AuthApp – Portable, Reusable Solid App for GDPR-compliant Access Granting

Andreas Both[1,2], Thorsten Kastner[1], Dustin Yeboah[1], Christoph Braun[3], Daniel Schraudner[4], Sebastian Schmid[4], Tobias Käfer[3], and Andreas Harth[4]

[1] DATEV eG, Nuremberg, Germany
[2] Leipzig University of Applied Sciences, Leipzig, Germany
[3] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[4] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract. The Solid (Social Linked Data) technology family was developed to provide the foundation for Data Sovereignty in the context of web applications. The advantage of this innovative approach is the opportunity to dynamically bind an identity to a Solid application and a user-specific Solid data store (Solid Pod). These three basic components can be combined dynamically, allowing users to share their data with an application while retaining full control of the data in self-managed Solid Pods. This paper presents a prototype of a web-based user interface to grant access to data in a Solid Pod. To enable a dynamic binding into Solid-driven environments, we made the implementation available as a Solid application – AuthApp – with a specific focus on allowing users to configure the data access granting efficiently. To comply with data protection regulations, in particular Europe's GDPR, we extended the standard to include the validation of the purpose of data sharing. Unlike previous work, we also make full use of robust technologies to avoid the need to copy or store data outside the personal context, meaning all data remains under the user's control and so does the AuthApp.

Keywords: Solid · Authorization · Access Control · Data Sharing · Access Granting · Zero Trust · Data Sovereignty

1 Introduction

The technologies around Solid were designed to enable standards for connecting web user identities, web services, and data accessible via HTTP introduced by Berners-Lee et al.[5]. This so-called web decentralization project was intended to give users more control over their personal data. The core idea behind Solid is to create a decentralized web platform where users can store their data in Solid Pods (Personal Online Data Stores) and have more control over who can access and use their information (cf. [10]): "Pods are like secure web servers for data.". In a Solid-based architecture, applications and services request permission from users to access their data stored in Pods, providing users with data

[5] cf. https://solidproject.org/

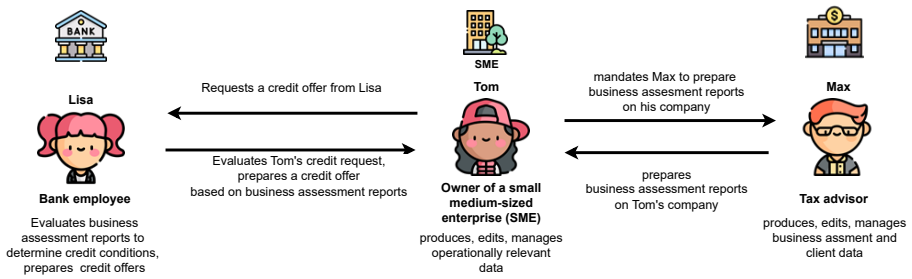


Fig. 1: Use case and personas

privacy and control. Using this approach users can share their data on-demand with Solid-based applications while the applications should only use the user's Pod to store data, s.t., even the applications' output would be under the control of the users. Solid originally was intended to enable end users to stay in control while sharing data via social media or with other end users (e.g., via a private chat). Building on top of machine-readable data representations using the Resource Description Framework (RDF)[6] it provides many benefits, in particular, a flexible metadata model and standardized, completely web-based operations. However, Solid's elaborated vision of interoperability would also be beneficial for organizations, as many of them intend to develop data-driven products or set up agile data chains to collaborate with their partners. This has already led to many use cases in different domains (e.g., [21,4,20]).

In our motivating example (cf. Figure 1), we sketched a real-world use case where a company intends to get a loan from a bank. However, since the company uses a service provider – a tax advisory firm – for financial transactions, not all data for a loan application is stored directly in the company. Therefore, the expected approach would be to establish an ad-hoc data chain where a request to the tax advisor from the company would be processed by the tax advisory office, s.t., the company can hand the data over to the bank bundled with the configuration of the requested loan. Finally, the bank would commit or reject the loan request, expressed again by shared data items. Hence, here a typical B2B data chain would be established where the data-sharing process would be in the center of activity. In contrast, data exchange in the end-user sector is much simpler and more centralized. For these operations, the Solid Application Interoperability [11] (`INTEROP`[7]) was defined that provides a flexible policy model for expressing access grants provided as an RDF vocabulary [11]. However, data sharing between organizations must meet higher standards than in the private sector, including GDPR [18] requirements and additional traceability rules as well as security and business confidentiality requirements and last but not least low integration costs. Implementing such functionality is cumbersome

[6] W3C RDF Working Group
[7] `PREFIX interop: <http://www.w3.org/ns/solid/interop#>`

and error-prone. However, as Solid is built on clear web-based communication standards, it should be possible to implement a Solid app – AuthApp – that can be dynamically integrated into another Solid app to ensure a compliant access granting process. Nevertheless, integrating such functionality into an external app would only be possible if a zero-trust [17] environment can be ensured. While following the Solid principles, the AuthApp should still ensure a zero-trust architecture as no communication to a system outside the current scope is done and – in contrast to previous works – no data is stored outside the users' Solid Pod. In this paper, we will address these requirements following research questions:

RQ1 Is it possible to establish a standalone, pure Solid web application that is pluggable to Solid applications while managing the access granting process?

RQ2 What limitations of the `INTEROP` specification exist w.r.t. the requirements of B2B scenarios?

The paper is structured as follows. In the next section, we will provide an overview of the related work. In Section 3, the requirements and expected functionalities are described derived from this information, the conceptional decisions are described derived. The implementation of the corresponding application is described in Section 4, followed by a discussion of the implications and learnings (Section 5). Section 6 concludes the paper and outlines future work.

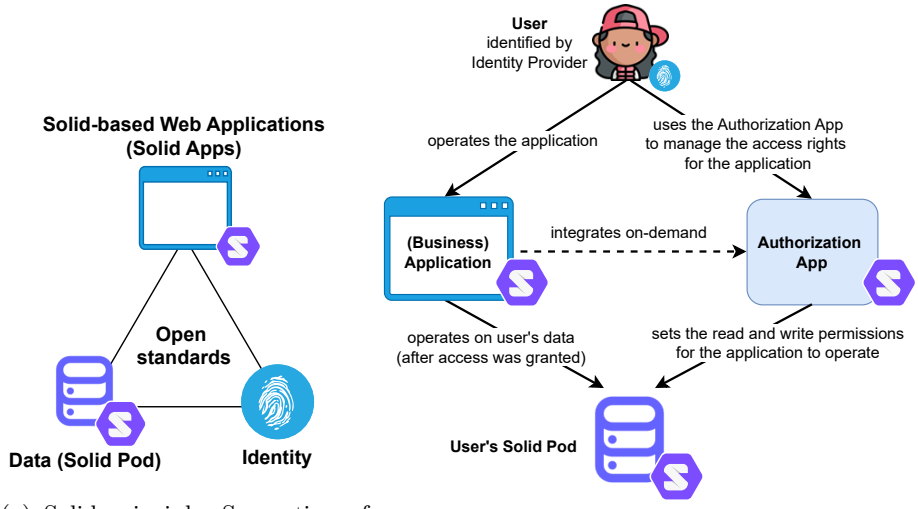
2 Related Work

Establishing data sovereignty in web-based data-driven ecosystem is the goal of the Solid movements [14,12,19,15]. The *Solid platform*, specifications and technologies aim for a safe, decentralized web where users are in complete control of their data.

Solid[8] is a project that aims to change the way web applications work today. It strives for true data ownership and improved data privacy. When using the web, people should have the freedom to choose where their data is stored and who can access it. This is achieved by decoupling identity, data and application connected via open standards. Figure 2a depicts the *Solid principle of connecting identity, data and applications through open standards*, i.e., they can be exchanged and reconnected on-demand. This core feature promises to support many use cases in a web-based data-driven ecosystem.

The Solid community has defined a list of inclusion criteria that must be met and a list of exclusion criteria that must not be met in order to be classified as a Solid compatible web application[9] (short: *Solid app*). In principle, any web application that complies with these guidelines is Solid compatible: Users must be able to log in using their WebID[10] and refer to the Identity Provider of their

[8] cf. https://solidproject.org/
[9] cf. https://solidproject.org/apps
[10] cf. https://www.w3.org/wiki/WebID



(a) Solid principle: Separation of data, application and identity.

(b) Interaction model.

Fig. 2: Conceptional view

choice in case of identification is required. Data consumed and generated by an app should be fetched, resp. stored in Solid Pods.

A *Solid Pod* is a personal or group-based storage space in which data is stored in a standardized format. Each Pod is like a secure web server for personal information. Unlike centralized cloud services, Solid Pods are hosted decentrally. This means that the data is not necessarily stored in a single location, but can be distributed across different servers. The owners of a Solid Pod have full control over their data. They can determine who has access to their information and which applications are allowed to access it. They use standardized formats and protocols[11] to ensure interoperability between different applications. This makes it possible for the same data to be used by different apps. Solid uses a decentralized extension of OpenID Connect to authenticate the identity of users. This ensures that only authorized persons can access the data.

Shape Trees[12][1] are a contribution from the Solid Community Group to foster data interoperability on the application level. While RDF provides interoperability through shared vocabularies and data formats, Shape Trees allow defining schemas to validate the combination of RDF triples. They can clearly define the resource organization in a Pod using RDF and shapes, providing a higher level of abstraction. They are used in Data Registrations (see Chapter 6 Solid Interoperability Specification [11]) to structure data in a Pod. This allows Shape Trees to guide applications and users by determining where data can be

[11] cf. https://solid.github.io/specification/
[12] cf. https://shapetrees.org/

written to and read from. Conversely, when used for formulating data *Access Needs* (see Chapter 8), Shape Trees allow a data requester to accurately describe the resources they wish to access in a Pod in a machine-readable format. By comparing the Shape Trees given in an Access Need with the Shape Trees in the Data Registrations of the considered Pod, the relevant resources in the Pod can be accurately located, and access grants can be set accordingly (if the user wishes to do it).

Although several specifications were defined several years ago, research in the field of Solid has only recently gained momentum. Because of the claim to general validity, establishing valid specifications that can fulfill all needs of end-users, business, organizations, and administrations is challenging.

Several publications focus on the technical foundations of Solid. For example, in [13] the decentralized verification of data with confidentiality was addressed using Blockchain technology. In [9] and [8] the communications/notifications in the Solid ecosystem are considered.

Other researchers focussed on particular use cases. In [20] the use case of machine-to-machine sales contracts were considered where Solid Pods are used for data storage. The data environment for building information modelling (BIM) using Solid was the topic of [21]. Self-verifying Web Resource Representations using Solid, RDF-Star and Signed URIs is the topic of [7].

In the context of the data sharing process – which is the most crucial operation in an ecosystem built for data sharing – the `INTEROP` [11][13] was defined for detailing how Social Agents and Applications can safely share and interoperate over data in Solid Pods. To the best of our knowledge, only one previous publication was dedicated to implement and validate the `INTEROP` specification. Bailly et al. [5] dedicated their work to the understandability and usability of the data sharing process for end-user. In contrast, our work aims for validating the dynamic, self-controlled data sharing process regarding the requirements of business and their data value chains.

3 Concept and Architecture

We now describe the requirements for a GDPR-compliant, zero-trust Solid app and the derived architecture while aiming for the goal of portability and reusability. Figure 2b shows the big picture of intended integration of an authorization app that can be deployed outside a business app and therefore provides additional (reusable) functionality on-demand. Hence, the user will have its well-known web application for managing data requests while the business app can reduce the implementation and maintain costs. In our use case, the authorization app would be integrated when the users need to access others data via the used business apps (see Figure 3). Even is this small use case, the number of data accesses and data sharing operations justifies a standalone authorization app to reduce the investments for establishing the three shown business apps.

[13] Editor's Draft, 7 November 2023, https://solid.github.io/data-interoperability-panel/specification/

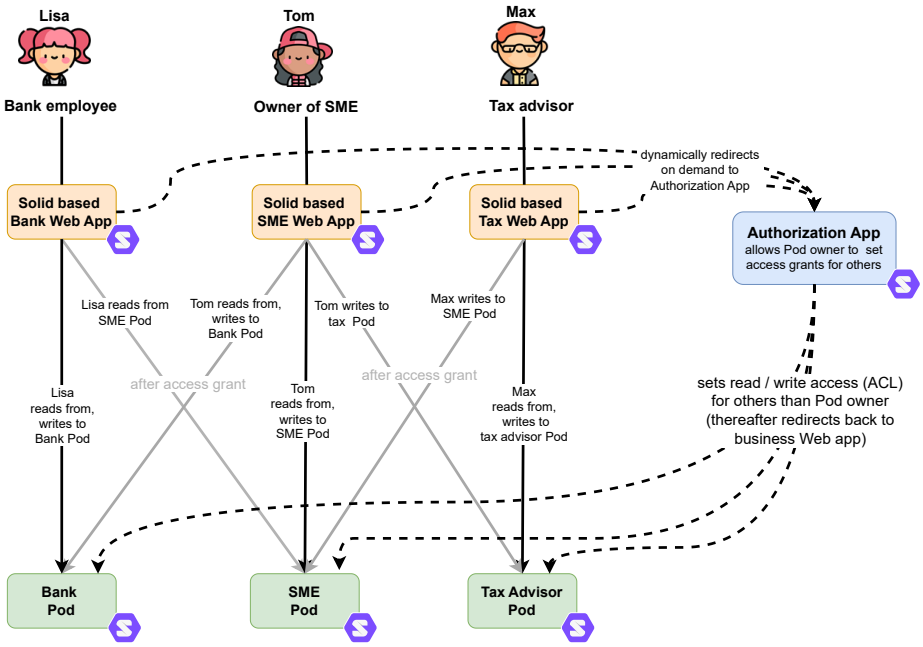


Fig. 3: High-level technical interactions for use case (see Figure 1).

As described earlier, our goal is to provide a portable, reusable, and GDPR-compliant pure Solid app. We collected the following **three requirement groups**, to address these non-functional requirements to be fulfilled by a Solid application (or any web application) dedicated to managing the access granting with a generalized approach.

R1 **Basic Operations** – A generalized web application for managing data sharing operations need to be enabled to support the following operations:
- R1.1 Receive new data sharing requests;
- R1.2 Grant or deny data sharing requests;
- R1.3 Revoke data sharing requests;
- R1.4 Monitor received, existing, denied and revoked data sharing requests.

A typical workflow is shown in Figure 4.

R2 **Data security** – The GDPR regulations [18] demand:
- R2.1 Access rights can be revoked;
- R2.2 Access to objects is granted on a per-object basis (i.e., access to specific web resources can be granted);
- R2.3 Consent must be specific and unambiguous;
- R2.4 The data owner can monitor all access authorizations and thus trace which data is currently being and has been shared with whom and why.

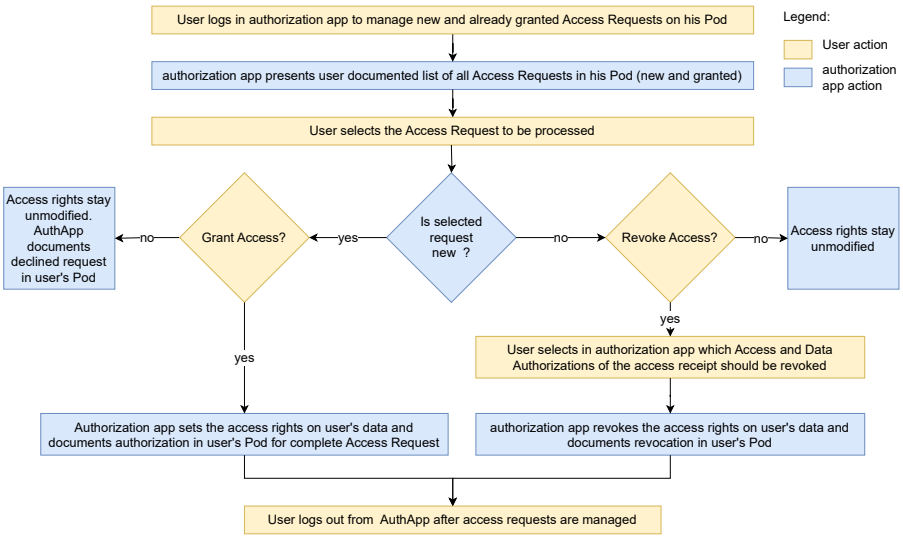


Fig. 4: AuthApp workflow: Operations for managing data access

Additionally, for zero-trust applications the following requirements need to be fulfilled (cf. [17,16]).

- R2.5 The security mechanisms associated with the zero trust architecture will be sufficiently trustworthy to ensure confidence in the zero trust deployment;
- R2.6 The application leaves no trace outside the user scope.

R3 **Portability and Reusability** – We interpret portability as the capability of integrating a generalized authorization application into existing Solid applications. Therefore, we derive the following requirements:

- R3.1 It needs to be possible to integrate the authorization application through configuration;
- R3.2 Interactions with the authorization application need to be implemented using standardized web technologies;
- R3.3 All required information can be accessed with automated processes (i.e., the data is machine-readable).

Note, that although separated within different requirement groups, some requirements are interconnected (e.g., R2.5 and R3.2).

4 Implementation

In this following, we derive our implementation decisions for the Solid app named AuthApp that follows the requirements and the concept described in Section 3. Hence, it is used for evaluating RQ1 and RQ2. The source code is available

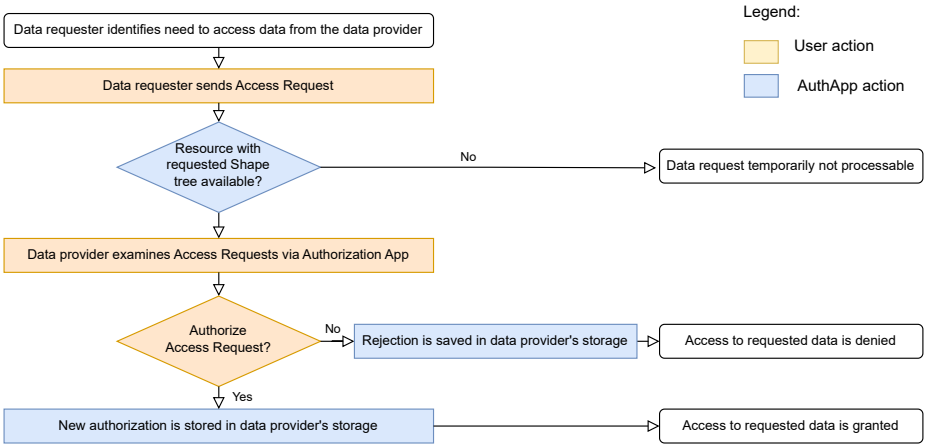


Fig. 5: Process for requesting data access and access granting.

```
@prefix interop:<http://www.w3.org/ns/solid/interop#>.

<#accessAuthorization>
 a interop:AccessAuthorization;
 interop:grantedBy <https://giver/profile/card#me>;
 interop:grantedAt "2024-02-02T11:11:51.331Z"^^xsd:dateTime;
 interop:grantee <https://grantee/profile/card#me>;
 interop:hasAccessNeedGroup
   <https://giver/access-inbox/91511f657390#bwaAccessNeedGroup>;
 interop:hasDataAuthorization
   <https://giver/data-authorizations/53a8739cde00#dataAuthorization>.
```

Fig. 6: Example of Access Authorization (RDF Turtle).

under the open-source MIT license[14]. Following the process shown in Figure 5, it is assumed that users have identified themselves to the application via the standard Solid approach (cf. Section 2).

4.1 `ACL` and `INTEROP` Vocabularies

Solid's specification Web Access Control (WAC) [6] and its corresponding Access Control List (`ACL`) are part of the Solid platform. Although not stable[15] Solid servers (e.g., Community Solid Server[16], Node Solid Server[17]) need to implement these standards to comply with the Solid ecosystem. Therefore, we conclude that for defining access rules, only ACLs [6] have to be used. They provide a

[14] https://github.com/DATEV-Research/Solid-authorization-app
[15] currently: Version 1.0.0, Editor's Draft, 2023-11-06
[16] https://github.com/CommunitySolidServer
[17] https://github.com/nodeSolidServer

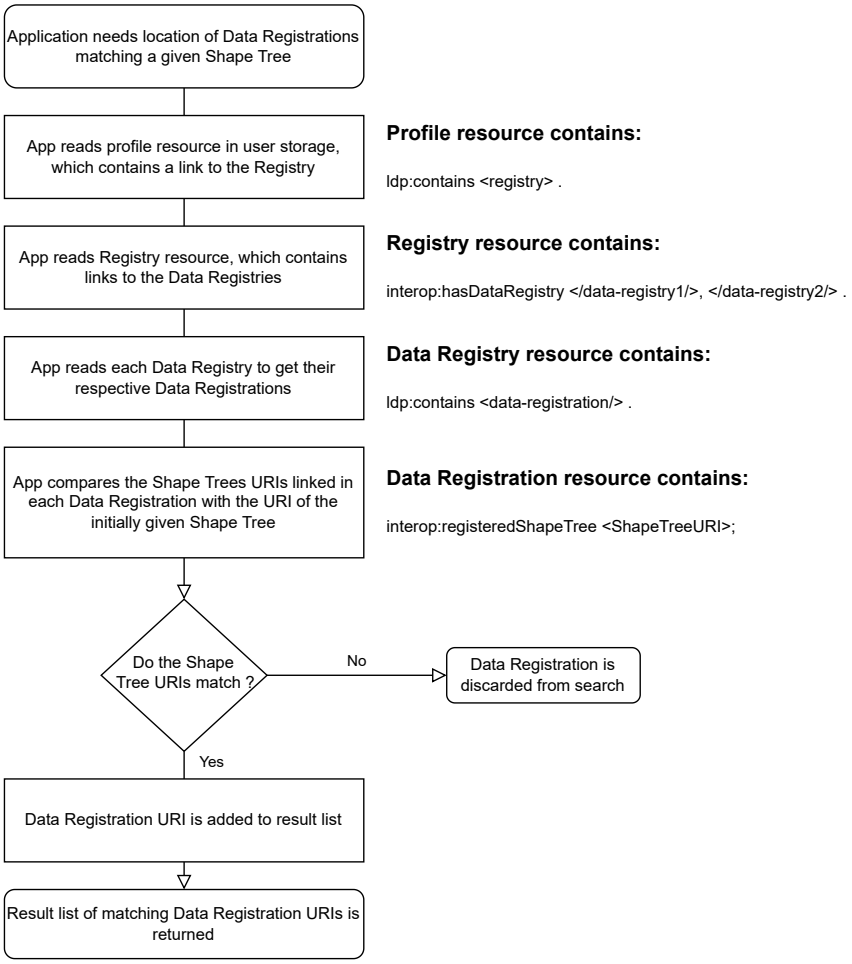


Fig. 7: Process for requesting data using shape trees.

standardized vocabulary to define the access to web resources of a Solid Pod (i.e., read-write operations). Hence, R1.2, R1.3, R2.1, and R2.2 are fulfilled.

Additionally, the `INTEROP` specification [11] describes how social agents and applications can safely share and interoperate over data in Solid Pods. It is fairly new[18] and needs to be considered as work in progress. We used the `INTEROP` specification anyway as a foundation for our implementation (Figure 6 shows an Access Authorization defined using the `INTEROP` vocabulary). However, some parts cannot be used directly as the definition is not stable or not even available. In particular, an *access inbox*, that was defined earlier intended to provide

[18] Editor's Draft, 7 November 2023

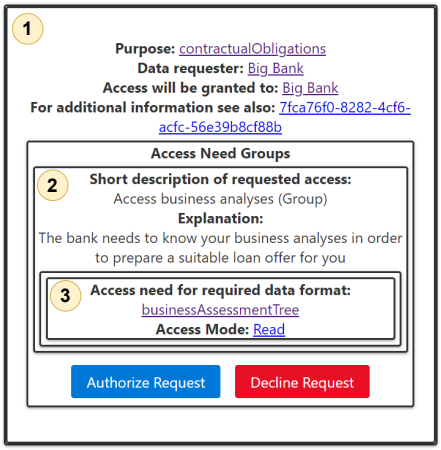
an endpoint for receiving data requests, was removed later[19] (due to possible spamming of public endpoints) in favor of Social Agent Invitation and Identity Verification which is up to now not completely specified and therefore an open issue[20]. So, we decided to follow the previous access-inbox definition. In our implementation, we use a standardized endpoint described using the property `pod:hasAccessInbox`[21], which provides a reference to the access inbox where clients can send data requests using HTTP `POST` (cf. R3.2). By intention the Pod's content is not accessible for a future collaborator, i.e., the data sharing request needs to contain a `shapetree` describing the required data, s.t., the AuthApp can automatically evaluate what web resources of the data provider's Pod would fulfill the need (the process is visualized in Figure 7). For this purpose, we used Data Registrations (see [11], Chapter 6) to identify available resources that could be shared. As a result, the data requests shown to the user will point to specific web resources, although the data requested (cf. R2.2 and R2.3).

Following the `INTEROP` specification, a data sharing request would contain a purpose defined as text. We considered this as not suitable for fulfilling R2.3 as text can be ambiguous and misleading. Additionally, in a multilingual B2B environment, one language-specific text might not be understandable by particular persons. Therefore, we decide to extend the data request by a IRI property, s.t., web resource can be referred that might be providing additional formal statements, multilingual representation etc. Hence, R1.1 and R2.3 is fulfilled.

4.2 User Interface

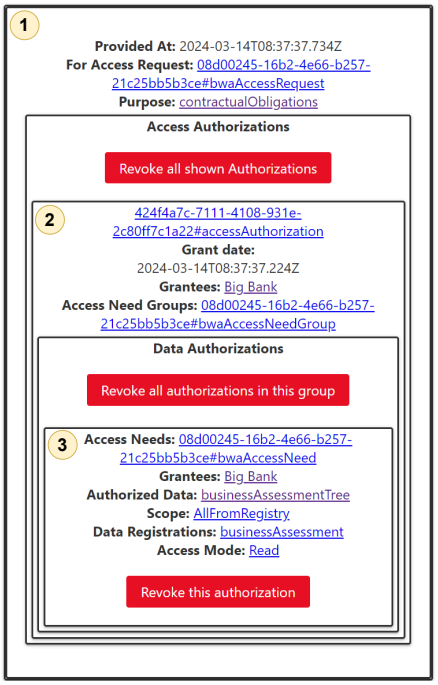
The main activity in the AuthApp has to be the management of incoming data requests and existing data shares (cf. R1.4). A screenshot of our white-label implementation is shown in Figure 8a based on the use case described earlier (i.e., the screenshot is taken from the context of the S.M. Enterprise). First, the basic information (1) is shown that is providing the context for all following information. Given the importance of the purpose, it is shown first together with the basic information of the data requester, the beneficiary of the data sharing. In the encapsulated block (2), the Access Need Groups showing the short description of the requested access and an additional explanation for the particular Access Need Group. Last (3), the specific data request information is shown, i.e., the required data format (`shapetree`) and the Access Mode[22]. Note that following the linked data principles, the corresponding identifiers are also provided via a link (IRI), s.t., users can collect additional information if needed. As Solid is following the Linked Data Platform [3] principles, all information about data sharing and corresponding requests can be represented in RDF. If done so, the corresponding data is machine-readable and can be represented in a monitoring UI. Figure 8 show screenshots for granting and declining data requests, as well as monitoring revoked accesses. Hence, R1.4 is fulfilled.

[19] https://github.com/solid/data-interoperability-panel/issues/280
[20] https://solid.github.io/data-interoperability-panel/specification/#access-request
[21] `@prefix pod : <https://sme.solid.aifb.kit.edu/> .`
[22] Note, that we used the same text labels as specified in the `INTEROP` specification



(a) Access granting / declining.

(b) Revoked access



(c) Revoking detailed access rules

Fig. 8: Screenshots of AuthApp

4.3 Integration

By design, the AuthApp is not integrated directly into the business applications. Hence, it needs to be identifiable by the application. For this purpose, the RDF property `pod:hasAccessInbox` (range: IRI pointing to an AuthApp implementation accessible for the user) is used (cf. Section 4.1). Hence, a business app just needs to redirect the web browser to the corresponding IRI to provide the user with the functionality of data sharing. Hence, R3.2 is fulfilled, s.t., any Solid app might dynamically be connected to the AuthApp. Note, that this mechanism is not customized for our implementation. Instead, it is a generalized solution to integrate other implementations of authorization apps or agents.

4.4 Solid Pod Structure

In the Solid ecosystem, RDF is used to represent all data in a generalized form, providing the benefit of establishing a lingua franca for data representation.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix space: <http://www.w3.org/ns/pim/space#>.
@prefix interop: <http://www.w3.org/ns/solid/interop#>.
@prefix pod: <https://sme.solid.aifb.kit.edu/>.
@prefix profile: <https://sme.solid.aifb.kit.edu/profile/>.

profile:card a foaf:PersonalProfileDocument ;
             foaf:maker <#me> ;
             foaf:primaryTopic <#me> .

<#me> a foaf:Person, foaf:Organization, interop:SocialAgent ;
      interop:hasRegistrySet profile:registry#set ;
      interop:hasAuthorizationAgent <https://authApp/> ;
      interop:hasAccessInbox pod:access-inbox .

<#set>  a interop:RegistrySet;
        interop:hasAgentRegistry </agents/> ;
        interop:hasAuthorizationRegistry </auth-registry/> ;
        interop:hasDataRegistry urn:DataContainer1,
            urn:DataContainer2, urn:DataContainer3 .
```

Fig. 9: Formal definition of configuration (RDF Turtle format) including the access-inbox definition, 3 data registries, the authorization agent definition, etc.

However, the data must be stored and findable. To fulfil R2.6 we decided to use only the user's Pod and did not implement any backend (in contrast to [5]). Hence, only components of the Solid platform are used, i.e., R2.5 is fulfilled too.

As all information is stored, endpoints are required that enable web clients to access the expected RDF. Therefore, we defined that the AuthApp assumes the following containers to be present on the top level of the user's Pod:

- `/access-inbox/` contains all Access Requests;
- `/authorization-archive/` contains outdated or withdrawn Access Authorizations;
- `/authorization-receipts/` contains the receipts of authorized Access requests;
- `/authorization-registry/` contains the current Access Authorizations;
- `/data-authorizations/` contains the current Data Authorizations.

To maintain a clear access point for these authorization-relevant information, we store the configuration data to the user's Solid Webprofile [2] inside the Solid Pod (i.e., the authorization app can be customized for each user and there is no need for centralizing this configuration within an organization). Hence, it is under complete control of the user and also centralized to enable any application to descriptively access such information. Note, that for all three participants in our use case (cf. Figure 3) and their individual business applications, the same instance of the AuthApp was used, validating its general applicability. An example configuration is shown in Figure 9. Therefore, R3.3 is fulfilled, too.

5 Discussion

Our validation via an implementation shows that the requirements can be fulfilled using the Solid platform specifications. However, in the following, we will discuss several findings and limitations.

In general, our implementation shows that implementing a pure Solid app is possible and thus represents a possible cornerstone of the Solid ecosystem. Nevertheless, several limitations were observed during the validation. We have used the well-known inbox mechanism here to establish an initial connection between the prospective data supplier and the data provider following the `INTEROP` specification. While this enables a sound and safe way to initialize a data chain using just a data structure definition (`shapetrees`), it lacks a GDPR-conform representation of the purpose of the data request. However, the inbox mechanism follows the GDPR principle of data thrift, avoidance, and minimization as it does not share any information about the Pod of the future data provider. Additionally, the `INTEROP` specification does not specify how to deal with withdrawn or outdated authorizations. Actually, it is not defined how to represent a withdrawn Access Authorization (and how it differs from a valid one). An authorization app can only recognize that a data access grant is outdated by the fact that a new data release description is created with the reference that this replaces the old one via the property `interop:replaces`. Hence, the withdrawn permit does not contain any indication that it has been replaced by a new one. Therefore, extensions of the `INTEROP` specification (or an additional specification) are required to overcome these challenges, which might otherwise render the functionality unusable.

Although RDF is well suited for representing data in a machine-readable form, it needs to be validated if it is acceptable that the stored data about the granted and denied requests (including requested data formats, requesters, and beneficiaries) can be changed by the user. From a legal perspective, there might be a need for a write-only data container, s.t., this data stays consistent and is immutable. Furthermore, a standardization of the vocabulary for describing the structure of Solid Pod (cf. Section 4.4) seems to be reasonable to prevent implementation-specific behavior that might reduce the portability.

Finally, as useful and convenient as an external authorization app is for the user, there might raise the issue that users expect that this is the only way how they handle data requests to their Pod. However, applications might still internally create `ACL` without redirecting to the authorization app. This might have the implication of different, inconsistent, or even non GDPR-conform data representations which might become problematic in a well-regulated business context and therefore hindering the adoption of the Solid technologies. From this observation, one might derive the need to extend the Solid protocol, s.t., users or organizations can restrict the authorization apps that are allowed to change such data in their Solid Pods.

6 Conclusions

In this paper, we have presented an implementation of a Solid application that can dynamically bind into Solid environments, s.t., the data access granting processes don't have to be implemented by each application, thus, reducing the implementation costs of Solid applications. We provided two major contributions: (1) We collected the requirements for business-driven use cases where a web-based data chain should be established, including a major subset of GDPR conformance; (2) We provided a generalized implementation that can be used in the Solid ecosystem for managing data requests and monitoring data shares of a Solid Pod. In contrast to previous work, our approach does not require additional components that needs to be trusted. As the application is running completely in the user context, the zero-trust requirements are also fulfilled. Additionally, we extend the standard by introducing a Linked-Data-based representation of GDPR-compliant data access purposes that will enable automatic logical validation of access requests in the future. To the best of our knowledge, this is the first implementation providing all of these features. Hence, we have not just validated the Solid concepts but also provided a usable implementation that can be seen as a cornerstone for populating a Solid-based web-based software ecosystem. In general, our implementation proves the feasibility of the Solid standards for establishing dynamic applications, where autonomously granting access should also be possible for non-expert users while the required data is stored in the private Solid Pod only, hence, providing a maximum of data protection and sovereignty.

In future work, to increase the safety of data chains reasoning capabilities should be integrated into authorization apps as well as into the `INTEROP` specification as currently the data formats (`shapetrees`) of the data requests and the Data Registrations are not validated semantically (i.e., it is assumed that they are matching by definition). Additionally, establishing a protocol or immutable data container for storing the current configuration and history of data shares would create an increase level of safety from the legal perspective.

Acknowledgments This work has been supported in part by the German ministry BMBF under grant 16DTM107B (*MANDAT*).

References

1. Shape Trees Specification, https://shapetrees.org/TR/specification/
2. Solid WebID Profile, https://solid.github.io/webid-profile/
3. Linked data platform 1.0 (2015), https://www.w3.org/TR/2015/REC-ldp-20150226/
4. Abid, A., Cheikhrouhou, S., Kallel, S., Jmaiel, M.: Novidchain: Blockchain-based privacy-preserving platform for COVID-19 test/vaccine certificates. Software: Practice and Experience **52**(4), 841–867 (2022)
5. Bailly, H., Papanna, A., Brennan, R.: Prototyping an end-user user interface for the Solid Application Interoperability Specification under GDPR. In: The Semantic Web. pp. 557–573. Springer Nature (2023)

6. Berners-Lee, T., Story, H., Capadisli, S.: Web access control. Version 1.0.0, Editor's Draft, 2023-11-06 (2023), https://solid.github.io/web-access-control-spec/
7. Braun, C.H.J., Käfer, T.: Self-verifying web resource representations using Solid, RDF-star and signed URIs. In: The Semantic Web: ESWC 2022 Satellite Events. pp. 138–142. Springer, Cham (2022)
8. Braun, C.H.J., Käfer, T.: Web push notifications from Solid Pods. In: Web Engineering. pp. 487–490. Springer International Publishing, Cham (2022)
9. Capadisli, S., Guy, A., Lange, C., Auer, S., Sambra, A., Berners-Lee, T.: Linked data notifications: A resource-centric communication protocol. In: The Semantic Web. pp. 537–553. Springer International Publishing, Cham (2017)
10. Dedecker, R., Slabbinck, W., Wright, J., Hochstenbach, P., Colpaert, P., Verborgh, R.: What's in a Pod?–a knowledge graph interpretation for the Solid ecosystem. In: 6th Workshop on Storing, Querying and Benchmarking Knowledge Graphs (QuWeDa) at ISWC 2022. pp. 81–96 (2022)
11. Justin Bingham, Eric PrudHommeaux, E.P.: Solid Application Interoperability. W3C Editor's Draft. Nov. 2023, https://solid.github.io/data-interoperability-panel/specification
12. Mansour, E., Sambra, A.V., Hawke, S., Zereba, M., Capadisli, S., Ghanem, A., Aboulnaga, A., Berners-Lee, T.: A demonstration of the Solid platform for social web applications. In: Proceedings of the 25th International Conference Companion on World Wide Web. p. 223–226. WWW '16 Companion (2016). https://doi.org/10.1145/2872518.2890529
13. Ramachandran, M., Chowdhury, N., Third, A., Domingue, J., Quick, K., Bachler, M.: Towards complete decentralised verification of data with confidentiality: Different ways to connect Solid Pods and Blockchain. In: Companion Proceedings of the Web Conference 2020. p. 645–649. WWW '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3366424.3385759
14. Sambra, A.V., Mansour, E., Hawke, S., Zereba, M., Greco, N., Ghanem, A., Zagidulin, D., Aboulnaga, A., Berners-Lee, T.: Solid: a platform for decentralized social applications based on linked data. MIT CSAIL & Qatar Computing Research Institute, Tech. Rep. (2016)
15. Seneviratne, O., van der Hiel, A., Kagal, L.: Tim Berners-Lee's Research at the Decentralized Information Group at MIT, p. 201–213. ACM, 1 edn. (2023)
16. Shore, M., Zeadally, S., Keshariya, A.: Zero trust: The what, how, why, and when. Computer **54**(11), 26–35 (nov 2021). https://doi.org/10.1109/MC.2021.3090018
17. Stafford, V.: Zero trust architecture. NIST special publication **800**, 207 (2020), https://doi.org/10.6028/NIST.SP.800-207
18. The European Parliament and the Council of the European Union: Regulation (EU) 2016/679 (General Data Protection Regulation) GDPR, https://gdpr-info.eu/
19. Verborgh, R.: Re-decentralizing the Web, For Good This Time, p. 215–230. ACM, 1 edn. (2023), https://doi.org/10.1145/3591366.3591385
20. Wang, X., Braun, C.H.J., Both, A., Käfer, T.: Using schema.org and Solid for linked data-based machine-to-machine sales contract conclusion. In: Companion Proceedings of the Web Conference 2022. p. 269–272. WWW '22, Association for Computing Machinery (2022). https://doi.org/10.1145/3487553.3524268
21. Werbrouck, Jeroen and Pauwels, Pieter and Beetz, Jakob and van Berlo, Léon: Towards a decentralised common data environment using linked building data and the Solid ecosystem. In: Advances in ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations (AECO) : Proceedings of the 36th CIB W78 2019 Conference. pp. 113–123 (2019)