Data Consumers* choice, a *Trusted Environment* is created or an anonymized file is created using k-anonymisation or synthetic data generation.
7. The *Data Consumer* is provided with an anonymized data file or presented login information for a temporary R environment which provides access to the data through a privacy preserving layer using DataSHIELD as in [EEM$^+$21].

4.4.4 Withdraw Consent

Data Contributors can, at any time, reject their consent to any request they have previously accepted using a simple workflow visualized in Figure 4.11. Steps 1 to 4 are used to fetch the accepted requests from the *Data Contributor's Pod*. With one click, the *Data Contributor* can trigger Step 5 to reject a previously accepted request. The *Request Portal* then continues to delete the ACL file, denying read access to the data file, using an HTTP DELETE request and adding `dpv:hasWithdrawalTime` with the current

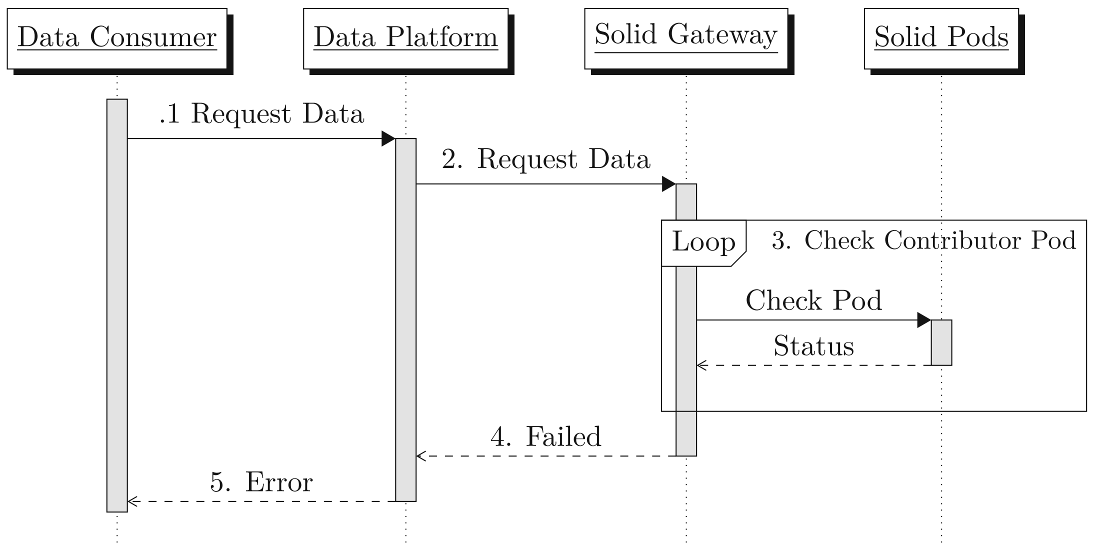

Figure 4.10: Fetching data for a request but not reaching threshhold

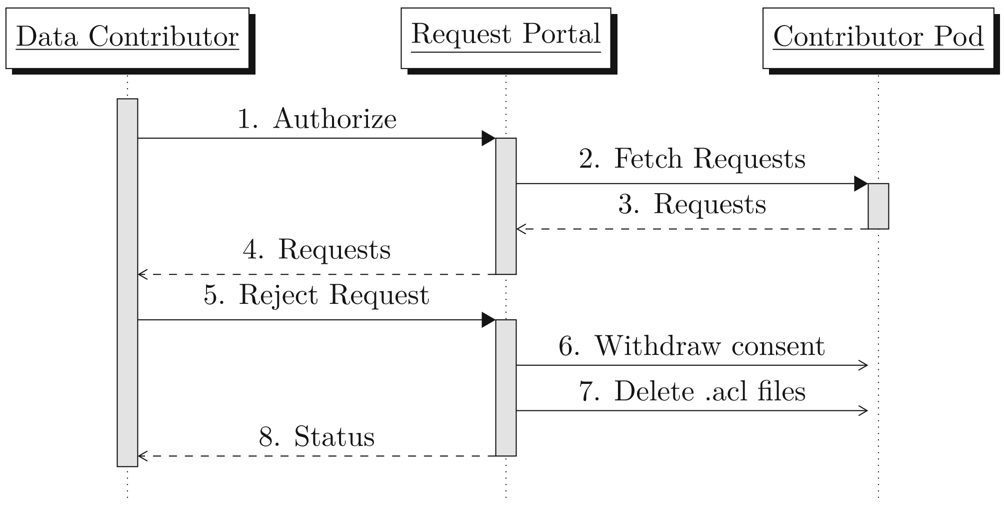

Figure 4.11: Rejecting a data request

time to the consent file. Alternatively, the *Solid Gateway* also treat consent withdrawn if the consent file is no longer accessible, indicated by a HTTP 403 Forbidden or 404 Not Found error.

4.5 Solid Gateway API and Data Management

The *Solid Gateway* provides a set of HTTP endpoints to handle data requests, retrieve data, and manage contributor registrations. In addition to these primary functions, it is also essential to support transparency and traceability within the system by tracking

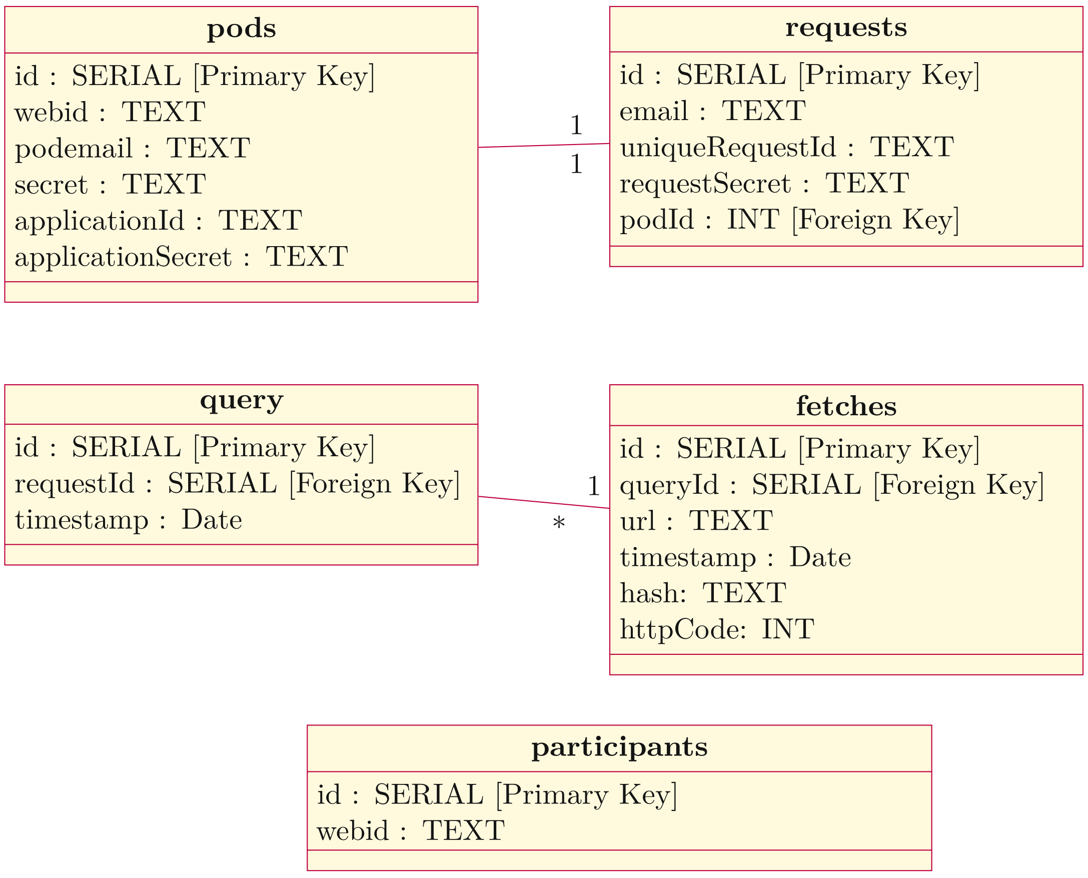

Figure 4.12: Solid Gateway UML

WebIDs, data requests, and provenance information. To reinforce these aspects, we align parts of our implementation with the principles of FAIR data management and selected the RDA Data Citation Recommendations by Rauber et al. [RAUP16].

Since we avoid persistent storage of personal data to protect the privacy and sovereignty of *Data Contributors*, mechanisms such as global identifiers, versioning, or permanent citation records are not compatible with our proposed design. Our focus remains on minimizing data exposure and maintaining full user control, which currently limits the application of recommendations aimed at long-term traceability and reuse.

The *Solid Gateway* uses a structured relational database to manage data requests and track provenance, consisting of five key tables: *pods*, *requests*, *query*, *fetches*, and *participants*, as shown in Figure 4.12. For each new data request, a row is added to the *requests* table, linking it to a specific *Solid Pod* via the *podId* foreign key. The *pods* table stores metadata about *Solid Pods* managed by the *Solid Gateway*, including WebIDs and the corresponding credentials to authenticate it's queries against Solid Servers. The *query* table logs every instance in which the *Solid Gateway* fetches data, storing timestamps to provide a temporal record of retrieval attempts over time, addressing the need for basic timestamping as suggested in R7 (Query Timestamping) [RAUP16].

Each individual data retrieval is recorded in the *fetches* table, which references the corresponding *query* entry and stores the URL of the accessed resource, the http code returned, a timestamp and a cryptographic hash of the retrieved content. The hash functions as an integrity check, allowing to detect modifications in the same resource across repeated retrievals, in line with the verification of result sets as proposed in R6 (Result Set Verification) [RAUP16]. This approach ensures that, despite the system's focus on privacy and temporary data handling, the integrity and consistency of data fetches can be monitored and verified over time. However, due to the scope of the thesis, other aspects of the RDA recommendations are not implemented.

The Participants table, which remains independent of the other tables, maintains a list of WebIDs of *Data Contributors* who have been registered with the *Solid Gateway*. These WebIDs are used to determine which *Data Contributors* should be queried for request acceptances.

The *Solid Gateway* offers five machine-actionable HTTP endpoints using JSON as content type, as seen in Table 4.1.

E1 is used to create a new request as described in 4.3. An identification of the *Data Consumer* as well as the data requests details are submitted with the body of the HTTP POST request to the *Solid Gateway*. Figure 4.13 shows the creation of a new WebID in detail. All credentials generated by the *Solid Gateway* are created randomly using the crypto.js[9] randomBytes function.

E2 is used to check the amount of accepts for a given data request. Authentication is provided by the *Data Platform* in the form of two HTTP headers.

E3 is used to retrieve data for a given data request. Authentication is provided by the *Data Platform* in the form of two HTTP headers.

E4 is an internal endpoint that returns the current state of the *Solid Gateway*. It should be integrated into the monitoring of the *Data Platform*.

E5 is the only external endpoint. It is used to register as a *Data Contributor* by providing a WebID, which is saved in the internal database of the *Solid Gateway*.

[9] `https://github.com/nodejs/node/blob/v23.5.0/lib/crypto.js`

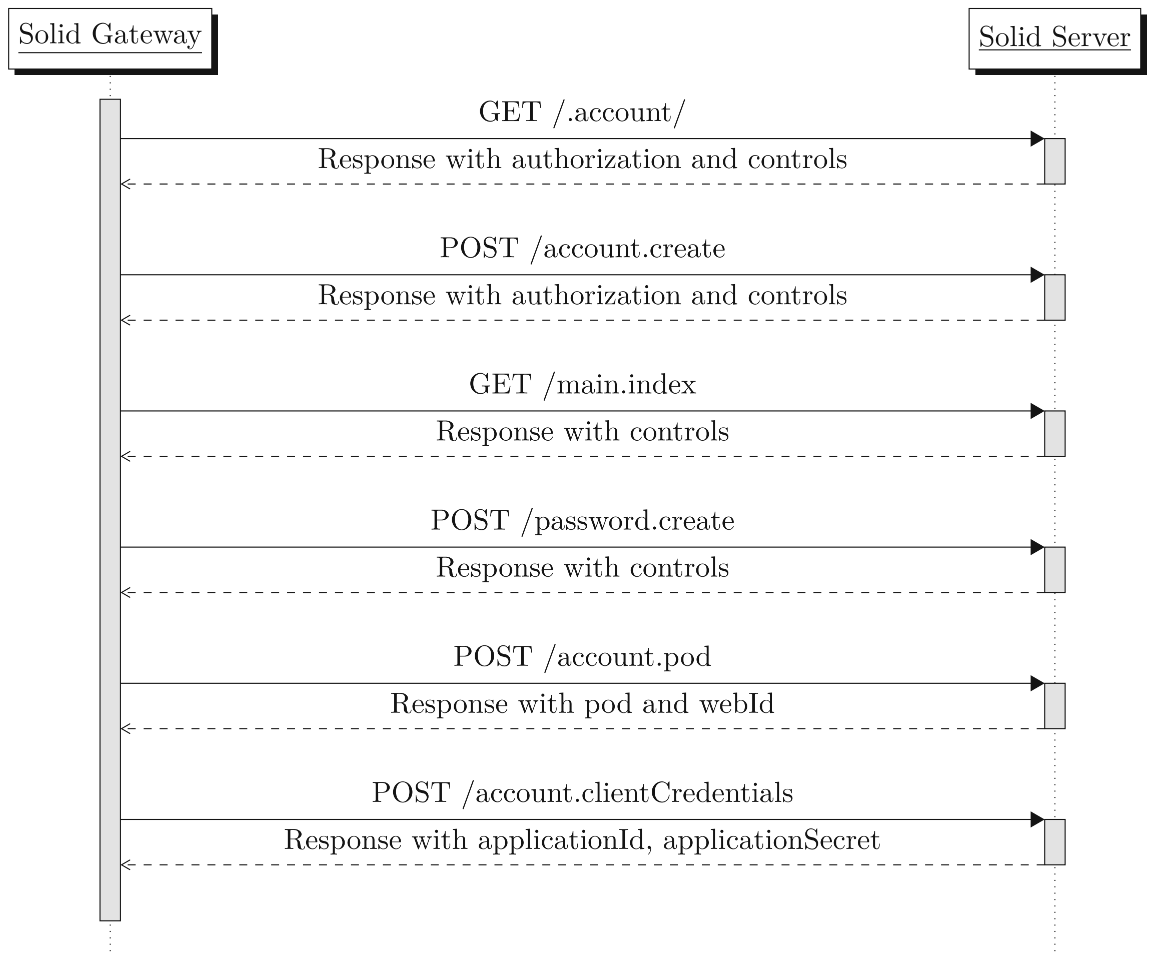
Solid Gateway
Solid Server
GET /.account/
Response with authorization and controls
POST /account.create
Response with authorization and controls
GET /main.index
Response with controls
POST /password.create
Response with controls
POST /account.pod
Response with pod and webId
POST /account.clientCredentials
Response with applicationId, applicationSecret

Figure 4.13: Creating a WebID and Client Application

Table 4.1: API Endpoints of the SolidGateway and their details

#	**Endpoint**	**Method**	**Authentication**	**Purpose**	**Response**
E1	/int/request	POST	None	New Request	id and secret to use to authenticate for this request against the gateway server
E2	/int/participants	GET	id and secret	Request Status	checks each participants pod for the authenticated request and gives back the number of accepts
E3	/int/data	GET	id and secret	Request Data	fetches data of each participant for the authenticated request and gives back the data if the threshold is met
E4	/int/health	GET	None	Health	status of the gateway, either healthy or not healthy
E5	/ext/join	GET	None	Participate	adds a WebID to the list of participants

4.6 Summary

In this chapter, the conceptual design of integrating *Solid* as a decentralized data storage solution into existing *Data Platforms* has been thoroughly explored and detailed. This prototype implementation demonstrates a shift from traditional centralized data repositories to a decentralized model. By adapting and extending the frameworks proposed by Sun et al. [SOSD23] and Ekaputra et al. [EEM$^+$21], our proposed changes enhance the overall data sovereignty for *Data Contributors*.

Central to this conceptual design is the introduction of the *Solid Gateway*, an component that bridges external *Solid Pods* with the *Data Platform*. This gateway facilitates the retrieval and management of data from *Solid Pods* with request specific WebIDs. In addition, we introduce the *Request Portal*, a *Solid* application for *Data Contributors* to accept requests and manage consent. We propose the *Granular* data sharing approach, which focuses on *data minimization* compared to existing approaches [SOSD23]. The detailed flowcharts and sequence diagrams provided within the chapter effectively illustrate the operational flow and interactions, making it clear how data requests are processed and managed in a manner that respects user consent and minimizes data exposure.

To strengthen transparency and traceability, the *Solid Gateway* implements selected aspects of the FAIR principles and the RDA Dynamic Data Citation Recommendations from Rauber et al. [RAUP16] such as basic provenance tracking and integrity verification.

In conclusion, we provide a comprehensive blueprint for integrating *Solid Pods* into *Data Platforms*, addressing key issues such as data sovereignty, sharing mechanisms, and provenance. The next steps, following this conceptual foundational work, will involve testing and evaluation of the system to ensure it meets the requirements stated in Sections 3.3 and 3.4.

CHAPTER 5

Evaluation and Result

5.1 Implementation

For the evaluation of both functional and nonfunctional requirements, we have artificially created 500 *Data Contributors*; each of them has their own unique *Solid pod*, data and *WebID*. We deploy our own *Solid Server*, which also acts as an *Identity Provider*.

5.1.1 Data

Datasets are structured to represent health data using the RDF format, where each piece of information is expressed as a triple component: subject, predicate, and object. The TTL files are structured according to the FHIR v5.0.0 standard and are compatible with the RDF model used by Solid. The primary structure of each file includes:

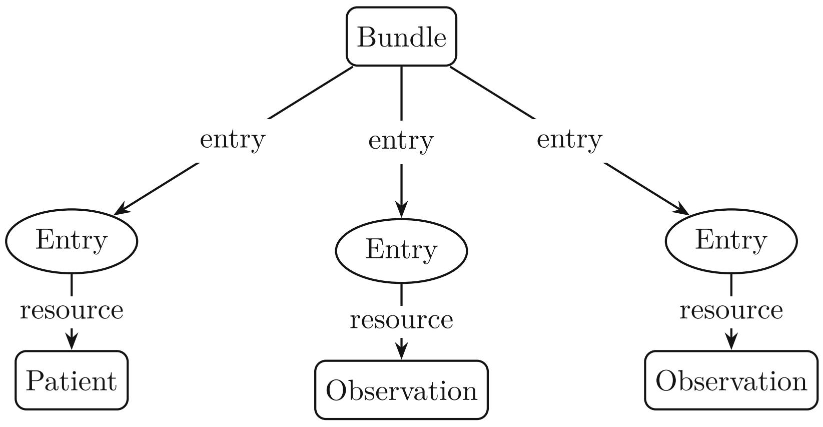

Figure 5.1: FHIR Structure of a Data Set

- **Bundle as Root**: Each data file starts with a *Bundle* resource as the root, encapsulating all other resources related to a data contributor as seen in Figure 5.1.
- **Entries for Resources**: Each resource (*Patient*, *Observation*) is included as an entry in the *Bundle*. These entries are linked via fhir:entry relations pointing to individual resources defined elsewhere in the TTL file. This is the second layer, as seen in Figure 5.1.
- **Resource Details**: The last layer in Figure 5.1 describes each resource, such as *Patient* or *Observation*. Both are detailed through a series of triples that define its attributes and relationships. For *Observations* we make use of the fhir:issued, fhir:code and fhir:valueQuantity relationships to obtain the time of recording, the *Coding* resource where we match the code like "BUN" or "CREAT" as well as the actual recorded value inside via the *Quantity* resource. For *Patients* we use fhir:birthDate, fhir:gender, fhir:language, and fhir:country.

We generate multiple datasets tailored for various experimental setups, including a realistic dataset designed to simulate the data of 500 Data Contributors. Each Data Contributor is represented by a complete patient profile that contains the attributes birth date, gender, language, and country information. The number of observations per contributor varies between 2 and 52 entries, with a median of 22 entries per dataset. In particular, TCH and HDL are the observations that occur the most frequently within the dataset, as illustrated in Figure 5.2. We will refer to this dataset as REAL500. It serves as a representative model for experiments that require realistic variability in the data while maintaining structured profiles. The datasets used range from 3 to 125 kilobytes in size, all 500 datasets have 23.2 megabytes combined.

We also introduce PERF500, a synthetic dataset in which each data contributor is assigned an identical dataset comprising a complete patient profile. Each profile includes attributes such as birth date, gender, language, country, and 26 observations. This dataset is designed to provide consistent and uniform test data, enabling controlled scalability when increasing the number of datasets used in experiments. The uniformity ensures that all Data Contributors can fulfill any data request, eliminating the risk of excluding contributors due to data variability. For example, when a subset of 100 contributors is used from the 500 available, the dataset is referred to as PERF100, guaranteeing the inclusion of all selected contributors in the experiment. Although this approach sacrifices a degree of realism, it enhances the controllability and simplicity of the experimental setup, enabling more precise evaluation and reproducibility. The complete dataset, consisting of 500 files, is 36.4 megabytes in size. A single file is exactly 72.8 kilobytes.

To automate experimental setups, including tasks such as uploading data to individual pods, managing data request acceptances, and generating scenarios for test cases, we developed a custom repository named solid-tools[1]. This repository comprises a collection

[1]`https://github.com/kooperativ97/solid-tools`

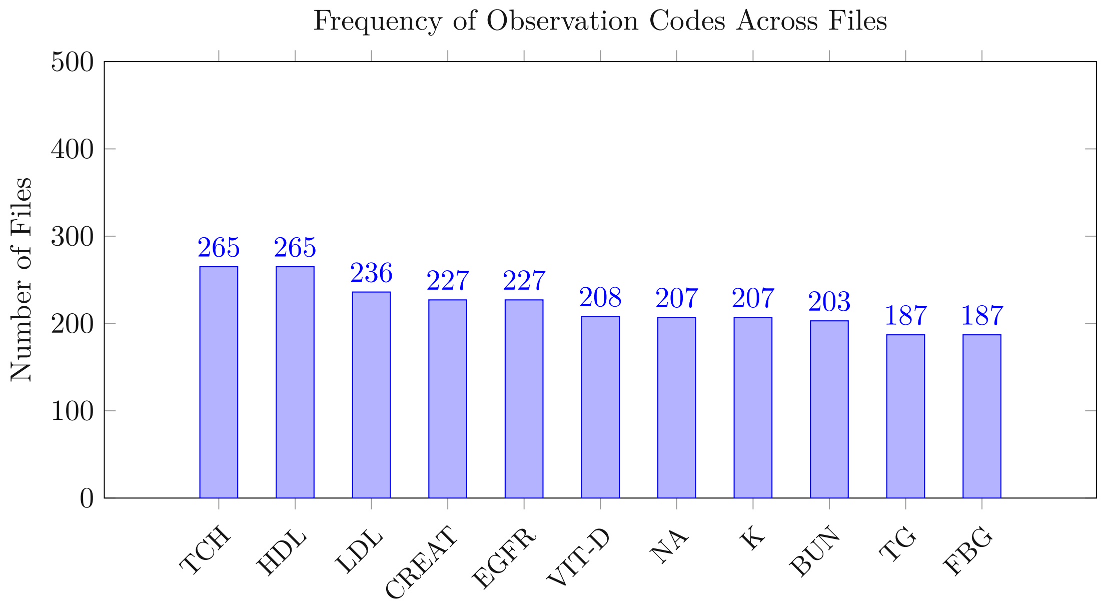

Figure 5.2: Frequency of Top 10 Observation Codes

of JavaScript utilities designed to be executed with NodeJS, alongside a set of predefined test data. We use these scripts to reduce manual intervention and ensure consistency between our different experiment scenarios.

5.1.2 Solid Server

In all tests, we work with the CommunitySolidServer[2], an open and modular implementation of the Solid specifications[3] via docker using the solidproject/community-server:7.1[4] image. We chose the *file.json* configuration, which saves all data inside pods as plain files on the *Solid Server* and uses Web Access Control (WAC)[5] to manage data sharing. The server is published through *nginx* with a *letsencrypt*[6] certificate for the domain verysolid.de. The virtual server has 2 gigabytes of RAM, 2 virtual CPU cores, and a data connection of 1000 Mbit/s.

[2]https://github.com/CommunitySolidServer/CommunitySolidServer
[3]https://solidproject.org/TR/
[4]https://hub.docker.com/r/solidproject/community-server
[5]https://solid.github.io/web-access-control-spec/
[6]https://letsencrypt.org/

5.2 Functional Evaluation

In this section, we demonstrate the fulfillment of the functional requirements outlined in Table 3.1 through our proposed system. For each specified requirement, we have developed one or more corresponding test cases. Each case includes a name, a description, the expected and actual results, and a pass/fail status.

5.2.1 FR1 - Data Query & Analysis

This functional requirement is the most important one, as it shows the integration of *Solid Pods* into a data platform from start to finish. We have defined three test cases for this requirement.

Table 5.1: Test Data Request Creation

Test name	Data Request Creation
Description	Data Consumer logs into the *Analysis Interface*, navigates to the data request creation page, fills out and submits the data request form. The request data is passed on to the Solid Gateway and processed. In this process, the gateway creates a new WebID, and generates a TTL file that is uploaded to the *DP Pod* with public access settings. The *Analysis Interface* receives a key and secret from the gateway to later authorize requests against the gateway.
Expected result	A TTL file is created and stored in the *DP Pod* which contains the request information and correctly references the new WebID created for the request. The gateway returns a key and secret to the *Analysis Interface* which authenticate later requests against the gateway.
Actual result	The request file as seen in Listing 5.3 was uploaded to the *DP Pod*, contains all relevant information such as requested observations and patient data, as well as a description and purpose and is publicly available under `https://verysolid.de/WellFort3-pod/requests/switchboard_and_adiuvo.ttl`. The credentials where submitted in the response of the *Solid Gateway* as seen in Listing 5.2.
Test passed	True

Table 5.1 shows the first test case for FR1. In a first step, a user 'tobias.hajszan@research.tuwien.ac.at' was created with the *Experiment Setup Interface*. Then a request was created that requests the gender, birth date, and country of a patient. In addition, observations of blood urea nitrogen "BUN" and creatinine "CREAT" were requested. This represents the first step, as shown in the sequence diagram in Listing 4.3. Upon submission of the request, a POST request is issued to the *Solid Gateway*. The

```json
{
    "email": "tobias.hajszan@research.tuwien.ac.at",
    "data": {
        "consent": {
            "purpose": "ResearchAndDevelopment",
            "description": "We want to find the cure",
            "processing": "Analyse",
            "duration": "2025-01-01"
        },
        "attributes": {
            "patient": ["Gender", "BirthDate", "Country"],
            "observation": ["BUN", "CREAT"]
}}}
```

Listing 5.1: Request to Solid Gateway to create a new Request

```json
{
    "email": "tobias.hajszan@research.tuwien.ac.at",
    "request-id": "switchboard_and_adiuvo",
    "request-secret": "O24KjWyZmG1QkudYVEZmAXszWLX6165g"
}
```

Listing 5.2: Response from the Solid Gateway

endpoint used is E1 as listed in Table 4.1. The request contains a json body with the content shown in the Listing 5.1. Based on the data submitted, a corresponding request file is generated in RDF Turtle format and saved to the *WellFort* data pod to /requests/switchboard_and_adiuvo-pod.ttl.

The name of each request is randomly generated and made up of three random words. This pattern allowed for easier development and testing and should be replaced with a UUID in the future.

The *Solid Gateway* answers the response with HTTP 200 ok and a json body containing a key and secret, as seen in Listing 5.2. This key and secret are randomly generated by the *Solid Gateway* and allow the *Analysis Interface* to authenticate against the *Solid Gateway* to check the status of a request. These credentials are not the credentials for the WebID that is created, only for the *Solid Gateway* to access endpoints E2 and E3 as seen in Table 4.1.

The key and secret are saved by the *Analysis Interface* and linked to the logged-in user. The *Data Consumer* then sees a new entry on his request overview page where he can interact with the data request. At the same time, *Data Contributors* can now see the request in the *Request Portal* and can choose to accept it. By not accepting the data request, it is implicitly denied as no permissions are set for the *Solid Gateway* to retrieve any data.

```
@prefix fhir: <http://hl7.org/fhir/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .
@prefix vs: <http://verysolid.de/ns#> .

<https://verysolid.de/wellfort-pod/requests/switchboard_and_adiuvo>
    a dpv:PersonalDataHandling;
    dpv:DataController <https://verysolid.de/switchboard_and_adiuvo-pod/
    profile/card#me>;
    schema:email "tobi@verysolid.de";
    dpv:hasPurpose dpv:ResearchAndDevelopment;
    dpv:hasProcessing dpv:Analyse;
    dpv:hasImpact "We want to find the cure, description of the impact";
    dcterms:description "";
    dpv:hasExpiryTime "2025-08-01"^^xsd:dateTime;
    schema:dateCreated "2025-01-01"^^xsd:dateTime;
    vs:requestedObservations "BUN", "CREAT" ;
    vs:requiresPatientData "Gender", "BirthDate", "Country" .
```

Listing 5.3: Data Request created by Solid Gateway

```
<https://verysolid.de/contributor-pod/.acl#owner> a <http://www.w3.org/ns/
    auth/acl#Authorization>;
    <http://www.w3.org/ns/auth/acl#agent> <https://verysolid.de/contributor-
    pod/profile/card#me>, <mailto:undefined>;
    <http://www.w3.org/ns/auth/acl#mode> <http://www.w3.org/ns/auth/acl#Read
    >, <http://www.w3.org/ns/auth/acl#Write>, <http://www.w3.org/ns/auth/acl#
    Control>;
    <http://www.w3.org/ns/auth/acl#accessTo> <https://verysolid.de/
    contributor-pod/health/dataFhir.ttl>;
    <http://www.w3.org/ns/auth/acl#default> <https://verysolid.de/contributor
    -pod/health/dataFhir.ttl>.
<#78f4e9cb-47ef-41ac-b14b-73410b10b022> a <http://www.w3.org/ns/auth/acl#
    Authorization>;
    <http://www.w3.org/ns/auth/acl#agent> <https://verysolid.de/
    switchboard_and_adiuvo-pod/profile/card#me>;
    <http://www.w3.org/ns/auth/acl#mode> <http://www.w3.org/ns/auth/acl#Read
    >;
    <http://www.w3.org/ns/auth/acl#default> <https://verysolid.de/contributor
    -pod/health/dataFhir.ttl>.
```

Listing 5.4: ACL file in solid with read access for webid

Table 5.2 shows the second test for this functional requirement, the proactive choice of *Data Contributors*.

The procedure described in Table 5.2 holds for any *Data Consumer* that accepts the given request, given at least one data point for each requested observation, and all requested patient information can be extracted from the data of the *Data Owner*, which is checked

Table 5.2: Test Data Contributor Accepts Request

Test name	*Data Contributor* Accepts Request
Description	A *Data Contributor* logs into the *Request Portal*, reviews the requests and accepts a data request. Upon acceptance, the system updates the ACL settings in the *Data Contributor's Solid Pod* to grant the specified WebID access to the requested data.
Expected result	The ACL file in the *Data Contributor's Pod* is updated to reflect the new access rules, allowing the specified WebID read access to the requested data. This update should correctly reference the WebID provided by the data request, ensuring the request is authorized to access only the agreed-upon data. A consent file is created in their personal pod in the folder /requests/accepted/.
Actual result	The access rules were updated in the *Data Contributor's Pod* located at `https://verysolid.de/contributor-pod/data/dataFhir.ttl.acl`, providing read access to the specified data as seen in the ACL file in Listing 5.4. The WebID given access matches the one generated for the request, as intended. Also, the consent file for the data request `https://verysolid.de/WellFort3-pod/requests/switchboard_and_adiuvo.ttl` was saved into `https://verysolid.de/contributor-pod/requests/accepted/switchboard_and_adiuvo.ttl` with read access to `https://verysolid.de/switchboard_and_adiuvo-pod/profile/card#me`.
Test passed	True

in the Accept routine of the user application. In the *Standard* approach, which is oriented toward the implementation of Sun et. al. [SOSD23], the requesting WebId is authorized to access the whole data file.

The last step of the functional requirement is tested with the test described in Table 5.3.

We setup 100 *Solid Pods* that have accepted the request *switchboard_and_adiuvo*. We used JavaScript to create automated processes that were executed with NodeJS. The data files used are available in the data folder[7] of the repository and contain FHIR test data, the automations can be found as javascript files in the root of the repository[8]. For each pod a unique data file was uploaded containing the patient's gender, age, language, and country, as well as the requested observations; blood urea nitrogen "BUN" and creatinine

[7]`https://github.com/kooperativ97/solid-tools/tree/main/data`
[8]`https://github.com/kooperativ97/solid-tools/`

Table 5.3: Test Case: Data Consumer Queries Data

Test name	*Data Consumer* queries data
Description	A *Data Consumer* checks the status of a created request within the *Analysis Interface* and it shows, that the threshold of accepts was surpassed, enabling the setup of a isolated, temporary environment. The *Data Consumer* is given credentials to access an R environment, secured by DataSHIELD.
Expected result	The *Analysis Interface* sends a request the *Solid Gateway* which returns the requested data in case the minimum accepted requests threshold is surpassed. Then, the *Analysis Interface* processes the data the same way, as it would have been when the data was sourced from the data repository used in the original proposed architecture by [EEM$^+$21].
Actual result	An opal project folder has been created based on the data received from the *Solid Gateway* by the *Analysis Interface*, successfully replacing the original centralized data repository from the architecture by the *Solid Gateway*.
Test passed	True

"CREAT". The access rights have been set with the creation of ACL files for the relevant files. In addition, we added each WebID to the participant list of the solid gateway so that the pods can be queried. This was done by inserting each WebID in the participant table, as seen in Figure 4.12.

The user tobias.hajszan@tuwien.ac.at, which created the request, now initiates the data request through the *Analysis Interface*, as visualized in Figure 4.9. A request with HTTP header request-id and request-secret is sent to `E3` as listed in Table 4.1. The data platform will, based on the provided headers, authenticate as the request WebID and start contacting each participant to check for the accepted request. Each pod location is queried with an HTTPS GET at `https://verysolid.de/contributor-pod/requests/accepted/switchboard_and_adiuvo.ttl`. If this request is successful, the gateway will try to fetch the data, which is located at `https://verysolid.de/contributor-pod/health/dataFhir.ttl`.

In a next step, the gateway will extract only the relevant data from the received RDF data and pass a json structure with all the data collected to the *Analysis Interface* in the form of a json response as described by the JSON Schema in the Listing 5.6. Listing 5.5 shows an example of the json.

The *Analysis Interface* then creates a unique folder, creates the necessary files for opal and creates a project through opal's REST-API.

Given that the three test cases, seen in Table 5.1, Table 5.2 and Table 5.3 have passed, we show that the given functional requirement is fulfilled by our proposed framework.

```
[{
        'id': 'http://medicus.ai/fhir/Patient/0',
        'country': 'CY',
        'birthDate': '1959-01-01',
        'gender': 'female',
        'observations': [{
                'value': 8.09,
                'biomarker': 'Blood Urea Nitrogen',
                'biomarker_code': 'BUN',
                'issued': '2019-06-21T07:53:40Z'
            }, {
                'value': 7.85,
                'biomarker': 'Blood Urea Nitrogen',
                'biomarker_code': 'BUN',
                'issued': '2019-02-07T22:54:03Z'
            }, {
                'value': 82.61,
                'biomarker': 'Creatinine',
                'biomarker_code': 'CREAT',
                'issued': '2019-08-08T04:49:25Z'
}]}]
```

Listing 5.5: Response from the Solid Gateway

5.2.2 FR2 - Privacy-Preserving Analytics

Building on [EEM+21] while retaining its core privacy-preserving mechanisms, we adapted the workflow of the infrastructure to fetch data from our proposed *Solid Gateway* instead of the *Secure Repository*. The infrastructure continues to ensure that all computations on sensitive personal data occur within a controlled and secure environment. *Data Consumers* interact only with aggregated results, never directly accessing raw datasets. This ensures that the privacy of individual *Data Contributors* remains intact. Using the *Analysis Interface*, we provide *Data Consumers* with credentials to temporary R environments that are secured using DataSHIELD[9].

To further enhance privacy, we implemented a *Granular* approach that minimizes data exposure by reducing the absolute number of RDF triples transferred, as evaluated in Section 5.4. We tested this requirement using the test case presented in Table 5.4.

[9]https://datashield.org/

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "id": {
        "type": "string",
        "format": "uri"
      },
      "country": {
        "type": "string"
      },
      "birthDate": {
        "type": "string",
        "format": "date"
      },
      "gender": {
        "type": "string"
      },
      "language": {
        "type": "string"
      },
      "observations": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "value": {
              "type": "number"
            },
            "biomarker": {
              "type": "string"
            },
            "biomarker_code": {
              "type": "string"
            },
            "issued": {
              "type": "string",
              "format": "date-time"
            }
          },
          "required": ["value", "biomarker", "biomarker_code", "issued"]
        }
      }
    },
    "required": ["id", "observations"]
  }
}
```

Listing 5.6: Schema for the JSON response from Solid Gateway

Table 5.4: Test Case: Privacy-Preserving Isolated Environment

Test name	Privacy-Preserving Isolated Environment
Description	A *Data Consumer* triggers the fetching of data from the data request through the *Analysis Interface*. The *Solid Gateway* retrieves data from the *Data Contributor's Solid Pods* and provides it to the *Analysis Server* which creates a temporary folder to import the data as project into Opal.
Expected result	The *Trusted Analysis Environment* receives the data in the same way, as it would have been from the *Secure Repository*. Opal imports the received data upon retrieval, therefore providing access via DataSHIELD.
Actual result	A folder containing the requested data is created based on the *Solid Gateways* response. Opal successfully imports the data as a project and can therefore be used as a data source for DataSHIELD.
Test passed	True

5.2.3 FR3 - Provenance Logging

To support accountability and transparency in data processing, our objective was to comply with the RDA Dynamic Data Citation Recommendations by Rauber et al. [RAUP16] by implementing a provenance tracking mechanism directly within the *Solid Gateway*, as described in Section 4.5. The *Solid Gateway* maintains a structured relational database that records relevant event data related to data requests and retrievals. This includes logging of timestamps for each request and fetch operation (R7), the storage of cryptographic hashes to verify the integrity of retrieved data (R6), and metadata on the *Solid Pods* involved.

This test case as described in Table 5.5 verifies that the provenance tracking mechanism in the *Solid Gateway* accurately records changes in data over time. We re-execute the same query on the same data request at two different times. In our test, the first execution captures the initial state of the data. Before the second execution, a data value in one of the *Data Contributor's Solid Pod* is modified. The subsequent query fetch returns a different hash value for the same URL. We use an SQL query[10], which compares the hash values of the fetch operations for the same URL from different executions to identify inconsistencies.

5.2.4 FR4 - Study-Specific Sharing

We have implemented a safeguard that inspects the *Data Contributors'* data before creating access conditions for the WebID requesting the data. In our proposed *Request*

[10]https://github.com/kooperativ97/solid-gateway/blob/main/files/compare_query_hashes.sql

Table 5.5: Test Case: Provenance Logging

Test name	Provenance Logging
Description	A *Data Consumer* triggers data fetches for the same data request two separate times. Between the first and the second fetch, the data in one of the *Data Contributors Pods* changes.
Expected result	We expect, that all requests from the *Solid Gateway* against the pods return the same data, thus same hash value logged. One pod will yield different data and therefore have a different hash value for the same url. We built an SQL query to find hash inconsistencies between urls of two data fetches.
Actual result	The SQL query returned the inconsistent fetch correctly.
Test passed	True

Portal that runs directly in the browser, we have implemented this safeguard in the routine of accepting a data request, which is tested with the test case presented in Table 5.6.

Table 5.6: Test Case: Sharing of Unavailable Data

Test name	Sharing of Unavailable Data
Description	A *Data Contributor* tries to accept a data request that requests observations that are not in the data of the *Data Contributors* pod.
Expected result	The *Request Portal* detects that some of the requested data cannot be provided as it is not found in the pod of the *Data Contributor*. An error message is shown, the accept fails and no access rules are created. The data file of the *Data Contributor* can be found here[11]. The request is available here[12]
Actual result	The accept routine returned the error "Cannot accept request. Missing data for the following observation codes: BUN"
Test passed	True

5.2.5 FR5 - Data Minimization Transfer

With our proposed *Granular* approach which is implemented in the *Request Portal*, only relevant information, which is relevant to the data request, is shared with the *Solid Gateway*. For this test case in Table 5.7, we again used the same request as in FR1 seen in the Listing 5.3.

[11]`https://github.com/kooperativ97/solid-gateway/blob/main/files/tests/fhir_data_fr4.ttl`

[12]`https://github.com/kooperativ97/solid-gateway/blob/main/files/tests/data_request_fr4.ttl`

Table 5.7: Test Case: Data Minimization in Data Sharing

Test name	*Data Minimization* in Data Sharing
Description	A *Data Contributor* accepts a data request which asks to share general information as well as two types of observations. The accept leads to the extraction of those information from the data and sharing of the created subsets.
Expected result	The *Request Portal* evaluates the request and detects that all of the requested data is available in the *Data Contributor's Pod*. The *Solid* application starts to extract the information from the main data file[13] and creates a new file with only the requested data inside. The newly created data file[14] of the *Data Contributor* only contains the data requested in the request file[15].
Actual result	The newly created data file only contains the requested patient information as well as only the requested observations.
Test passed	True

5.2.6 FR6 - Interoperability Standards

The proposed architecture consistently applies standardized data formats at each stage of the data processing pipeline. Personal data stored within *Solid Pods* are saved in RDF Turtle, a widely adopted serialization format for Linked Data. In this work, we use the FHIR ontology for healthcare data and the DPV ontology for consent management. Similarly, data requests issued by *Data Contributors* and published by the *Solid Gateway* are also represented as RDF Turtle documents following the DPV ontology, as demonstrated in Listing 5.3.

The *Solid Gateway* itself provides JSON-based HTTP endpoints, ensuring simple and widely compatible machine-to-machine communication to manage requests and retrieve data.

Finally, results exported through the *Analysis Interface* are made available in CSV format, a common standard for tabular data that is used in statistical tools, programming languages, spreadsheet applications, and external analysis environments.

5.2.7 FR7 - Sharing Transparency

When consent is given to a data request by the *Data Contributor*, a new folder is created within the *Data Contributor's Pod* and the relevant data for this request are extracted and

[13] `https://github.com/kooperativ97/solid-gateway/blob/main/files/tests/fhir_data_fr5.ttl`

[14] `https://github.com/kooperativ97/solid-gateway/blob/main/files/tests/fhir_data_fr5_granular.ttl`

[15] `https://github.com/kooperativ97/solid-gateway/blob/main/files/tests/fhir_data_fr5_request.ttl`

put into a new data file. This file resides inside the *Solid Pod* and is shared by creating an ACL file that references the request WebID as `agent`[16] which is allowed to read the created data file. In addition, a consent file is also saved within the *Data Contributor's Pod*, which references the data request as seen in Figure 5.7. All information created by the process of accepting a data request stays in the control of the *Data Contributor*, since each access control file states that the *Data Contributor* holds all rights to each file as seen in Figure 5.4. We tested this requirement using the test case presented in Table 5.8.

Table 5.8: Test *Data Contributor* Accepts Request

Test name	*Data Contributor* Gives Consent
Description	A *Data Contributor* logs into the *Request Portal*, reviews the open requests and accepts a data request. The system updates the ACL settings in the *Data Contributor's Solid Pod* to grant access for the specified WebID to the requested data and creates a consent file that references the request file.
Expected result	A consent file should be created at `https://verysolid.de/contributor-pod/requests/accepted/switchboard_and_adiuvo/` together with an ACL file that gives read access to the agent with WebID `https://verysolid.de/switchboard_and_adiuvo-pod/profile/card#me`. The consent file has to reference the data request file `https://verysolid.de/wellfort-pod/requests/switchboard_and_adiuvo`
Actual result	The consent file is created, authorized to the agent `https://verysolid.de/switchboard_and_adiuvo-pod/profile/card#me` and references `https://verysolid.de/wellfort-pod/requests/switchboard_and_adiuvo` as `dpv:hasConsentNotice`.
Test passed	True

5.2.8 FR8 - Consent Withdrawal Enforcement

When consent is given to a data request, a new folder is created, and the relevant data for this request are extracted and put into a new data file. In addition, a consent file, as seen in Listing 5.7 is created based on the data request. For both files, an ACL file is created that gives read permissions to the WebID that requests the data. When withdrawing consent through the *Request Portal*, ACL authorizing read access to the data file is simply removed, thereby removing read permission to the *Data Contributor's* data. Further, the

[16] `http://www.w3.org/ns/auth/acl#agent`

```
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .

<https://verysolid.de/contributor-pod/requests/accepted/
    switchboard_and_adiuvo>  a dpv:Consent;
    schema:dateCreated "2025-01-01"^^xsd:dateTime;
    dpv:DataSubject <https://verysolid.de/contributor-pod/profile/card#me>;
    dpv:hasConsentNotice <https://verysolid.de/wellfort-pod/requests/
    switchboard_and_adiuvo>;
    dpv:hasExpiryTime "2025-08-01"^^xsd:dateTime.
```

Listing 5.7: Consent File created by the Request Portal

Request Portal updates the consent file by appending the `dpv:hasWithdrawalTime` with the current timestamp. As *Solid Pods* can be completely controlled by the owner of the pod, we can not enforce such behavior, as the user might use another service to manage data inside the pod. Therefore, if the consent file becomes inaccessible, indicated by a HTTP 403 Forbidden or 404 Not Found error, we will also treat it as a withdrawal of consent. We tested this requirement using the test case presented in Table 5.9.

Table 5.9: Test *Data Contributor* Withdraws Consent

Test name	*Data Contributor* Withdraws Consent
Description	A *Data Contributor* logs into the *Request Portal*, reviews the accepted requests and withdraws the consent. The system updates the ACL settings in the contributor's *Solid Pod* to withdraw access for the specified WebID to the requested data.
Expected result	The for the request created ACL file is deleted from the *Data Contributor's Pod*, thus access is no longer possible. The consent file is updated by appending the `dpv:hasWithdrawalTime` with the current timestamp.
Actual result	The consent file is updated according to Listing 5.8, the ACL file for the data is deleted.
Test passed	True

```
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dpv: <http://www.w3.org/ns/dpv#> .

<https://verysolid.de/contributor-pod/requests/accepted/
    switchboard_and_adiuvo>  a dpv:Consent;
    schema:dateCreated "2025-01-01"^^xsd:dateTime;
        dpv:hasWithdrawlTime "2025-02-01"^^xsd:dateTime;
    dpv:DataSubject <https://verysolid.de/contributor-pod/profile/card#me>;
    dpv:hasConsentNotice <https://verysolid.de/wellfort-pod/requests/
    switchboard_and_adiuvo>;
    dpv:hasExpiryTime "2025-08-01"^^xsd:dateTime.
```

Listing 5.8: Consent File after Consent is Withdrawn via the Request Portal

5.3 Performance Analysis

This section analyzes performance differences in fetching and processing data using *Standard* and *Granular* strategies. Performance is evaluated on the basis of the time taken to fetch data from *Solid Pods* and process them locally. The objective is to determine which strategy offers optimal efficiency in operational contexts.

5.3.1 Methodology

Performance analysis focuses on measuring the time required to fetch and process data using two distinct strategies:

- **Standard Strategy:** Fetches the complete data file.
- **Granular Strategy:** Fetches a data file that is already filtered using the *Granular* approach to only contain the data required by the data request.

Experiments are conducted with different combinations of parameters to ensure a complete evaluation.

- **Strategies:** *Standard* and *Granular*.
- **Number of Data Contributors:** 100, 200, 300, 400 and 500.
- **Datasets:** REAL and PERF.

For each combination of strategy, number of *Data Contributors*, and dataset, the fetching process via the *Solid Gateway* is triggered 10 times to mitigate any anomalies due to network latency or server performance problems. Subsequently, the results are averaged to obtain a more reliable measure of the fetching and processing times. We visualize the exact points of measurement in Figure 5.3.

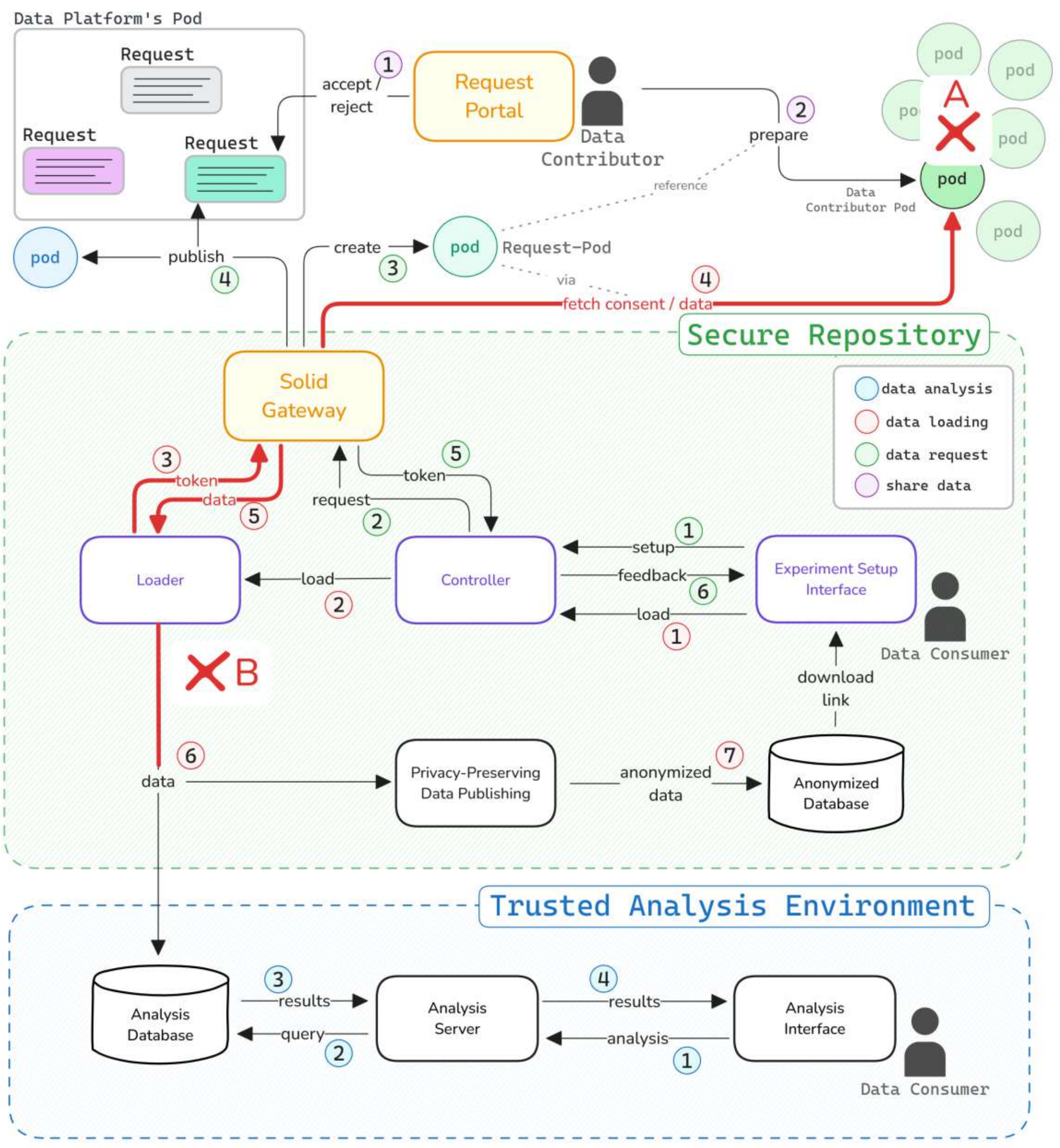

Figure 5.3: Performance evaluation for our proposed Solution

We measure fetching performance from point *A* until the data is received and processed by the *Loader*, denoted by point *B*.

Data for these experiments are stored in RDF format within *Solid Pods*. The fetch and process times are recorded, allowing for a direct comparison between the strategies. We calculate for each combination three metrics.

- **Total Fetch Time:** The time taken from initiating the fetch requests from the *Solid Gateway* until all data have been received from the *Solid Pods*.
- **Total Process Time:** The time required to process the received data, such as filtering out only relevant parts of the data.
- **Number of Requests:** The number of HTTP requests from the *Solid Gateway* against the pods of the *Data Contributors*.

Consent is verified by the *Solid Gateway* for each *Data Contributor* before trying to fetch data from the pod. The results of this experiment are presented in Table 5.10 and show the average fetch and process times, as well as the number of requests, for the *Standard* and *Granular* strategies in the datasets and the varying numbers of *Data Contributors*.

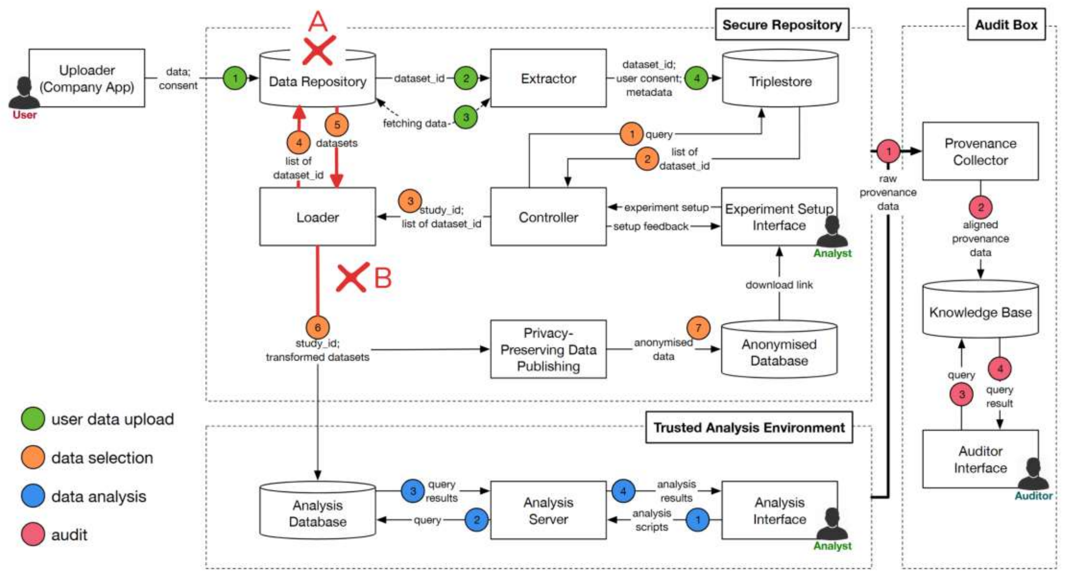

Figure 5.4: Performance evaluation for *WellFort*

Further, we compare both strategies to *WellFort* to analyze the impact of replacing the local repository with our proposed *Solid Gateway*. We therefore set up the repository with the REAL and PERF datasets in JSON format. We include performance metrics in

Table 5.10: Average Fetch and Process Times for Standard and Granular Strategies

Dataset	n	Strategy	Fetch Time (ms)	Process Time (ms)	Requests
PERF	100	Standard	9114.84	2663.88	200
PERF	100	Granular	6619.59	1094.41	200
PERF	100	WellFort	5238.10	29376.14	200
PERF	200	Standard	14427.34	5562.09	400
PERF	200	Granular	13028.65	2107.20	400
PERF	200	WellFort	10185.68	55859.11	400
PERF	300	Standard	20876.19	7910.25	600
PERF	300	Granular	19081.23	3055.63	600
PERF	300	WellFort	14987.45	81451.45	600
PERF	400	Standard	26310.83	10413.48	800
PERF	400	Granular	24579.80	4077.69	800
PERF	400	WellFort	20120.70	109056.16	800
PERF	500	Standard	34804.93	11646.90	1000
PERF	500	Granular	31186.70	5055.21	1000
PERF	500	WellFort	25963.35	141788.81	1000
REAL	100	Standard	6056.07	1428.16	141
REAL	100	Granular	4821.22	649.95	141
REAL	100	WellFort	2073.06	10459.24	80
REAL	200	Standard	9559.74	2404.30	279
REAL	200	Granular	8413.20	1286.75	279
REAL	200	WellFort	3944.75	18803.51	154
REAL	300	Standard	13519.03	3634.75	419
REAL	300	Granular	12350.11	1856.25	419
REAL	300	WellFort	5914.63	28652.83	236
REAL	400	Standard	17039.32	4547.29	553
REAL	400	Granular	16027.04	2448.80	553
REAL	400	WellFort	7496.11	37236.88	302
REAL	500	Standard	27212.34	5773.50	682
REAL	500	Granular	20332.15	2957.45	682
REAL	500	WellFort	9626.40	47449.35	364

Table 5.10 and show our detailed results in Subsection 5.3.4. We again measure from point A to B as shown in Figure 5.4.

In addition to comparing fetching strategies, we conduct an evaluation to analyze the effect of parallelizing both fetching and processing on overall performance in Subsection 5.3.5. The objective of this analysis is to determine how increasing the number of parallel workers impacts the total time required to retrieve and process the data in the *Solid Gateway*. We run the experiments only for our largest datasets *PERF500* and *REAL500* and increase the number of workers from one to eight. Again, for each number of workers, we run the query process 10 times to ensure stability and mitigate anomalies in network or processing performance.

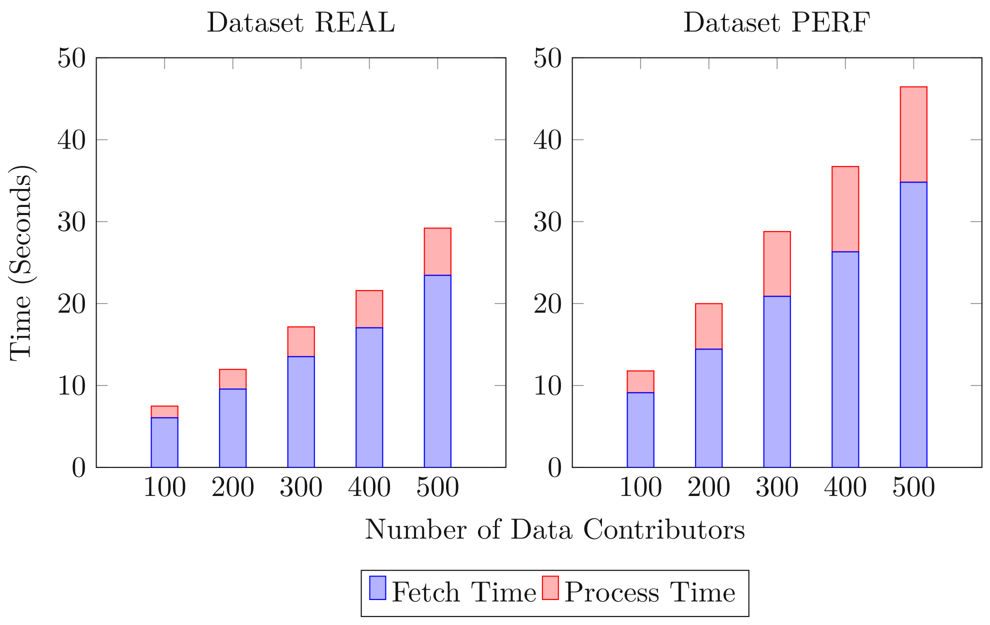

Figure 5.5: Comparison of fetching and processing times by number of *Data Contributors* for the REAL and PERF datasets, using the *Standard* strategy.

5.3.2 Standard Approach

The *Standard* approach fetches the entire data of a *Data Contributor* in one request, regardless of the query specifics. This approach is similar to the approach used by Sun et al. in [SOSD23] where the filtering also happens inside the temporary analysis environment.

- **Fetch Time:** The fetch time for the *Standard* approach are shown in Figure 5.5. Because the entire data of each *Data Contributor* is retrieved in one request, the fetch time is mainly dependent on the network speed and the data volume at each *Data Contributor's Pod*.

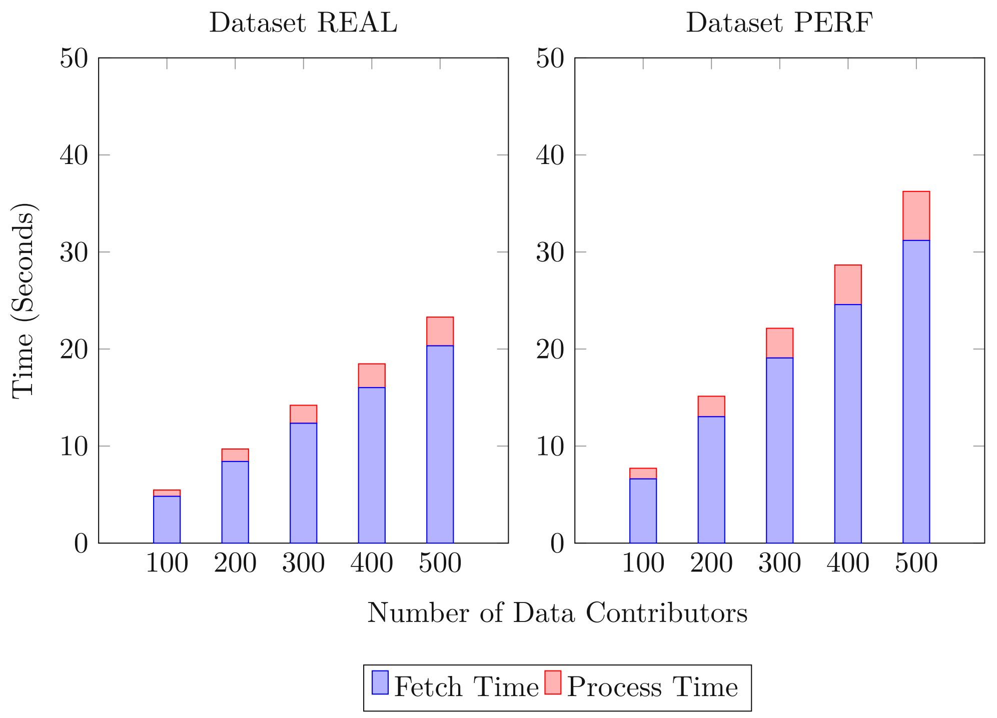

Figure 5.6: Comparison of fetching and processing times by number of *Data Contributors* for the REAL and PERF datasets, using the *Granular* strategy.

- **Processing Time:** The *Standard* approach suffers from increased processing times, as seen in Figure 5.5 due to the need to handle larger volumes of data, many of which are not relevant to the request at hand. This inefficiency is evident in the graphs, where the processing times are higher compared to the *Granular* approach, as seen in Figure 5.6.

5.3.3 Granular Approach

The *Granular* approach refines the data retrieval process by ensuring that only the necessary data elements required for a given data request are fetched. Unlike the *Standard* approach, where an entire dataset is transferred and filtered afterward, the *Granular* strategy pre-selects the relevant data within the *Solid Pod* before transmission. This minimizes both data exposure and processing overhead while improving *data minimization.*

When a *Data Contributor* grants consent to a request (as outlined in Figure 4.6), a pre-filtered dataset is generated containing only the necessary observations. In this case, the FHIR `Bundle` is constructed with only the *CREAT* and *BUN* `Observations` from the original dataset, along with the `Patient` resource.

- **Fetch Times:** The results, visualized in Figure 5.6, show that the fetch times in the *Granular* approach remain similar to the *Standard* approach. This suggests that the reduction in data volume does not significantly impact network latency, as the number of HTTP requests remains identical in both strategies.

- **Process Time:** The *Granular* strategy significantly reduces processing times, as seen in Figure 5.6. Since irrelevant data are never transferred, filtering becomes unnecessary for the *Solid Gateway*, leading to a marked decrease in computational overhead[17].

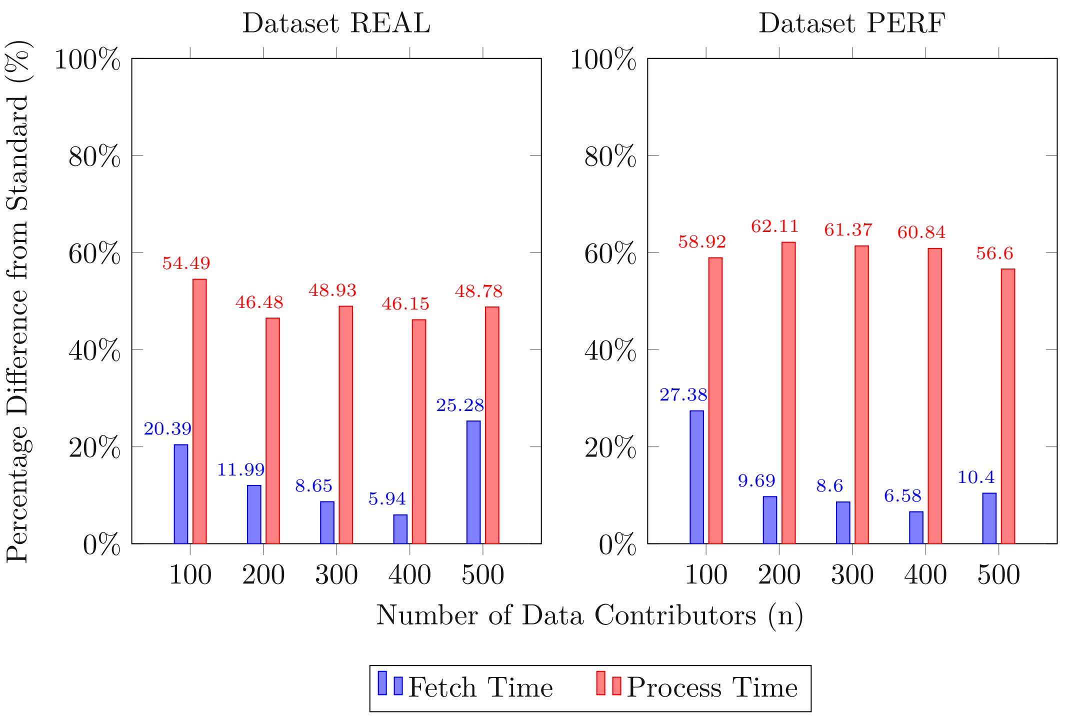

Figure 5.7: Comparing *Granular* to *Standard* strategy: Percentage difference in Fetch and Process Times for PERF and REAL datasets.

Figure 5.7 illustrates that the *Granular* strategy does not significantly impact the fetch time, as both approaches require one request per data contributor. However, *Granular* retrieval prevents unnecessary large data transfers, leading to improved network efficiency in scenarios with larger datasets. The *Granular* approach consistently outperforms the *Standard* approach in terms of processing efficiency, achieving reductions of over 50% in processing times, particularly in the PERF dataset, as shown in Table 5.10.

[17]Data is still filtered at the *Solid Gateway* to make sure, that only relevant data is passed on to the data platform.

5.3.4 Comparison to WellFort

To provide further context to our system's performance, we conducted comparative experiments against the original *WellFort* architecture. The results are visualized in Figure 5.8 for fetching and Figure 5.9 for processing, where we contrast each strategy across both the REAL and PERF datasets. The exact measurements are displayed in Table 5.10. As shown in Figure 5.4, we measure fetching performance in the original WellFort architecture from the *Data Repository* (point A) until the data is received and processed by the *Loader*, denoted by point B.

Fetching

When it comes to fetching data, *WellFort* demonstrates better performance, especially when the number of *Data Contributors* increases. *WellFort* gains a slight advantage by hosting its centralized repository locally, thereby eliminating the latency associated with external pod queries.

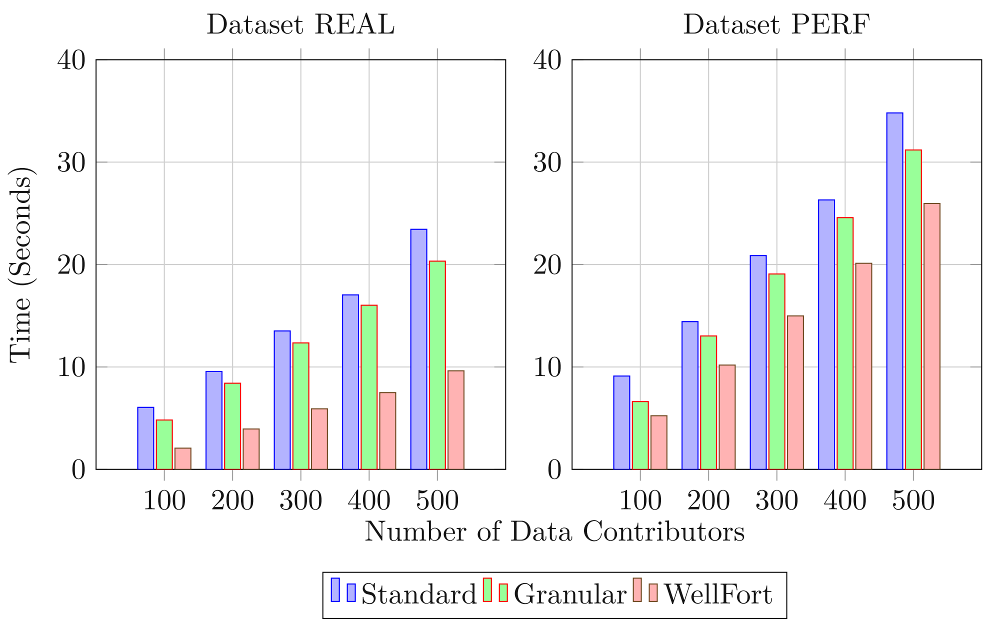

Figure 5.8: Comparison of fetching times by number of *Data Contributors* for the REAL and PERF datasets for *WellFort*, *Standard* and *Granular* strategy.

The main reason, however, is that *WellFort* leverages a triplestore with metadata about each *Data Contributor*, which allows it to quickly preselect only the *Data Contributors* who fulfill the data request (e.g., who have provided both consent and the required data attributes). This pre-filtering is highly efficient: in our experiment with the REAL dataset, only 182 out of 500 *Data Contributors* were involved in the data fetch for our data request. As a result, *WellFort* reduced the number of HTTP requests necessary and thereby improved its fetching performance in scenarios where consent is low paired with a high number of *Data Contributors*.

In contrast, the *Solid Gateway* must iterate over all registered *Data Contributors*, checking each *Solid Pod* individually for consent. This results in more HTTP requests when only a subset of *Data Contributors* are eligible, which explains the slower fetch performance under the REAL dataset. Table 5.10 shows that for *REAL500*, *WellFort* only needed 364 HTTP requests, whereas the *Solid Gateway* made 682 requests against *Solid Pods*. This ratio can also be observed in Figure 5.8, highlighting that network latency only plays a minor role in this context.

However, in scenarios where all *Data Contributors* participate (as in the PERF dataset), the performance gap between the *Granular* strategy and *WellFort* diminishes significantly, showing that the distributed pod-based approach can compete under ideal consent coverage.

Processing

The key difference lies in processing performance. As shown in Figure 5.9, *WellFort's* processing times are substantially higher than both the *Standard* and *Granular* approaches using our *Solid Gateway*.

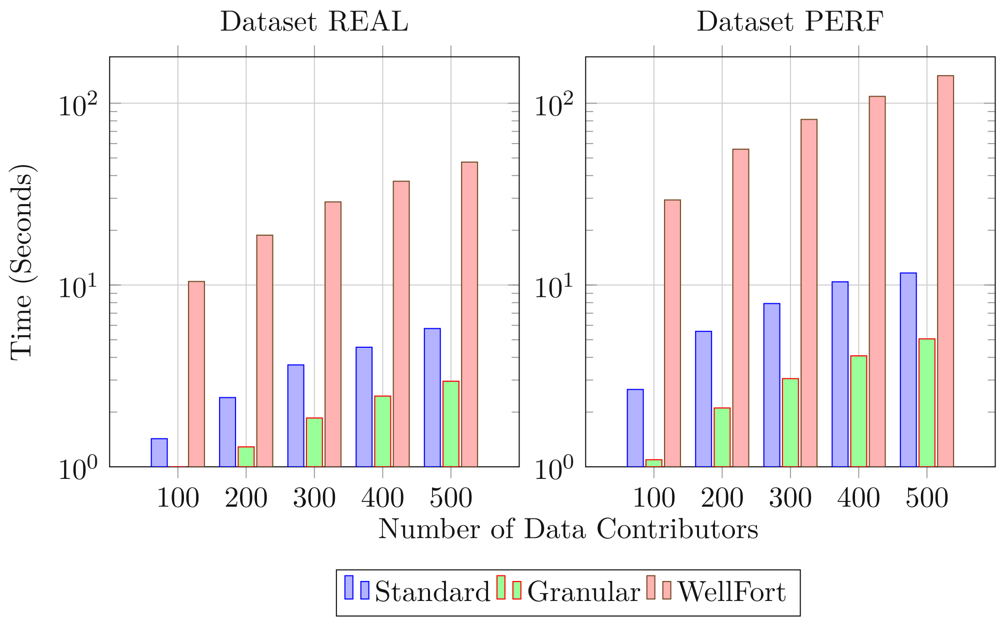

Figure 5.9: Comparison of processing times by number of *Data Contributors* for the REAL and PERF datasets for *WellFort*, *Standard* and *Granular* strategy.

This limitation is due to *WellFort's* extraction pipeline implementation, where fetched JSON data from the repository is filtered and transformed within their *Loader* component.

A detailed code analysis of this *Loader* would be necessary to identify optimization opportunities and reduce its processing overhead.

In contrast, our *Solid Gateway* processes data more efficiently. Furthermore, our *Granular* strategy minimizes processing needs by transferring only relevant, pre-filtered data from the *Data Contributor's Pod*. The *Solid Gateway*, therefore, requires even less effort to prepare data for the analysis environment, as the subset of information has already been curated by the *Data Contributor's* browser during the consent process.

We show the total time taken for fetching and processing time for the *Standard* and *Granular* approach next to *WellFort* in Figure 5.10. For this comparison, we only used the *REAL500* and *PERF500* datasets. Our *Solid Gateway* using the *Granular* strategy yields the best results overall. Even though *WellFort* has better fetching performance, the necessary processing is very high, thus leading to overall worse performance in our tests.

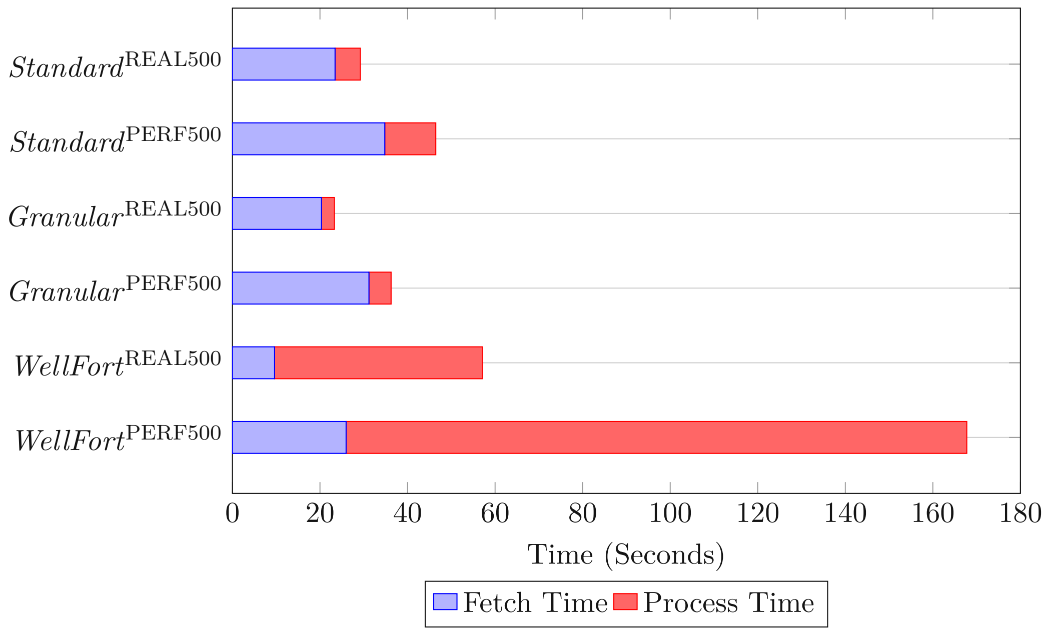

Figure 5.10: Total fetch and process time comparison across strategies (*Standard*, *Granular* and *WellFort*) and data sets (*REAL500* and *PERF500*)

Through these results, we validate that our implementation is a viable alternative to Invenio[18] as a local data repository when it comes to both fetching and processing performance.

5.3.5 Parallelization of the *Solid Gateway*

In our evaluation, we looked at how the total data retrieval and processing time changed with increasing numbers of parallel workers. We perform the same data extraction and processing operations for both the *PERF500* and *REAL500* datasets using the *Granular* approach while adjusting the worker count from one to eight as illustrated in Figure 5.11.

With parallelization, however, we encounter a limitation in measurement granularity: since each parallel unit of execution performs both fetching and subsequent processing of data for a single *Data Contributor*, the two operations overlap in time. As a result, we are unable to cleanly separate and analyze the durations of fetching and processing individually. Therefore, our evaluation focuses exclusively on the total time required to complete the entire data pipeline across all *Data Contributors*.

Despite this limitation, we observe a clear reduction in overall execution time as the number of parallel workers increases. For the *PERF500* dataset, for example, the execution time reduced from 36.24 seconds with a single worker to 17.33 seconds with eight workers. This corresponds to roughly 54% reduction in overall time taken to obtain and process the requested data. Our data show that the most important increase happens while scaling from one to four workers. In our test scenario the performance benefits diminish beyond four parallel threads.

[18] `https://inveniosoftware.org/`

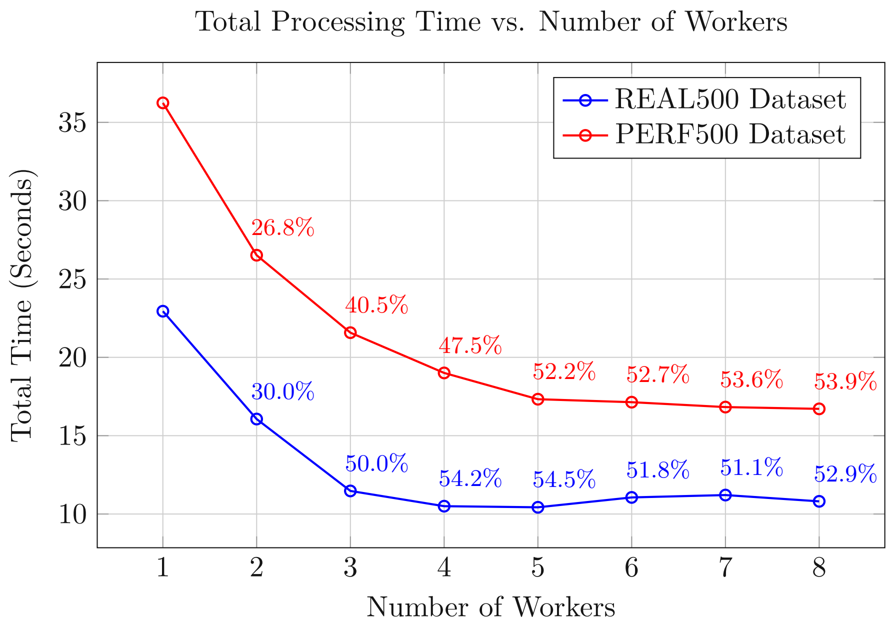

Figure 5.11: Comparing total time for PERF500 and REAL500 using the *Granular* strategy with number of workers from one to eight.

During our tests, the *Solid Server* and the *Solid Gateway* operated well within its resource limits. The CPU usage never exceeded 60%, the RAM consumption remained at approximately one third of the available memory in both systems. In addition, we monitored network traffic on servers using the `nload`[19] command line tool. Our observations confirmed that both upload and download speeds remained below 12 Mbit/s throughout the experiments, ensuring that the network bandwidth did not become a limiting factor.

These metrics indicate that the system's resource utilization was efficient, and the server was not significantly stressed even under increased parallelization.

5.4 Effectiveness in Data Exposure

This section evaluates the effectiveness of both *Standard* and *Granular* data fetching strategies in reducing unwanted data exposure. Effectiveness is measured by the amount of data leakage, defined as the transfer of non-essential triples that do not contribute to fulfilling the specific query.

5.4.1 Methodology

Data stored within *Solid Pods* are in RDF format, so each data point is represented as a triple consisting of a subject, a predicate, and an object. The experiment aims to quantitatively assess the extent of data exposure of different data retrieving strategies by counting the triples transferred in the context of a query. We use metric m, *Amount of Leaked Information* which is defined as $m \equiv |L|$ and denotes the absolute number of leaked data points [WE18], in our case RDF triples. However, this metric does not consider the sensitivity of exposed information, and since not all triples directly expose personal information of *Data Contributors*, we propose an adaptation.

Each triple in the patient data section is considered critical as it provides identifiable information about the patient. This includes details such as birth date, gender, and other demographics represented by specific triples of the data. Given that each of these triples is a direct contribution to the overall data exposure, each triple is counted individually. Typical triples included in the patient dataset contain:

- `fhir:birthDate` "1959-01-01"
- `fhir:gender` "female"
- `fhir:language` "es"

Observation data typically contain less directly identifiable information but are rich in clinical details, such as medical conditions or diagnostic results. FHIR resources such as

[19]`https://linux.die.net/man/1/nload`

`ReferenceRange`, `Coding`, and `Interpretation`, though part of an `Observation's` dataset, primarily detail the medical context rather than the patient's personal identity. For example, a triple indicating a high result for a fasting blood glucose test (`fhir:interpretation` "high") reveals information about the observation status but does not inherently increase the exposure of personal data.

In the experiment, each set of observation details, including associated triples within an observation unit (e.g. those contained within a observation resource), is counted as one unit in terms of data collection, but is analyzed separately regarding its contribution to data exposure. This segregation ensures that the evaluation remains focused on the extent of medically relevant information retrieved without unnecessarily amplifying perceived personal data exposure.

This methodology allows for a more fair assessment of data exposure in this scenario. By distinguishing between the types of information conveyed by different triples, the experiment can effectively measure the impact of data fetching strategies on data sovereignty.

Experiments are repeated with different combinations of parameters to ensure a complete evaluation.

- **Strategies:** *Standard* and *Granular*.
- **Number of Data Contributors:** 100, 200, 300, 400 and 500.
- **Datasets:** REAL and PERF.

For each combination of strategy, number of *Data Contributors*, and dataset, the fetching process via the *Solid Gateway* is triggered once, and the transmitted data are saved. We then use automated scripts to extract the relevant information.

- **Transferred Triples:** The number of triples that are actually transferred from the *Solid Pod* to the *Solid Gateway*.

As the consent has to be checked each time the *Solid Gateway* queries the *Solid Pods*, for each *Data Contributor*, one request has to be done before fetching actual data.

The results of this experiment are presented in Table 5.11 and show the number of triples transferred, for *Standard* and *Granular* strategies in the datasets and the varying numbers of *Data Contributors*.

5.4.2 Standard Approach

The *Standard* approach involves fetching the entire dataset for each data request, regardless of the specific requirements of the query. Although straightforward, this method typically results in high levels of data exposure.

Table 5.11: Number of Transferred Triples for *Standard* and *Granular* Strategies

Dataset	n	Strategy	Total	Transferred	Relevant	Irrelevant	Irrelevant %
REAL	100	Granular	2288	134	134	2154	0.00
REAL	200	Granular	4445	249	249	4196	0.00
REAL	300	Granular	6782	385	385	6397	0.00
REAL	400	Granular	9032	496	496	8536	0.00
REAL	500	Granular	11128	612	612	10516	0.00
PERF	100	Granular	3400	300	300	3100	0.00
PERF	200	Granular	6800	600	600	6200	0.00
PERF	300	Granular	10200	900	900	9300	0.00
PERF	400	Granular	13600	1200	1200	12400	0.00
PERF	500	Granular	17000	1500	1500	15500	0.00
REAL	100	Standard	2288	2288	134	2154	94.14
REAL	200	Standard	4445	4445	249	4196	94.40
REAL	300	Standard	6782	6782	385	6397	94.32
REAL	400	Standard	9032	9032	496	8536	94.51
REAL	500	Standard	11128	11128	612	10516	94.50
PERF	100	Standard	3400	3400	300	3100	91.18
PERF	200	Standard	6800	6800	600	6200	91.18
PERF	300	Standard	10200	10200	900	9300	91.18
PERF	400	Standard	13600	13600	1200	12400	91.18
PERF	500	Standard	17000	17000	1500	15500	91.18

- **Triples Transferred:** Each query in the *Standard* approach results in the transfer of all triples in a *Solid Pod*, which, for our standardized datasets, equals 17000 triples for PERF500 and 11128 for REAL500 as can be seen in Table 5.11.
- **Unwanted Data Leakage:** Given that specific queries only required a fraction of the available data, the unwanted data leakage was substantial, with 94.5% of triples being irrelevant to the data request for the REAL500 dataset and 91.18% for all PERF datasets.

While there is a mechanism in place that prevents *Data Contributors* from submitting any data when not all requested triples of the data request can be covered by the *Data Contributor's* data, the results demonstrate a significant inefficiency in the *Standard* approach, as a large volume of irrelevant information is consistently transferred, increasing the risk of data exposure and violating principles of *data minimization*.

5.4.3 Granular Approach

Contrasting with the *Standard* method, the *Granular* approach aims to fetch only those triples that are directly relevant to the data consumer's query. This approach significantly reduces unwanted data leakage.

- **Triples Transferred:** Implementing the *Granular* approach, each query specifically fetched only the necessary triples, significantly lowering the total number of triples transferred. With the *Granular* approach, only 612 of 11128 relevant triples have been transferred for the REAL500 dataset, only 5.5% of the total data had to be shared and transferred with the *Solid Gateway*.
- **Unwanted Data Leakage:** The *Granular* approach dramatically reduced unwanted data leakage; no unwanted critical triples have been transferred, as shown in Table 5.11.

The *Granular* approach, with its targeted data retrieval strategy, stands out for its efficiency in reducing unwanted data leakage and, therefore, enhancing data sovereignty. As demonstrated in the REAL500 dataset, the granular approach efficiently transferred only 5.5% of the total data, ensuring that no irrelevant or unnecessary data were communicated. This precision not only streamlines the processing and reduces bandwidth consumption but also reduces the data at risk for potential privacy breaches.

Discussion

To better understand the implications of different data retrieval strategies on privacy and performance, we compared our *Granular* approach with two widely adopted models: the strategy employed by *TIDAL* and the architecture used by *WellFort*. The *Standard* approach used in our evaluation corresponds to the data access method implemented by *TIDAL*, where entire files are shared and fetched if they contain any relevant information for a request. While straightforward, this model leads to a significant transfer of irrelevant data, as all information within a file is made available once access is granted, even if only a small part is needed.

In contrast, our proposed *Granular* strategy limits the scope of access by creating minimal, request-specific files that contain only the precise data needed for a given request. These files are generated dynamically and selectively authorized by the *Data Contributor*, resulting in drastically reduced data exposure and improved alignment with the principle of *data minimization*.

WellFort, on the other hand, adopts a different model entirely. It collects and stores the complete datasets of all *Data Contributors* in a centralized local repository ahead of any specific request. Although this setup allows for quick querying and filtering, it implies that the platform operator always has access to all personal data. This storage model, though beneficial for fast data access, undermines data sovereignty principles, as it posses full datasets without a direct and current need for their processing.

Our evaluation results, discussed in Section 5.4, show the clear benefits of the *Granular* approach. Using the *REAL500* dataset, the *Standard* (i.e., TIDAL) strategy exposed approximately 94.5% of transferred triples as irrelevant to the actual request. In comparison, the *Granular* strategy limited data transfer strictly to requested information, resulting in zero leakage of unrelated data when measuring individual data points.

Ultimately, the *Granular* strategy not only ensures *data minimization* when measuring absolute data points but also achieves competitive performance. We thus provide a more privacy-respecting and technically scalable alternative to both the *TIDAL* and *WellFort* designs.

5.5 Guidelines for Integrating Solid Pods into Existing Data Platforms

To support the integration of *Solid Pods* as decentralized storage in existing data platforms, we propose the following guidelines.

G1 Solid Infrastructure
A *Solid Server* should be hosted directly by the platform provider. For improved reliability and availability, at least two independent *Solid Server* instances should be deployed. These servers act as both the *Identity Provider* for request-specific WebIDs and as the host for published data requests.

G2 Processes Adaption
The *Solid Gateway* should be integrated as a central component of the data platform to manage request-based access to *Solid Pods*. The *Solid Gateway* introduces a controlled workflow in which data can only be retrieved after explicit consent from *Data Contributors*. The *Solid Gateway* does not store data itself; it acts as a mediator between the platform and external *Solid Pods*. Its integration should replace direct database queries in the platform's data flow and processing pipeline.

G3 Data Sharing Algorithm
Data minimization should be enforced through a sharing strategy that extracts only the necessary data for each request. This is implemented via a client-side *Request Portal* that inspects requests and processes data locally in the user's browser. This application interacts solely with the user's *Solid Pod* and does not involve processing by the data platform.

G4 Consent via User Application
To ensure continuous data sovereignty, the platform should provide a client-side *Request Portal* that allows contributors to easily review and revoke access to their data. This application interacts solely with the user's *Solid Pod* and does not involve back-end processing by the data platform.

G5 Privacy-Preserving Software
Data retrieved from *Solid Pods* through the *Solid Gateway* must only be processed

within secure isolated environments without third-party access. Privacy-preserving software, such as DataSHIELD, should be used to ensure that only aggregated or anonymized results are accessible to the party interacting with the data.

G6 Provenance and Logging
The *Solid Gateway* logs relevant provenance metadata (timestamps, hashes, WebIDs, etc.) in its local database. This provenance information should be integrated into existing logging and auditing systems of the platform to maintain traceability and compliance.

5.6 Summary

In this chapter, we evaluated the proposed integration of *Solid Pods* into a privacy-preserving data platform in multiple dimensions. First, Section 5.2 verified that the solution meets the core functional requirements, showing that user consent, non-disclosive analytics, and decentralized data hosting can all be maintained without undermining platform workflows.

Section 5.3 compared two retrieval strategies, *Standard* and *Granular*, in terms of data fetch times and processing overhead and evaluated the performance benefits of applying parallelization to the *Solid Gateway*. The *Granular* approach significantly reduced processing time while maintaining comparable fetch times to the *Standard* approach. Additionally, we demonstrated that increasing the number of parallel workers can reduce the total execution time by more than 54%.

Further, we compared our implementation against the *WellFort* [EEM$^+$21] architecture in Subsection 5.3.4, which relies on a centralized local repository. While *WellFort* showed better fetch performance due to pre-filtering via metadata stored in a triplestore, it incurred substantially higher processing times, leading to an overall slower execution time. Our evaluation confirmed that the decentralized *Solid Gateway* using the *Granular* strategy outperformed *WellFort* in total retrieval and processing time, especially under full consent scenarios.

Section 5.4 quantified how our *Granular* retrieval reduces data leakage by measuring the proportion of unnecessary triples transferred. We showed that the proposed selective sharing mechanism effectively reduces the transfer of irrelevant or sensitive data. In contrast, the *Standard* approach, which corresponds to the sharing model of *TIDAL* [SOSD23], results in substantial data exposure due to full file transfers when any part of the file is relevant. Moreover, *WellFort*'s architecture stores entire personal datasets centrally, even without current processing needs, which violates the principle of data minimization.

Finally, Section 5.5 draws on these findings to propose a set of guidelines for implementing *Solid* in existing data platforms that rely on local data repositories, ensuring better alignment with user sovereignty, privacy principles, and system performance.

CHAPTER 6

Conclusion

Data sovereignty and privacy protection drove the development of innovative data platforms that balance secure data access with regulatory compliance. As concerns about centralized data control and exposure risks grow, distributed analysis techniques have gained traction, allowing computations across multiple data sources while restricting direct access to raw data. Solutions such as DataSHIELD demonstrated the feasibility of privacy-preserving analytics, allowing institutions to collaborate without fully sharing sensitive information. However, many existing frameworks either rely on centralized repositories for processing or lack user-centric control over data access, especially with regard to *data minimization*, proposed by the GDPR in Art. 5. To address this, we examined two complementary approaches. *WellFort*, which provides a structured, privacy-preserving analytics environment, and *TIDAL*, which leverages *Solid Pods* to give individuals full authority over their data. By combining key aspects of these models, we proposed an integrated solution that enhances data sovereignty for data contributors while maintaining the analytical capabilities required for collaborative research for those who consume donated data.

To achieve this, we designed and implemented the *Solid Gateway*[1], a mediator that allows data platforms to interact with *Solid Pods* while ensuring controlled request-based data access. Our design introduced a granular data-sharing mechanism that segments and retrieves only the minimal data required for a data request. Furthermore, our solution was integrated into the privacy-preserving data platform *WellFort* by replacing its *Secure Repository* with our proposed *Solid Gateway*, confirming the practicability of our solution. Through detailed sequence diagrams, we demonstrated how our architecture facilitated data requests, consent management, and data retrieval using DataSHIELD as privacy-preserving layer.

[1]`https://github.com/kooperativ97/solid-gateway`

To validate our approach, we performed a comprehensive functional evaluation, verifying that all core requirements such as privacy preservation, data minimization, transparency, and consent management were met. By implementing structured data request handling and integrating *Solid Pods* for storage, we ensured that *Data Contributors* retain full control over their data. Functional tests confirmed that our system effectively governs access, enforces consent policies, and allows users to revoke permissions at any time, giving direct control to *Data Contributors*.

In addition to functional verification, we performed an efficiency analysis comparing *Standard* and *Granular* data retrieval strategies. Our results demonstrated that the *Standard* approach lead to substantial data exposure by transferring a significant amount of irrelevant personal information. Our proposed *Granular* approach, in contrast, effectively minimized unwanted data leakage by retrieving only the necessary data for analysis.

Finally, we compared our proposed solution against the original *WellFort* architecture and demonstrated that our approach offered competitive fetch performance under full consent conditions and significantly improves processing efficiency. While *WellFort* benefits from using metadata to instantly query consent and data availability, our system reduced the overhead of filtering and transformation.

In conclusion, our work presents a significant step toward integrating user-controlled decentralized storage into data platforms. To support the adoption of this approach, we proposed a set of guidelines outlining how *Solid Pods* can be integrated into existing data platforms, ensuring a well-defined interaction between decentralized data storage and privacy-preserving computation environments. These guidelines provide practical steps for platform providers to transition toward a more user-centric data architecture while maintaining compliance with data governance standards.

6.1 Research Questions Revisited

In this section, we revisit the research questions introduced in Chapter 1 and summarize how they have been addressed throughout this work, providing references to the corresponding sections where each topic is explored in detail.

Main Research Question: What is a proper solution to combine *Solid* with privacy-preserving analysis tools in a way that avoids centralized data storage while still meeting the functional requirements of data platforms?

We started by investigating current approaches in data platforms and distributed analysis in Chapter 2 to identify the current state of data platforms in regards to privacy-preservation and data sovereignty. In Chapter 3, we synthesized these observations into concrete functional requirements that guided our integration strategy. To fulfill these requirements, we proposed the *Solid Gateway* in Chapter 4 as an architectural mediator, bridging *Solid Pods*, which grant users autonomous control over their personal data, with an existing privacy-preserving platform architecture; *WellFort's Trusted*

Analysis Environment. We showed, that the resulting framework supports non-disclosive computations through DataSHIELD with request-based data fetching from *Solid Pods*.

Section 5.2 details how the system components interact to satisfy key functional requirements: user consent is explicitly captured and enforced at all times, data ingestion avoided unnecessary transfers through selective retrieval, and overall analysis workflows remained consistent with existing data platform standards. Moreover, Sections 5.3 and 5.4 demonstrate that this solution addresses *data minimization*, as it minimizes the exposure of irrelevant information during research tasks. Together, the results validate that *Solid Pods* can be integrated into privacy-preserving data platforms while still meeting the functional requirements of data platforms.

RQ1: In what way can *Solid* be integrated into privacy-preserving computation platforms to minimize the performance impact?

The integration of *Solid* into privacy-preserving computation platforms introduces performance challenges mainly due to the decentralized nature of *Solid Pods*, which requires individual connections to each *Data Contributor's Pod* and data file. In this thesis, we analyzed this impact through the evaluation of two retrieval strategies: the *Standard* approach, in which full datasets are fetched and filtered within the *Solid Gateway*, and the *Granular* approach, which minimizes the data fetched by creating minimal request-specific files containing only the necessary information. In addition, we investigated the effect of parallelization of the *Solid Gateway*, employing multiple concurrent workers to fetch and process data from *Solid Pods*. Both strategies were evaluated and compared in depth in Section 5.3 using different parameters for number of *Data Contributors* and dataset.

The evaluation revealed that, for 500 *Data Contributors* for the PERF dataset, the *Standard* strategy completed the data retrieval phase in 34.8 seconds and required an additional 11.6 seconds for local processing, as the *Solid Gateway* had to filter large amounts of irrelevant data. In contrast, the *Granular* strategy required 31.2 seconds for data retrieval and reduced the processing time to 5 seconds due to the minimized pre-filtered datasets. As detailed in Section 4.5, this performance difference arises because the *Granular* strategy creates a small file, including only relevant participant data such as patient attributes and requested types of observations. The *Granular* strategy provides clear privacy benefits by reducing the volume of irrelevant data transferred and processed, as further evidenced by the results in Section 5.4.

We further evaluated the effectiveness of our solution by comparing it to the original *WellFort* architecture. While *WellFort* demonstrated faster data fetching performance, primarily due to its centralized infrastructure and preselection of eligible *Data Contributors* through their triplestore, it suffered from significantly higher processing times. We attribute it to the inefficient extraction pipeline in the *Loader* component, which introduced substantial overhead.

Moreover, when comparing the total time for fetching and processing, our *Granular* strategy outperformed *WellFort* across both the *REAL500* and *PERF500* datasets. These results support the viability of our decentralized, user-centric approach as an alternative to centralized repositories such as *Invenio*.

Furthermore, we applied parallelization to the *Solid Gateway* and observed that increasing the number of parallel workers significantly reduced the overall execution time. Specifically, our experiments showed that by scaling up to eight workers, the overall time taken was reduced by approximately 54% lowering the time required from 31.2 seconds to only 16.7 seconds. Thus, we demonstrated that parallelization of fetching is highly effective in decentralized data retrieval using *Solid*.

This implementation serves as a baseline for further optimization. Implementing caching strategies and load balancing between different *Solid Servers* would further reduce latency and improve scalability, which are promising areas for future development.

In conclusion, integrating *Solid* into privacy-preserving computation platforms proved not only feasible but also competitive in terms of performance. While decentralized architectures inherently introduce coordination overhead due to data fragmentation across individual *Solid Pods*, our evaluation showed that with appropriate design choices, such as pre-filtered data via the *Granular* strategy and parallelized fetching, this overhead can be minimized effectively. In fact, our system outperformed the centralized *WellFort* architecture, which relies on a local repository (*Invenio*), particularly in terms of total data processing time. Despite *WellFort*'s faster initial data selection due to its centralized triplestore, the extensive post-processing required in its *Loader* component led to significantly longer execution times. This highlights that our decentralized approach using *Solid*, when well-engineered, can effectively replace local data repositories to support data sovereignty and privacy-preserving computation in data platforms.

RQ2: To what extent can we reduce the information exposure from individuals contributing to the privacy-preserving computation platforms by using *Solid*?

We demonstrated that *Solid's* decentralized architecture, combined with the *Granular* sharing strategy, provided a powerful means to reduce information exposure. When used correctly, *Solid's* fine-grained access control mechanisms, allow *Data Contributors* to share only the specific information relevant to a given research request. In our proposed system, this was achieved through the dynamic creation of minimal request-specific data files that are selectively authorized for access via ACL rules tied to the WebID of the data request. This ensured that *Data Contributors* retain control and transparency over exactly which parts of their data are shared.

As presented in Section 5.4, the evaluation demonstrated a significant reduction in information exposure when comparing *Standard* and *Granular* sharing strategies. Using the REAL500 dataset, the *Standard* approach exposed approximately 94.5% irrelevant triples, while the *Granular* approach reduced data transfer to only the explicitly requested information, resulting in no leakage of unrelated data when measuring absolute data points. This indicated that *Solid*, when combined with our *Granular* sharing strategy, can minimize data exposure to the smallest possible subset required for the request, thus upholding the principle of *data minimization*.

Compared to existing systems such as *TIDAL* [SOSD23] and *WellFort* [EEM$^+$21], our *Granular* strategy demonstrates superior adherence to the principle of *data minimization*. In *TIDAL*, while data is also stored and retrieved from a personal *Solid Pod*, access is granted to complete files if they contain relevant information, regardless of whether the entire content is needed. This behavior is mirrored by our *Standard* strategy, where the evaluation showed that a large part of the shared data was irrelevant to the actual research request. Similarly, *WellFort* stores all *Data Contributors* data in a centralized data repository, where data is stored ahead of time, even if it is not immediately needed for a specific analysis. This storage model effectively results in full data availability to the platform operator, which, while convenient for analysis, undermines the principle of *data minimization* as personal data is stored by a third party without an immediate need for processing.

In contrast, our *Granular* strategy ensured that only the exact, relevant subset of data is prepared and authorized for access at the time of the request. This means no additional or unrelated personal data was exposed, stored, or processed beyond what was explicitly required. By dynamically creating request-specific data snapshots and enforcing access control on a per-request basis, our system effectively minimized data leakage. This positions our approach as a privacy-preserving alternative that exceeds both *TIDAL* and *WellFort* in terms of limiting unnecessary information exposure.

However, it is important to recognize the context and limitations of this evaluation. Both *Standard* and *Granular* strategies were tested on self-constructed datasets, specifically designed to allow control over the proportion of irrelevant versus relevant information. This controlled setup was essential to assess the effectiveness of the *Granular* sharing approach within our architecture. However, it also means that the reported minimization rates cannot be directly compared to other studies or applied as universal benchmarks. In fact, the proportion of irrelevant data in the test dataset directly influences the measurable degree of minimization: a higher presence of irrelevant data leads to greater minimization effects, while a more focused dataset would reduce the observed gains. Therefore, while the reduction of approximately 95% in our evaluation provides strong evidence of the method's capacity under controlled conditions, it should be interpreted as a proof of concept rather than as a definitive performance metric.

Nevertheless, this work established a reproducible experimental framework and validated the underlying mechanism by which *Solid* can effectively reduce data exposure through granular sharing. It showed that, beyond specific numbers, the architectural approach scales its minimization benefits proportionally to the amount of irrelevant data present. This is a valuable property in contexts with large, heterogeneous personal datasets. Future research should aim to extend this evaluation to real-world data and conduct comparative studies with alternative privacy-preserving systems to further substantiate these findings and refine minimization strategies in practice.

6.2 Limitations and Future Research

Building on this work, several avenues for improvement remain. Future research could explore the integration of federated learning for decentralized computation, improving data reproducibility, and extending query capabilities within the *Solid* ecosystem.

6.2.1 Solid Pods and Federated Learning

While the present work focuses on the use of *Solid Pods* as a decentralized storage solution, as argued in Chapter 2, a promising direction for future research is the integration of federated learning within the *Solid* ecosystem. Frameworks like WebFed [LYZS21] offer a browser-based platform for federated learning that might serve as a valuable building block to integrate these capabilities into the *Solid Server* itself or as solid-based application like the *Request Portal*. By developing extensions or modules that allow *Solid Pods* to interface with a federated learning layer, future research could explore hybrid models where user data remain securely stored within their pods, yet participate in collaborative model training.

Including computational frameworks like federated learning into the *Solid* ecosystem might lead to fragmentation among *Solid* servers. Different levels of support for such extensions could be offered by each provider's server, therefore complicating interoperability and maybe reducing freedom of choice. Future studies should therefore focus on creating a consistent extension to the *Solid Protocol* that combines computational modules to solve this problem. While maintaining the fundamental ideas of data sovereignty and user control, this standardization would guarantee that any *Solid* server, independent of the provider, can support advanced analytics and privacy-preserving computations in a uniform and interoperable manner.

6.2.2 Reproducibility

The architecture proposed in this thesis already establishes an initial form of reproducibility by creating copies of the requested data during the acceptance of a data request, as outlined in Section 4.5. These copies serve as isolated snapshots, decoupled from their original evolving sources within the contributor's *Solid Pod*. However, without additional safeguards, the integrity of these snapshots cannot be guaranteed, as they remain modifiable by the data contributor or other authorized agents.

To address this, a future enhancement could involve encrypting the snapshot at the moment of creation using a public key associated with the specific data request. This key would only be known to the *Solid Gateway* or the associated request-specific pod. By doing so, a tamper-evident snapshot could be created that remains stored within the *Data Contributor's Pod* but whose contents cannot be altered without invalidating the encryption. Although the *Data Contributor* retains the right to delete the snapshot entirely, the encrypted form ensures that any unauthorized modifications of the stored data would be detectable. In this way, the system would preserve the sovereignty of data while introducing a mechanism to ensure the integrity and authenticity of the shared data over time.

This cryptographic approach would complement the existing provenance features and directly support RDA Data Citation Recommendation R6 (Result Set Verification) [RAUP16]. However, fundamental questions remain regarding how persistent identifiers (R1), versioning, and formal citation mechanisms (R8, R10) can be integrated into decentralized, user-centric environments like *Solid* without reintroducing centralization. Therefore, future work should explore how FAIR principles and robust data citation practices can be adapted to data platforms that leverage *Solid Pods* as a decentralized and user-centric storage solution.

6.2.3 SPARQL

A current limitation in the *Solid* ecosystem is the absence of a native SPARQL interface. Although earlier specifications included a native SPARQL GET operation, this feature was removed from the Community Solid Server following discussions on GitHub[2]. As a workaround, developers rely on frameworks such as Comunica[3] to run SPARQL queries on RDF data. However, this approach typically requires transmitting the entire dataset to the device running Comunica for local filtering, which may expose data unnecessarily. In contrast, commercial solutions like Inrupt's[4] Query service offer a Quad Pattern Fragment (QPF) interface for querying RDF data in *Solid Pods*, yet the lack of publicly available implementation details creates vendor lock-in and leads to a fragmented landscape of query capabilities.

A promising direction for future research is to integrate the Comunica library directly into open source *Solid* server implementations to allow SPARQL queries to be processed locally, thus preserving data privacy and improving query efficiency. In this context, ESPRESSO provides a valuable reference, demonstrating how decentralized and privacy-preserving search, including SPARQL querying, can be enabled across *Solid Pods* through local indexing and federated query propagation [RSM+23].

6.2.4 Autonomous Agents for Consent Management

A potential direction for future research involves the design and implementation of autonomous agents that manage data access requests on behalf of *Data Contributors*, based on user preferences. These agents assess data requests and check compliance with the contributor's consent policies and request parameters, such as the purpose of use, the requesting entity, or the sensitivity of the requested data. Performing similarly to smart rules or smart contracts as described in [NPDS19] or [Com23], such agents could autonomously accept, reject, or modify access permissions by updating ACL files, thus eliminating the need for manual approval of each individual request.

[2] https://github.com/nodeSolidServer/node-solid-server/issues/962
[3] https://comunica.dev/
[4] https://www.inrupt.com/

6.2.5 Solid Server Setup

A notable limitation of the current architecture lies in the simplified deployment of the *Solid Server* environment. In the present setup, all patient data are hosted on a single *Solid Server* instance. Although this configuration helps initial prototyping and evaluation, it does not reflect the decentralized and distributed nature envisioned by *Solid* for real-world applications, where users maintain their own pods across diverse independent infrastructures. Future research should therefore explore more realistic deployment models involving multiple, independently operated *Solid Server* distributed across different institutions or managed directly by individual users. Furthermore, we did not evaluate data that is linked between multiple pods or servers. Moreover, the parallelization algorithm should be further optimized to dynamically scale based on the number and distribution of distinct *Solid Server* instances, thereby possibly enhancing performance in distributed real-world deployments.

6.2.6 Data Minimization

A core principle of privacy-preserving data processing is *data minimization*, as established in Article 5 of the *GDPR*. This principle requires that personal data must be adequate, relevant, and limited to what is necessary for their intended purpose. Although this thesis focused on minimizing the absolute number of data points transmitted, additional strategies could further reduce information exposure beyond selective retrieval. Incorporating more advanced information loss metrics, such as those proposed by Wagner et al. [WE18], would allow for a more detailed evaluation of minimization effectiveness, making the evaluation of future sharing algorithms better comparable to other privacy-preserving approaches.

List of Figures

List of Tables

Bibliography

[BCvS+20] Oya Beyan, Ananya Choudhury, Johan van Soest, Oliver Kohlbacher, Lukas Zimmermann, Holger Stenzhorn, Md. Rezaul Karim, Michel Dumontier, Stefan Decker, Luiz Olavo Bonino da Silva Santos, and Andre Dekker. Distributed Analytics on Sensitive Medical Data: The Personal Health Train. *Data Intelligence*, 2(1-2):96–107, January 2020.

[BFRLG21] Arnaud Braud, Gael Fromentoux, Benoit Radier, and Olivier Le Grand. The Road to European Digital Sovereignty with Gaia-X and IDSA. *IEEE Network*, 35(2):4–5, March 2021.

[CGFR+20] Stephanie Russo Carroll, Ibrahim Garba, Oscar L. Figueroa-Rodríguez, Jarita Holbrook, Raymond Lovett, Simeon Materechera, Mark Parsons, Kay Raseroka, Desi Rodriguez-Lonebear, Robyn Rowe, Rodrigo Sara, Jennifer D. Walker, Jane Anderson, and Maui Hudson. The CARE Principles for Indigenous Data Governance. *Data Science Journal*, 19:43, November 2020.

[CLC+18] Andy Crabtree, Tom Lodge, James Colley, Chris Greenhalgh, Kevin Glover, Hamed Haddadi, Yousef Amar, Richard Mortier, Qi Li, John Moore, Liang Wang, Poonam Yadav, Jianxin Zhao, Anthony Brown, Lachlan Urquhart, and Derek McAuley. Building accountability into the Internet of Things: the IoT Databox model. *Journal of Reliable Intelligent Environments*, 4(1):39–55, April 2018.

[Com23] European Commission. Decode: Decentralised citizens owned data ecosystem, 2023. Available at: `https://cordis.europa.eu/project/id/732546`.

[Dat19] Data Ethics Commission. Gutachten der datenethikkommission. `https://www.bmi.bund.de/SharedDocs/downloads/DE/publikationen/themen/it-digitalpolitik/gutachten-datenethikkommission.pdf?__blob=publicationFile&v=8`, 2019. Accessed: 10.01.2025.

[DCC+23] Alexandra Descarpentrie, Lucinda Calas, Maxime Cornet, Barbara Heude, Marie-Aline Charles, Demetris Avraam, Sonia Brescianini, Tim Cadman, Ahmed Elhakeem, Sílvia Fernández-Barrés, Jennifer R. Harris, Hazel Inskip, Jordi Julvez, Sabrina Llop, Katerina Margetaki, Silvia Maritano, Johanna Lucia Thorbjornsrud Nader, Theano Roumeliotaki, Theodosia Salika, Mikel Subiza-Pérez, Marina Vafeiadi, Martine Vrijheid, John Wright, Tiffany Yang, Patricia Dargent-Molina, and Sandrine Lioret. Lifestyle patterns in European preschoolers: Associations with socio-demographic factors and body mass index. *Pediatric Obesity*, 18(12):e13079, December 2023.

[DDO$^+$20] Timo M. Deist, Frank J.W.M. Dankers, Priyanka Ojha, M. Scott Marshall, Tomas Janssen, Corinne Faivre-Finn, Carlotta Masciocchi, Vincenzo Valentini, Jiazhou Wang, Jiayan Chen, Zhen Zhang, Emiliano Spezi, Mick Button, Joost Jan Nuyttens, René Vernhout, Johan Van Soest, Arthur Jochems, René Monshouwer, Johan Bussink, Gareth Price, Philippe Lambin, and Andre Dekker. Distributed learning on 20 000+ lung cancer patients – The Personal Health Train. *Radiotherapy and Oncology*, 144:189–200, March 2020.

[EAK$^+$19] Fajar J. Ekaputra, Peb Ruswono Aryan, Elmar Kiesling, C. Fabianek, and E. Gringinger. Semantic Containers for Data Mobility: A Seismic Activity Use Case. 2019.

[EEM$^+$21] Fajar J. Ekaputra, Andreas Ekelhart, Rudolf Mayer, Tomasz Miksa, Tanja Sarcevic, Sotirios Tsepelakis, and Laura Waltersdorfer. Semantic-enabled architecture for auditable privacy-preserving data analysis. 2021.

[EMMA$^+$23] Xavier Escribà-Montagut, Yannick Marcon, Demetris Avraam, Soumya Banerjee, Tom R P Bishop, Paul Burton, and Juan R González. Software Application Profile: ShinyDataSHIELD—an R Shiny application to perform federated non-disclosive data analysis in multicohort studies. *International Journal of Epidemiology*, 52(1):315–320, February 2023.

[Eur20] European Parliament. Directorate General for Parliamentary Research Services. *The impact of the general data protection regulation on artificial intelligence.* Publications Office, LU, 2020.

[FBP23] Khalid U. Fallatah, Mahmoud Barhamgi, and Charith Perera. Personal Data Stores (PDS): A Review. *Sensors*, 23(3):1477, January 2023. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[FE23] Marcu Florea and Beatriz Esteves. Is Automated Consent in Solid GDPR-Compliant? An Approach for Obtaining Valid Consent with the Solid Protocol. *Information*, 14(12):631, November 2023. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute.

[Fin13] Arlene G Fink. *Conducting research literature reviews.* SAGE Publications, Thousand Oaks, CA, 4 edition, September 2013.

[Hal22] Harry Halpin. All that is Solid Melts into Air: Towards Decentralized Cryptographic Access Control. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–6, Vienna Austria, August 2022. ACM.

[HGP+22] Marius Herr, Michael Graf, Peter Placzek, Florian König, Felix Bötte, Tyra Stickel, D. Hieber, Lukas Zimmermann, Michael Slupina, C. Mohr, Stephanie Biergans, Mete Akgün, Nícolas Pfeifer, and O. Kohlbacher. Bringing the Algorithms to the Data - Secure Distributed Medical Analytics using the Personal Health Train (PHT-meDIC). *ArXiv*, abs/2212.03481:null, 2022.

[Hoc23] Patrick Hochstenbach. The Solid Protocol and Its Potential for Open Science. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 316–317, Santa Fe, NM, USA, June 2023. IEEE.

[JCS20] Heleen Janssen, Jennifer Cobbe, and Jatinder Singh. Personal information management systems: a user-centric privacy utopia? *Internet Policy Review*, 9(4), December 2020.

[JSYH22] Jan Jürjens, Simon Scheider, Furkan Yildirim, and Michael Henke. Tokenomics: Decentralized incentivization in the context of data spaces. In Boris Otto, Michael Ten Hompel, and Stefan Wrobel, editors, *Designing Data Spaces*, pages 91–110. Springer International Publishing, Cham, 2022.

[Kom21] Konstantinos Komaitis. Europe's ambition for digital sovereignty must not undermine the Internet's values. *Computer Fraud & Security*, 2021(1):11–13, January 2021.

[LS22] Christoph Schlueter Langdon and Karsten Schweichhart. Data spaces: First applications in mobility and industry. In Boris Otto, Michael Ten Hompel, and Stefan Wrobel, editors, *Designing Data Spaces*, pages 493–512. Springer International Publishing, Cham, 2022.

[LYZS21] Zhuotao Lian, Qinglin Yang, Qingkui Zeng, and Chunhua Su. WebFed: Cross-platform Federated Learning Framework Based on Web Browser with Local Differential Privacy, October 2021. arXiv:2110.11646 [cs].

[MPS+21] Virginia Graciano Martinez, Luís Ferreira Pires, Luiz Santos, J. Moreira, and Renata Souza. A Framework for Staging Personal Health Trains in the Cloud. 2021.

[NL22] Lars Nagel and Douwe Lycklama. How to build, run, and govern data spaces. In Boris Otto, Michael Ten Hompel, and Stefan Wrobel, editors, *Designing Data Spaces*, pages 17–28. Springer International Publishing, Cham, 2022.

[NPDS19] Dinh C. Nguyen, Pubudu N. Pathirana, Ming Ding, and Aruna Seneviratne. Blockchain for Secure EHRs Sharing of Mobile Cloud Based E-Health Systems. *IEEE Access*, 7:66792–66806, 2019.

[Pau20a] Jason Paulos. Investigating Decentralized Management of Health and Fitness Data. 2020.

[Pau20b] Jason Paulos. Investigating decentralized management of health and fitness data, September 2020. Available at `https://hdl.handle.net/1721.1/127459`.

[PSTW15] Eliza Papadopoulou, Alex Stobart, Nick K. Taylor, and M. Howard Williams. Enabling Data Subjects to Remain Data Owners. January 2015.

[PTRC07] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24:45–77, 01 2007.

[RAUP16] Andreas Rauber, Ari Asmi, Dieter Van Uytvanck, and Stefan Proell. Data Citation of Evolving Data: Recommendations of the RDA Working Group on Data Citation (WGDC). 2016. Publisher: Research Data Alliance Version Number: 1.

[RSM$^+$23] Mohamed Ragab, Yury Savateev, Reza Moosaei, Thanassis Tiropanis, Alexandra Poulovassilis, Adriane Chapman, and George Roussos. ESPRESSO: A Framework for Empowering Search on Decentralized Web. In Feng Zhang, Hua Wang, Mahmoud Barhamgi, Lu Chen, and Rui Zhou, editors, *Web Information Systems Engineering – WISE 2023*, volume 14306, pages 360–375. Springer Nature Singapore, Singapore, 2023. Series Title: Lecture Notes in Computer Science.

[SMH$^+$16] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: A Platform for Decentralized Social Applications Based on Linked Data. 2016.

[SOSD23] Chang Sun, Marc Gallofré Ocaña, J. V. Soest, and M. Dumontier. ciTIzen-centric DAta pLatform (TIDAL): Sharing distributed personal data in a privacy-preserving manner for health research. *Semantic Web*, 14:977–996, 2023.

[SPM$^+$22] Luiz Olavo Bonino da Silva Santos, Luís Ferreira Pires, Virginia Graciano Martinez, João Luiz Rebelo Moreira, and Renata Silva Souza Guizzardi. Personal Health Train Architecture with Dynamic Cloud Staging. *Sn Computer Science*, 4:null, 2022.

[SSD23] Chang Sun, J. V. Soest, and M. Dumontier. Analyze Decentralized Personal Health Data Using Solid, Digital Consent, and Federated Learning. 2023.

[SSN+24] Kai Schmid, Jannik Sehring, Attila Németh, Patrick N. Harter, Katharina J. Weber, Abishaa Vengadeswaran, Holger Storf, Christian Seidemann, Kapil Karki, Patrick Fischer, Hildegard Dohmen, Carmen Selignow, Andreas Von Deimling, Stefan Grau, Uwe Schröder, Karl H. Plate, Marco Stein, Eberhard Uhl, Till Acker, and Daniel Amsel. DistSNE: Distributed computing and online visualization of DNA methylation-based central nervous system tumor classification. *Brain Pathology*, 34(3):e13228, May 2024.

[ST22] Banerjee S and Bishop Trp. dsSynthetic: synthetic data generation for the DataSHIELD federated analysis system. *PubMed*, 2022.

[SZT+19] Zhenwei Shi, Ivan Zhovannik, Alberto Traverso, Frank J. W. M. Dankers, Timo M. Deist, Petros Kalendralis, René Monshouwer, Johan Bussink, Rianne Fijten, Hugo J. W. L. Aerts, Andre Dekker, and Leonard Wee. Distributed radiomics as a signature validation study using the Personal Health Train infrastructure. *Scientific Data*, 6(1):218, October 2019.

[TL22] Fabrice Tocco and Laurent Lafaye. Data Platform Solutions. In Boris Otto, Michael Ten Hompel, and Stefan Wrobel, editors, *Designing Data Spaces*, pages 383–393. Springer International Publishing, Cham, 2022.

[WE18] Isabel Wagner and David Eckhoff. Technical privacy metrics: A systematic survey. *ACM Computing Surveys*, 51(3):1–38, June 2018.

[ZGA+23] Rui Zhao, Naman Goel, Nitin Agrawal, Jun Zhao, Jake Stein, Ruben Verborgh, Reuben Binns, Tim Berners-Lee, and Nigel Shadbolt. Libertas: Privacy-Preserving Computation for Decentralised Personal Data Stores, September 2023. arXiv:2309.16365 [cs].

[ZHM+24] Daniela Zöller, Christian Haverkamp, Adeline Makoudjou, Ghislain Sofack, Saskia Kiefer, Denis Gebele, Michelle Pfaffenlehner, Martin Boeker, Harald Binder, Kapil Karki, Christian Seidemann, Bernd Schmeck, Timm Greulich, Harald Renz, Stefanie Schild, Susanne A. Seuchter, Dativa Tibyampansha, Roland Buhl, Gernot Rohde, Franziska C. Trudzinski, Robert Bals, Sabina Janciauskiene, Daiana Stolz, and Sebastian Fähndrich. Alpha-1-antitrypsin-deficiency is associated with lower cardiovascular risk: an approach based on federated learning. *Respiratory Research*, 25(1):38, January 2024.