Decentralized, Purposeful Online Community Application Framework

by

Serkan Özel

Submitted to the Department of Computer
Engineering in partial fulfillment of
the requirements for the degree of
Bachelor of Science

Undergraduate Program in Computer Engineering
Boğaziçi University
Spring 2020

Decentralized, Purposeful Online Community Application Framework

APPROVED BY:

Prof. Suzan Üsküdarlı ....

(Project Supervisor)

DATE OF APPROVAL: 09.07.2020

ACKNOWLEDGEMENTS

Special thanks to Murat Seyhan for his project that we built upon and my advisor Suzan Üsküdarlı for her great support.

ABSTRACT

Decentralized, Purposeful Online Community Application Framework

A Purposeful online community is a community that has a purpose to do together and in order to achieve that they use online platforms. Several online platforms exist for them to use, mainly general purpose social media applications. However, every community's needs are different and they need a separate application for best usage of the web. We build a decentralized application framework a community using the Solid project that started at MIT. It brings decentralization of user data storage, which means better privacy, reusing existing data and easily proces-able and analyzable data.

ÖZET

Amerkezi, Amaçlı Çevrimiçi Topluluk Uygulama Kütüphanesi

Amaçlı çevrimiçi topluluklar belli bir amacı olan ve bunu başarmak için online platformlar kullanan topluluklardır. Sosyal medya başta olmak üzere çeşitli online platformlar bu amaçlar için kullanılabilmektedir. Ama her topluluğun kendine has ihtiyaçları ve veri yapısı olduğundan topluluklar kendilerine özel uygulama gereksinimi duyarlar. Bu projede amaçlı çevrimiçi topluluklar için bir uygulama kütüphanesi oluşturduk. Bunu yaparken de MIT'de ortaya atılan Solid projesini kulladık. Solid amerkezi halde kullanıcı verisini saklanması, daha iyi gizlilik, verilerin tekrar kullanılabilirliği ve bilgisayar tarafından işlenebilir veri oluşması demektir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

LIST OF ACRONYMS/ABBREVIATIONS

POC	Purposeful Online Communities
WAC	Web Access Control
OIDC	OpenID Connect
LDP	Linked Data Platform
RDF	Resource Description Framework

1. INTRODUCTION AND MOTIVATION

- Human beings are social creatures and they need to help each other to reach their goals. For this purpose, people form communities where people sharing same aims come together.
- Recently, the most effective and fastest medium that carries information needed to be shared between community members has become the internet. Thus, communities started to switch their communication methods.
- In this project, we are aiming to build an application that combines POC(Purposeful Online Community) idea that is proposed in Murat Seyhan's master thesis and the Solid project's principles.
- The following image shows how an Purposeful Online Community application works.

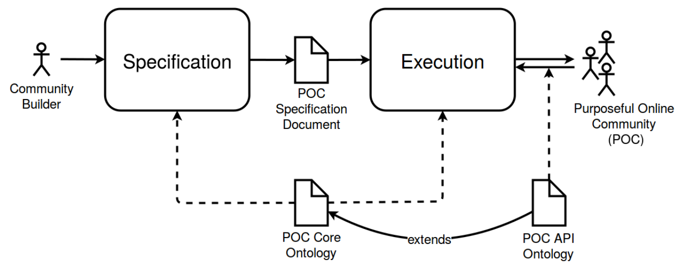
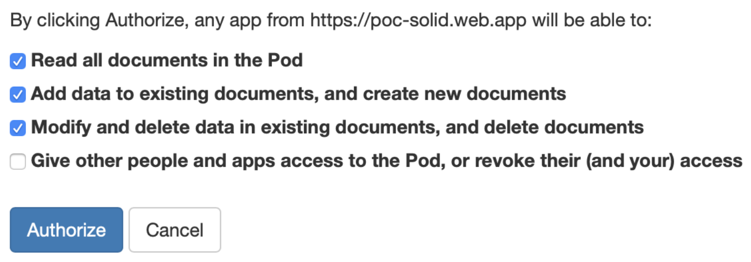


Figure 1.1. The POC Model, source: [1]

- Our project is a prototype application for the future web. Our project helps communities by ensuring they do not lose time and money on communication issues by ensuring they can utilize the web most efficiently for their purposes.
- In this project we will build an application for a story lovers community. The purpose of this community is to collect and share old stories and rate them. Our application ensures they can create a story which is a custom datatype, they can validate an entry is a story and they can rate a story.
- Solid(derived from socially linked data) project is started at MIT University and aims to re-decentralize the web we have today. The origin of the web is decentralized, therefore Solid is a project to re-decentralize it. Solid uses existing W3C standards and protocols. It makes data storage decentralized so that every user

manages his or her data and gives permission to applications or other users. This ensures better privacy, reuse of existing data by newer applications and avoids vendor lock in which means users will be able to switch between application without worrying about losing their data they produce in one application in the another application.

- Solid is a platform, built using the existing web. It allows people to look at the same data with different apps at the same time.



By clicking Authorize, any app from https://poc-solid.web.app will be able to:

- [✓] **Read all documents in the Pod**
- [✓] **Add data to existing documents, and create new documents**
- [✓] **Modify and delete data in existing documents, and delete documents**
- [ ] **Give other people and apps access to the Pod, or revoke their (and your) access**

Figure 1.2. An application trying to get access to a user's pod in Solid

- Solid is guided by the principle of "personal empowerment through data" which is fundamental to the success of the next era of the web. Data should empower people. Imagine if all our current apps talked to each other, collaborating and conceiving ways to enrich and streamline our personal life and business objectives? That's the kind of innovation, intelligence and creativity Solid apps will generate. With Solid, one will have far more personal agency over data - and decide which apps can access it.

2. STATE OF THE ART

- The Web is full of data. Interconnecting the data in the web and giving relations to them is called Semantic Web [2]. The data that is connected is called Linked Data [3].
- Sambra, Andrei Vlad et al's paper, [4] "Solid : A Platform for Decentralized Social Applications Based on Linked Data" is the introduction paper of the Solid project. In Solid, users' data is managed independently of the applications that create and consume this data. The user's data is stored in a Web-accessible personal online datastore (or pod). Solid allows users to have one or more pods from different pod providers, while at the same time enabling users to easily switch between providers. Developers can use Solid protocols, which is based on existing W3C recommendations, for reading, writing and access control of the contents of users' pods. Users can also control access to their data, and have the option to switch between applications at any time. This is paradigm shift in integrating social features into Web applications.
- The W3C recommendation given in 2015 and lead by Malhotra, Arwe, and Speicher's, [5] "Linked Data Platform Specification", discusses how a Linked Data compliant HTTP server should behave and what is the standart way of communicating. It defines set of rules on accessing, updating, creating and deleting Linked Data over HTTP. An LDP server is an HTTP server that applies these rules. An LDP Client is an entity who consumes the data served by an LDP Server. Our application conforms to this specification.
- The representation model of linked data in Semantic Web is the Resource Description Framework(RDF) [6] model. RDF is a model providing a standard way to identify data elements and express relations between them. In RDF, relations are expressed as triples where each triple consists of a subject, a predicate, and an object, each one is represented with a URI [7]. These triples form a graph and there are ways to query information in these graphs. These graphs called RDP graphs. A graph can be encoded in text in several ways such as Turtle [8], JSON-LD [9], and RDF/XML [10].

- The following figure shows how a rdf graph looks like.

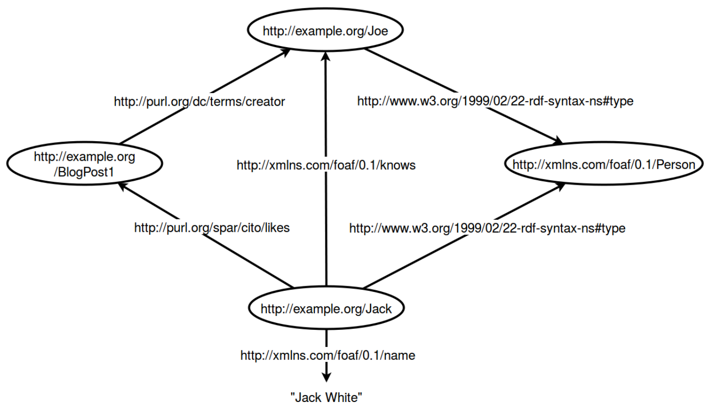
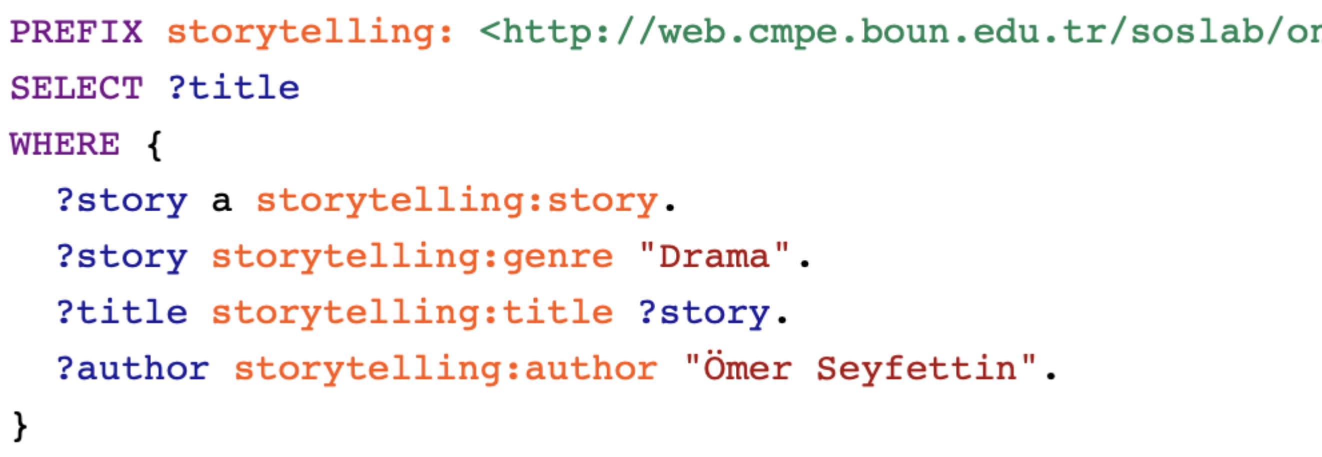

http://example.org/Joe
http://purl.org/dc/terms/creator
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://example.org /BlogPost1
http://xmlns.com/foaf/0.1/knows
http://xmlns.com/foaf/0.1/Person
http://purl.org/spar/cito/likes
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://example.org/Jack
http://xmlns.com/foaf/0.1/name
"Jack White"

Figure 2.1. An RDF graph, source: [1]

- One of the query languages of RDF graphs is SPARQL [11] and we use this language for querying our RDF graph. The query language is similar to SQL and can be used to perform semantic queries in the RDF graph.



```
PREFIX storytelling: <http://web.cmpe.boun.edu.tr/soslab/or
SELECT ?title
WHERE {
  ?story a storytelling:story.
  ?story storytelling:genre "Drama".
  ?title storytelling:title ?story.
  ?author storytelling:author "Ömer Seyfettin".
}
```

Figure 2.2. Example Sparql Query for finding titles of drama storiesby Ömer Seyfettin

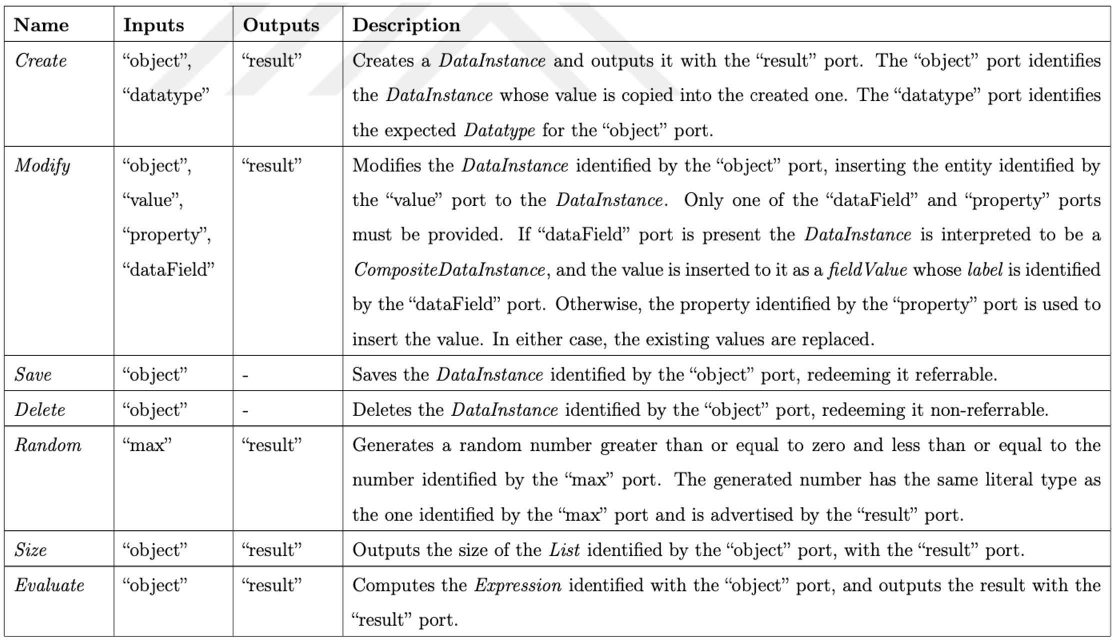
- OWL 2 Web Ontology Language (OWL 2) [12] is a knowledge representation language, enabling specification of domain knowledge in terms of concepts and their relationships. Resulting ontologies are explicit specifications that may be published on the Web, thereby provides means for shared semantics across applications.
- Human computation [13] is a research topic focused on exploiting human intelligence for computational tasks that are computationally difficult for computers,

but easy for humans to solve. Human computation systems form purposeful communities, where the actions of the community members help the community reach certain goals, even though the members may not be aware of what the goals are.

- Regarding using ontologies for creating purposeful online communities, Seyhan's paper [1] is a good one. He creates a prototype that takes a specification file encoded in turtle and creates a Web API based on the specification. This way, a likely non-technical person community builder could specify the tasks, roles, data formats and workflows of the community in Turtle and create an API that can be used as a server.
- The social media application's owner company can use the data by making using the data legal with a terms and conditions document that nobody will ever read. This is a violation of privacy. Facebook had a privacy related issue recently. We will improve privacy by using Solid. Solid makes user data decentralized, leading the data to be fully controlled by them and and offering better privacy.
- Facebook, Twitter, Whatsapp or collaborative documents applications that are used by purposeful online communities today. They do the job in general, however when a complex query is needed to be done to the data, it is impossible to do. The person who does such a search is limited by the social media application's search features. In Solid, the data produced is in Linked Data format, this means any search can be done to the data without need for access to any database. We will improve this aspect with Solid.
- Moreover, the datatypes specific to the community can not be defined in the social platforms that are used today. Our application handles this by letting non-tech savvy users to be able to design their own application.

3. METHODS

- We need to use and develop Seyhan's prototype for creating Purposeful Online Communities and additionally use Solid, Linked Data and Semantic Web principles in our applications.
- Data should not be in a central server. The Solid project states that the data of the users should be on their own PODs leading the their data to be managable by themselves. We should find a way how to communicate with a user or with the server. There may be a server but the data should not reside there. The server may coordinate the clients.
- Since the data is not needed to be stored on a central server, it is enough to build a browser application. Then, each user will be signed in with its own solid pod, then out application will have access to user's data if he or she allows.
- We use Turtle to represent our RDF data. We use the model developed by Murat Seyhan to build our community. The specification consists of writing specification of the community which includes defining data types, workflows, users, roles and so on. We use protege tool to build our community data.
- There are some several example types of steps in the POC core ontology developed by Murat Seyhan. They are CreateStep, DeleteStep, DisplayStep, EvaluateStep, FilterStep, GetStep, InsertStep, ModifyStep, RemoveStep, SaveStep and SizeStep. We need to develop each step's logic in some kind of an execution engine. Each step has input ports and output ports that is predefined. Refer to figures 3.1 and 3.2.
- In POC ontology there are several kind of datatypes. The literal value means simple values, they are from xsd datatypes. Derived Datatypes means, the datatype derive from a simple datatype, it has a additional condition. For example a image that is smaller than 300px can be called smallImage. Finally, CompositeDatatypes are datatypes that has several datafields that each has a datatype. Each datafield has a label that identifies it.
- We need to be able to get input from the user. The user needs to be able to select a data from a list, or he or she needs to be able to enter data to create a data of



Name	Inputs	Outputs	Description
Create	"object", "datatype"	"result"	Creates a *DataInstance* and outputs it with the "result" port. The "object" port identifies the *DataInstance* whose value is copied into the created one. The "datatype" port identifies the expected *Datatype* for the "object" port.
Modify	"object", "value", "property", "dataField"	"result"	Modifies the *DataInstance* identified by the "object" port, inserting the entity identified by the "value" port to the *DataInstance*. Only one of the "dataField" and "property" ports must be provided. If "dataField" port is present the *DataInstance* is interpreted to be a *CompositeDataInstance*, and the value is inserted to it as a *fieldValue* whose *label* is identified by the "dataField" port. Otherwise, the property identified by the "property" port is used to insert the value. In either case, the existing values are replaced.
Save	"object"	-	Saves the *DataInstance* identified by the "object" port, redeeming it referrable.
Delete	"object"	-	Deletes the *DataInstance* identified by the "object" port, redeeming it non-referrable.
Random	"max"	"result"	Generates a random number greater than or equal to zero and less than or equal to the number identified by the "max" port. The generated number has the same literal type as the one identified by the "max" port and is advertised by the "result" port.
Size	"object"	"result"	Outputs the size of the *List* identified by the "object" port, with the "result" port.
Evaluate	"object"	"result"	Computes the *Expression* identified with the "object" port, and outputs the result with the "result" port.

Figure 3.1. Example types of steps with inputports and outputports source: [1]

certain datatype.

- Each workflow consists of multiple steps. Each step is binded to other steps with pipes in a workflow. Pipes can carry data, as in PortPipe, define control dependency or bind a direct value to a step's input or output ports. The users should be able to perform three types of workflows: create a story, validate a story and rate a story.
- There needs to be two composite datatypes in our application. One of the is story and the other one is rate. Story includes details like, genre, author, title and content of the story. Rate includes the story rated, point out of 10 and a comment.
- We need to be able to generate a dynamic form when asking for user input. If the user will select an item from a list, the items should be listed.
- Our system defines some additional rules that is not found in Seyhan's paper. These rules are for simplicity and do not reduce the expressivity of the framework.
  - Human input is allowed in get type step and create type step.
  - The GetStep and CreateStep should have an annotation comment which explains the input needed from the user.

Name	**Inputs**	**Outputs**	**Description**
Insert	“object”, “target”, “index”	“result”	Inserts an item identified by the “object” port to the *List* identified by the “target” port, and outputs the resulting *List* with the “result” port. The “index” port expects an integer value and identifies the position in the list for insertion. If no value is provided for the “index” port, the item is appended to the end of the list.
Remove	“object”, “source”, “index”	“result”	Removes an item from the *List* identified by the “target” port, and outputs the resulting *List* with the “result” port. The item is either identified directly by the “object” port or with its index by the “index” port.
Get	“source”, “index”	“result”	The “result” port is used to output the item having the index, identified by the “index” port, in the *List*, identified with the “source” port.
Filter	“object”, “condition”	“result”	Filters the list identified with the “object” port using the *Expression* identified with the “condition” port, and outputs the resulting list with the “result” port. The *Expression* is computed for each item of the list, replacing the `@item` parameter in the *Expression*. If it computes false or null, the item is excluded.
Identity	-	-	Does not manipulate anything, and is intended to be used for displaying messages to users.

Figure 3.2. Example types of steps with inputports and outputports source: [1]

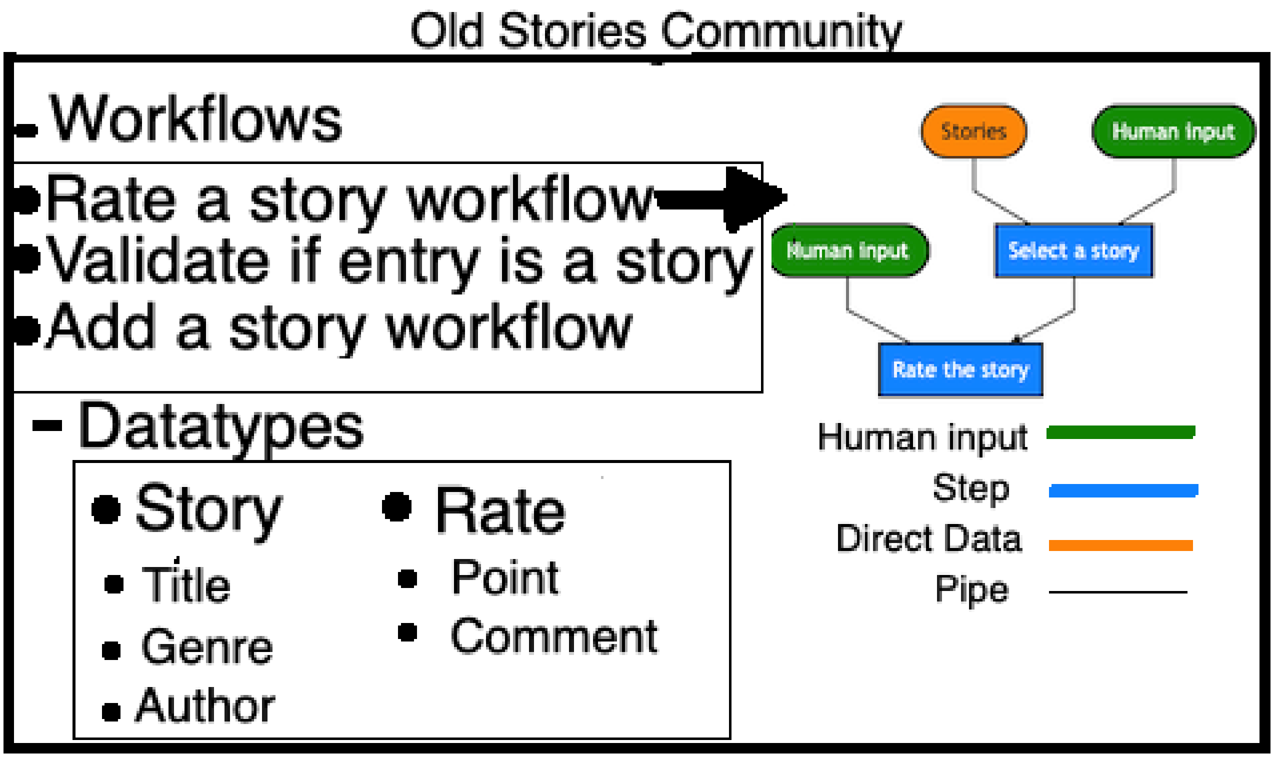
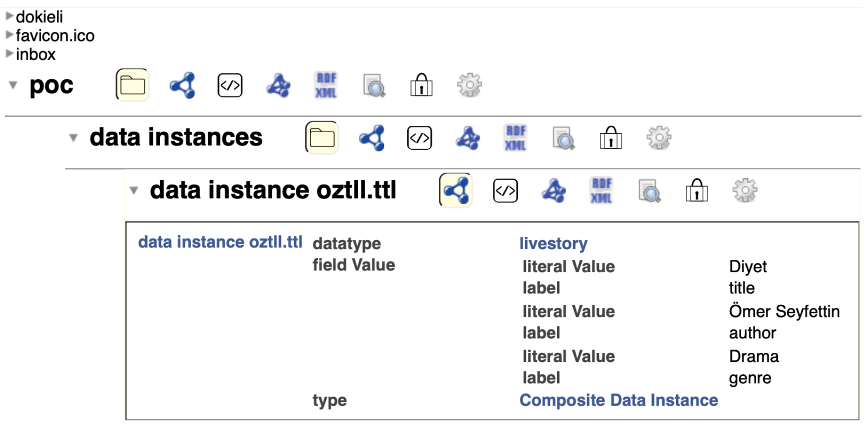


Figure 3.3. Our community’s high level drawing

4. RESULTS

- You see our system architecture in the figure 4.2. Our system uses a Fuseki [14] server that is a server for Linked Data in RDF format. This server is for storing the application specification that is built by community builder which is also a user.
- The Fuseki server also keeps track of all the users in the system. This allows our application to know who to collect data to aggregate in cases needed.
- In Solid, WAC(Web Access Control) [15] system is used. The access control system allows owner of the pod to control who can access his or her own pod. Applications and other users need permission to access, modify and delete data.
- Our application join's the user to poc Web Access Control group. The group allows each user in the group to access their poc folder in Solid pod. This is needed since after login the user and the application together, tries to access Solid pod of other users.
- The login is done with Solid. Solid uses WebID-TLS and WebID-OpenID Connect [16] login flows. We use WebID-OpenID Connect login flow in which users enter their WebID(user.solid.community or anotheruser.solid.provider). After they enter their provide they login with the credentials of their provider and their provider allows us to access to the user's pod, so the data of the user.


▸dokieli
▸favicon.ico
▸inbox
▾ poc
▾ data instances
▾ data instance oztll.ttl

data instance oztll.ttl	datatype	livestory	
	field Value	literal Value	Diyet
		label	title
		literal Value	Ömer Seyfettin
		label	author
		literal Value	Drama
		label	genre
	type	Composite Data Instance	

Figure 4.1. The solid pod and poc folder we use in it. There are other folders for other applications.

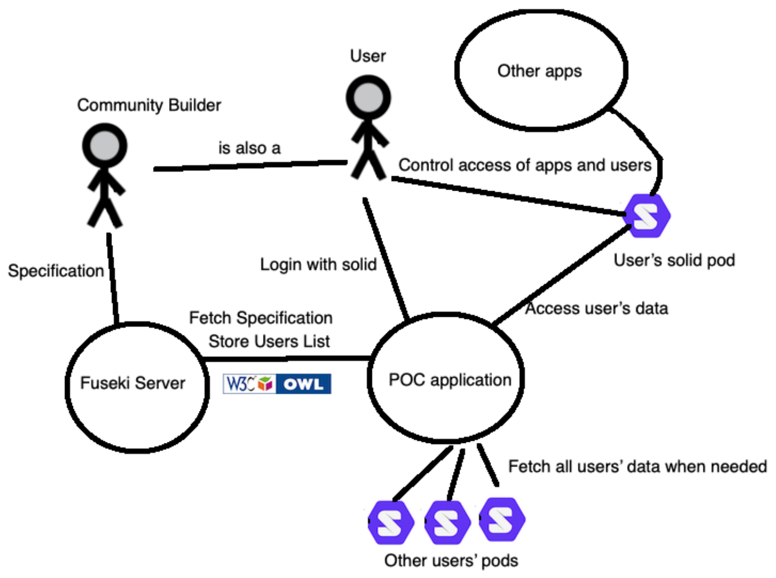


Figure 4.2. The system architecture

- We developed a web browser application satisfying all the requirements in Methods section, in which a user can see his or her workflows, community data, and other users' workflows and data as well.
- Our application first fetches the specification from the Fuseki Server to set it up. According to the specification our application offers workflows and datatypes to the users.
- When a get step is being executed the browser application goes all users' pod and augment the data in all of them and present to the user for selection.
- According to the data selected the user's pod that has created that data is modified if after get step the data is modified.
- If a comment or annotation is done on another user's content, the annotation data instance is stored in annotator user's pod whereas the original data is stays on the creator's pod.
- When a workflow is ongoing, our execution engine sorts the steps according to their dependencies between them and human input dependencies. The priority is to select a step that does not have any dependency to execute next, then the steps that has only human dependency.
- Thus, the workflow cannot have any cycles that is fatal in our system.
- Our framework uses ontologies [17] along with other W3C standards, which means it is structured data and it is possible to make a semantic search like shown with

an example sparql query in state of the art section.

- Privacy protecting data persistence is achieved with use of Solid pods.
- We saw that, non-tech savvy users can create their own application, no coding is being done to change application logic and datatypes.
- Contact and Source Code:
  - Source Code: https://gitlab.com/srknzl/solid-decentralized-web-applications
  - Email: serkan.ozel@boun.edu.tr

5. CONCLUSION AND DISCUSSION

- Our application framework is a step forward for making web better place for non-tech savvy persons by making them create their own application.
- Loops are not possible but not so required for online communities. In purposeful online communities, instead of loops people do the computation.
- Solid is is under development so it requires further work with respect to performance issues. If commercial companies supply Solid servers, they will optimize it and performance will be better.
- Our system takes long to execute workflows. This is due to Solid community pod server. However, not only that, the architecture will still be slower than today's web system. This is due to the excess amount of requests that is done to the Solid server. So applications that do not do a lot of requests perform better.
- Our system uses Solid for everything, from step instances storage to port binding variables. This causes the application to do a lot of requests.

6. FUTURE WORK

- In order to reduce time to fetch all users' data, there is a need for a central observer, keeping track of existing of data for each user.
- A user interface for workflow specification can be built for convenience.
- This application framework, if comes true, will be most influential on end users who need to use computers for a social activity.
- There is a potential market for Solid pod providers instead of today's Dropbox and Google Drive.
- Our requests takes long, the amount of work that is saved to Solid pod can be reduced to step level, and execution of a single level can be made in browser memory.

REFERENCES

1. Seyhan, M., “An Ontology Based Framework for Creating Purposeful Online Communities”, `https://www.cmpe.boun.edu.tr/content/ontology-based-framework-creating-purposeful-online-communities`, accessed at April 2020.

2. Berners-Lee, T., J. Hendler and O. Lassila, “The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities”, *ScientificAmerican.com*, 05 2001.

3. Berners-Lee, T., “Linked Data”, `https://www.w3.org/DesignIssues/LinkedData.html`, accessed at 25 April 2020.

4. Sambra, A. V., E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga and T. Berners-Lee, “Solid : A Platform for Decentralized Social Applications Based on Linked Data”, , 2016.

5. Speicher, S. and J. Arwe, “Linked Data Platform 1.0”, `http://www.w3.org/TR/ldp/`, accessed at 25 April 2020.

6. Cyganiak, R. and D. Wood, “RDF 1.1 Concepts and Abstract Syntax”, `http://www.w3.org/TR/rdf11-concepts/`, accessed at 25 April 2020.

7. Berners-Lee, T., R. Fielding and L. Masinter, *Uniform Resource Identifier (URI): Generic Syntax*, Tech. rep., 01 2005.

8. Beckett, D. and B.-L. T, “RDF 1.1 Turtle”, `http://www.w3.org/TR/turtle/`, accessed at 25 April 2020.

9. Sporny, M. and D. Longley, “JSON-LD 1.1: A JSON-based Serialization for Linked Data”, `http://www.w3.org/TR/json-ld/`, accessed at 25 April 2020.

10. Gandon, F. and S. G, “RDF 1.1 XML Syntax”, `http://www.w3.org/TR/rdf-syntax-grammar/`, accessed at 25 April 2020.

11. Group, T. W. S. W., “SPARQL 1.1 Overview”, `http://www.w3.org/TR/sparql11-overview/`, accessed at April 2020.

12. Group, W. O. W., “OWL 2 Web Ontology Language Document Overview (Second Edition)”, `http://www.w3.org/TR/owl2-overview/`, accessed at 25 April 2020.

13. Law, E. and L. von Ahn, *Human Computation*, Morgan Claypool Publishers, 1st edn., 2011.

14. Foundation, T. A. S., “Apache Jena Fuseki”, `https://jena.apache.org/documentation/fuseki2/`, accessed at 03 July 2020.

15. “Web Access Control (WAC)”, `https://github.com/solid/web-access-control-spec`, online; accessed 27 June 2020.

16. “OpenID Connect”, `https://www.google.com/search?client=safari&rls=en&q=openid+connect&ie=UTF-8&oe=UTF-8`, online; accessed 03 July 2020.

17. “Ontology”, `https://www.w3.org/standards/semanticweb/ontology`, online; accessed 27 June 2020.

APPENDIX A: DATA AVAILABILITY STATEMENT

- Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

APPENDIX B: STANDARDS, LAWS, REGULATIONS AND DIRECTIVES

- HTTP
- RDF
- OWL
- Linked Data
- LDP
- SPARQL
- Semantic Web